# Spezialisierung auf maschinelles Lernen.

## Learning Objectives

- Get familiar with the diagram and components of a neural network
- Understand the concept of a "layer" in a neural network
- Understand how neural networks learn new features.
- Understand how activations are calculated at each layer.
- Learn how a neural network can perform classification on an image.
- Use a framework, TensorFlow, to build a neural network for classification of an image.
- Learn how data goes into and out of a neural network layer in TensorFlow
- Build a neural network in regular Python code (from scratch) to make predictions.
- (Optional): Learn how neural networks use parallel processing (vectorization) to make computations faster.

In diesem Kurs lernen Sie neuronale Netze, auch Deep-Learning-Algorithmen genannt, sowie Entscheidungsbäume kennen. Dabei handelt es sich um einige der leistungsstärksten und am weitesten verbreiteten Algorithmen für maschinelles Lernen. Sie können sie selbst implementieren und zum Laufen bringen. In diesem Kurs erhalten Sie unter anderem auch praktische Ratschläge zum Aufbau maschineller Lernsysteme. Dieser Teil des Materials ist für diesen Kurs einzigartig. Wenn Sie ein praktisches System für maschinelles Lernen aufbauen, müssen Sie viele Entscheidungen treffen, z. B. sollten Sie mehr Zeit mit der Datenerfassung verbringen oder eine viel größere GPU kaufen, um ein viel größeres neuronales Netzwerk aufzubauen? Auch heute noch, wenn ich ein führendes Technologieunternehmen besuche und mit dem Team spreche, das dort an einer Anwendung für maschinelles Lernen arbeitet, schaue ich mir leider manchmal an, was sie in den letzten sechs Monaten gemacht haben, und denke: Mensch, jemand hätte es dir sagen können vielleicht schon vor sechs Monaten, dass dieser Ansatz nicht so gut funktionieren würde.

Mit einigen der Tipps, die Sie in diesem Kurs lernen, hoffe ich, dass Sie zu denjenigen gehören, die diese sechs Monate nicht verschwenden, sondern systematischere und bessere Entscheidungen darüber treffen können, wie praxistaugliches maschinelles Lernen aufgebaut werden kann Anwendungen. Lassen Sie uns also eintauchen. Im Detail sehen Sie dies in den vier Wochen dieses Kurses. In Woche 1 befassen wir uns mit neuronalen Netzen und der Durchführung von Schlussfolgerungen oder Vorhersagen. Wenn Sie ins Internet gehen und die Parameter eines neuronalen Netzwerks herunterladen würden, dass jemand anderes trainiert hat und dessen Parameter im Internet veröffentlicht wurden, dann würde die Verwendung dieses neuronalen Netzwerks zum Treffen von Vorhersagen als Inferenz bezeichnet werden, und Sie haben gelernt, wie neuronale Netzwerke funktionieren Arbeit und wie man in dieser Woche Schlussfolgerungen zieht. Nächste Woche erfahren Sie, wie Sie Ihr eigenes neuronales Netzwerk trainieren. Insbesondere wenn Sie über einen Trainingssatz mit beschrifteten Beispielen X und Y verfügen, wie trainieren Sie dann die Parameter eines neuronalen Netzwerks für sich selbst?

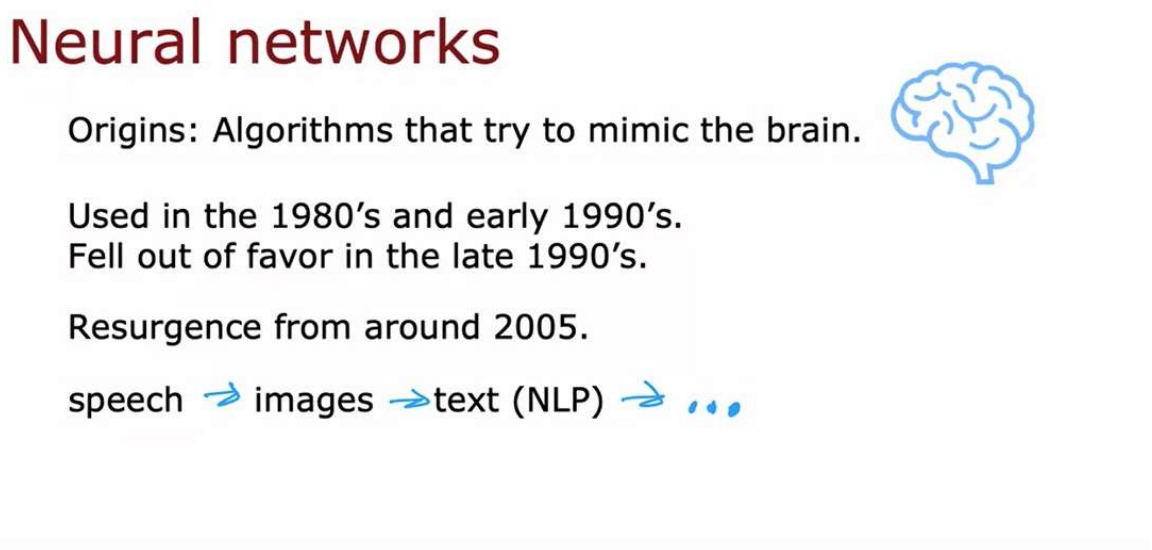
In der dritten Woche gehen wir dann auf praktische Ratschläge zum Aufbau von Machine-Learning-Systemen ein und ich verrate Ihnen einige Tipps, die meiner Meinung nach selbst hochbezahlte Ingenieure, die heute sehr erfolgreich Machine-Learning-Systeme aufbauen, nicht wirklich immer konsequent anwenden können und ich denke, das wird Ihnen helfen, Systeme effizient und schnell selbst aufzubauen. In der letzten Woche dieses Kurses lernen Sie dann etwas über Entscheidungsbäume. Während Entscheidungsbäume in den Medien nicht so viel Aufsehen erregen, gibt es lokal weniger Hype um Entscheidungsbäume im Vergleich zu neuronalen Netzen. Es handelt sich außerdem um einen der am weitesten verbreiteten und sehr leistungsstarken Lernalgorithmen,

von denen ich denke, dass die Wahrscheinlichkeit groß ist, dass Sie sie selbst verwenden, wenn Sie am Ende eine Anwendung erstellen. Lassen Sie uns nun in die neuronalen Netze eintauchen und zunächst einen kurzen Blick darauf werfen, wie das menschliche Gehirn, also das biologische Gehirn, funktioniert. Kommen wir zum nächsten Video.

## Neuronen und das Gehirn

Als neuronale Netze vor vielen Jahrzehnten erfunden wurden, bestand die ursprüngliche Motivation darin, Software zu schreiben, die nachahmen konnte, wie das menschliche Gehirn oder das biologische Gehirn lernt und denkt. Auch wenn sich neuronale Netze, manchmal auch künstliche neuronale Netze genannt, heute ganz anders entwickelt haben, als wir vielleicht denken, wie das Gehirn tatsächlich funktioniert und lernt. Einige der biologischen Motivationen sind immer noch in der Art und Weise vorhanden, wie wir heute über künstliche neuronale Netze oder computergestützte neuronale Netze denken. Werfen wir zunächst einen Blick darauf, wie das Gehirn funktioniert und wie dies mit neuronalen Netzen zusammenhängt. Das menschliche Gehirn, oder vielleicht allgemeiner gesagt, das biologische Gehirn, weist ein höheres oder leistungsfähigeres Maß an Intelligenz auf und alles andere würde bisher auf dem Programm stehen. Neuronale Netze begannen also mit der Motivation, Software zu entwickeln, die das Gehirn nachahmt.

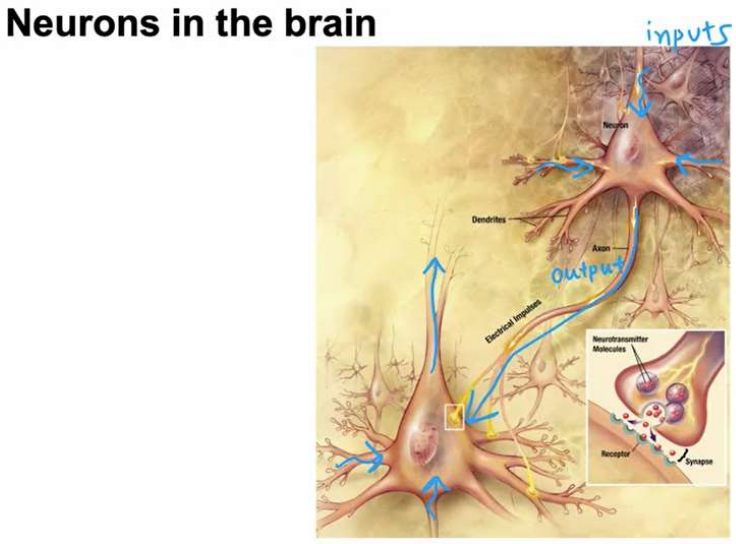
Die Arbeit an neuronalen Netzen begann bereits in den 1950er Jahren und geriet dann für eine Weile in Ungnade. In den 1980er und frühen 1990er Jahren gewannen sie dann wieder an Popularität und zeigten enormen Anklang bei einigen Anwendungen wie der handschriftlichen Ziffernerkennung, die schon damals zum Lesen von Postleitzahlen zum Schreiben von Post und zum Lesen von Dollarzahlen in handschriftlichen Schecks verwendet wurde. Doch Ende der 1990er Jahre geriet es wieder in Ungnade. Ab etwa 2005 erlebte es einen Aufschwung und wurde mit Deep Learning ein wenig umbenannt. Eines der Dinge, die mich damals überrascht haben, war Deep Learning und neuronale Netze bedeuteten sehr ähnliche Dinge.




Aber vielleicht wurde der Begriff „Deep Learning“ damals unterschätzt, weil er einfach viel besser klingt, weil es sich um „Deep Learning“ handelt. Es stellte sich also heraus, dass dies die Marke war, die in den letzten zehn oder anderthalb Jahrzehnten auf dem Vormarsch war. Seitdem haben neuronale Netze einen Anwendungsbereich nach dem anderen revolutioniert. Ich denke, der erste Anwendungsbereich, auf den moderne neuronale Netze oder Deep Learning einen großen Einfluss hatten, war wahrscheinlich die Spracherkennung, wo wir dank modernem Deep Learning und Autoren wie [inaudible] und Geoff Hinton viel bessere Spracherkennungssysteme sahen Dies war

maßgeblich daran beteiligt, und dann begann es Einzug in die Computer Vision zu halten. Manchmal sprechen die Leute immer noch von den ImageNet-Momenten im Jahr 2012, und das war vielleicht ein größeres Aufsehen, als [unhörbar] ihre Fantasie beflügelte und einen großen Einfluss auf Computer Vision hatte. In den darauffolgenden Jahren gelang es uns dann, in die Textwelt oder in die Verarbeitung natürlicher Sprache vorzudringen, und so weiter und so fort.

Mittlerweile werden neuronale Netze in allen Bereichen eingesetzt, vom Klimawandel über medizinische Bildgebung und Online-Werbung bis hin zu Produktempfehlungen, und in wirklich vielen Anwendungsbereichen des maschinellen Lernens kommen mittlerweile neuronale Netze zum Einsatz. Auch wenn die heutigen neuronalen Netze fast nichts damit zu tun haben, wie das Gehirn lernt, gab es schon früh die Motivation, Software zu entwickeln, die das Gehirn nachahmt.

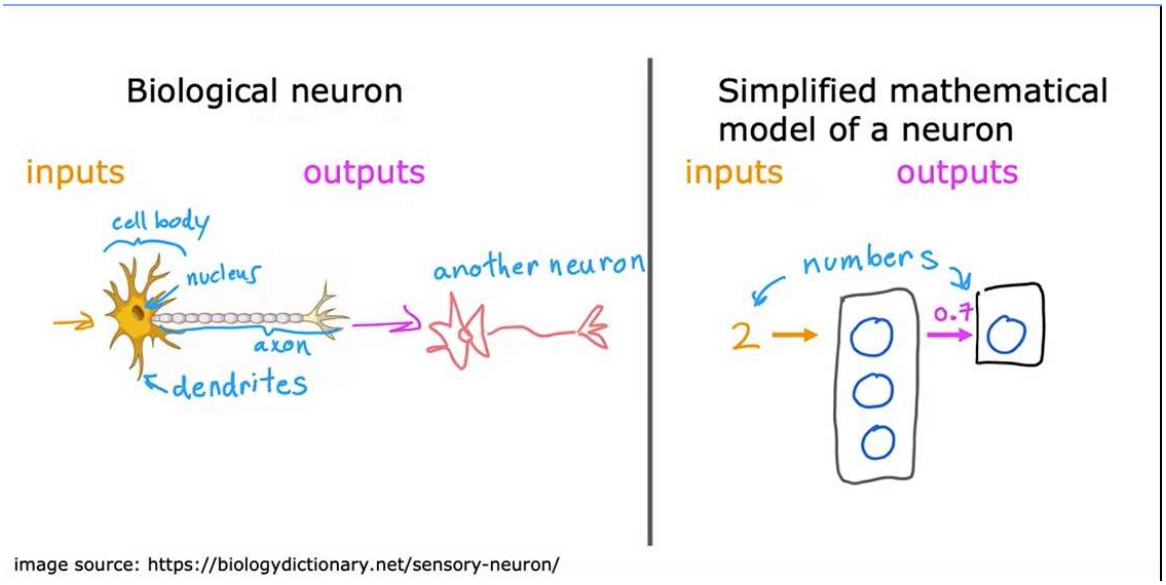



[Credit: US National Institutes of Health, National Institute on Aging]

Wie funktioniert das Gehirn? Hier ist ein Diagramm, das zeigt, wie Neuronen in einem Gehirn aussehen. Das gesamte menschliche Denken entsteht aus solchen Neuronen in Ihrem und meinem Gehirn, die elektrische Impulse senden und manchmal neue Verbindungen anderer Neuronen herstellen. Angenommen, ein Neuron wie dieses verfügt über eine Reihe von Eingängen, über die es elektrische Impulse von anderen Neuronen empfängt, und dann führt dieses Neuron, das ich eingekreist habe, einige Berechnungen durch und sendet diese Ausgänge dann über diese elektrischen Impulse an andere Neuronen, und Der Output dieses oberen Neurons wird wiederum zum Input für dieses Neuron unten, das wiederum Inputs von mehreren anderen Neuronen aggregiert, um dann vielleicht seinen eigenen Output an noch andere Neuronen zu senden, und das ist der Stoff, aus dem menschliches Denken besteht. Hier ist ein vereinfachtes Diagramm eines biologischen Neurons.

Ein Neuron besteht aus einem Zellkörper, wie hier links gezeigt. Wenn Sie einen Biologiekurs besucht haben, erkennen Sie möglicherweise, dass es sich dabei um den Kern des Neurons handelt. Wie wir auf der vorherigen Folie gesehen haben, verfügt das Neuron über unterschiedliche Eingaben. In einem biologischen Neuron werden die Eingangsdrähte Dendriten genannt, und dann sendet es gelegentlich elektrische Impulse über den Ausgangsdraht, der Axon genannt wird, an andere Neuronen. Machen Sie sich über diese biologischen Begriffe keine Sorgen. Wenn Sie sie im Biologieunterricht gesehen haben, erinnern Sie sich vielleicht an sie, aber Sie müssen sich keinen dieser Begriffe wirklich merken, um künstliche neuronale Netze aufzubauen. Aber dieses biologische Neuron kann dann elektrische Impulse senden, die als Input für ein anderes Neuron dienen. Das

künstliche neuronale Netzwerk verwendet also ein sehr vereinfachtes mathematisches Modell dessen, was ein biologisches Neuron tut. Ich werde hier einen kleinen Kreis zeichnen, um ein einzelnes Neuron zu kennzeichnen.



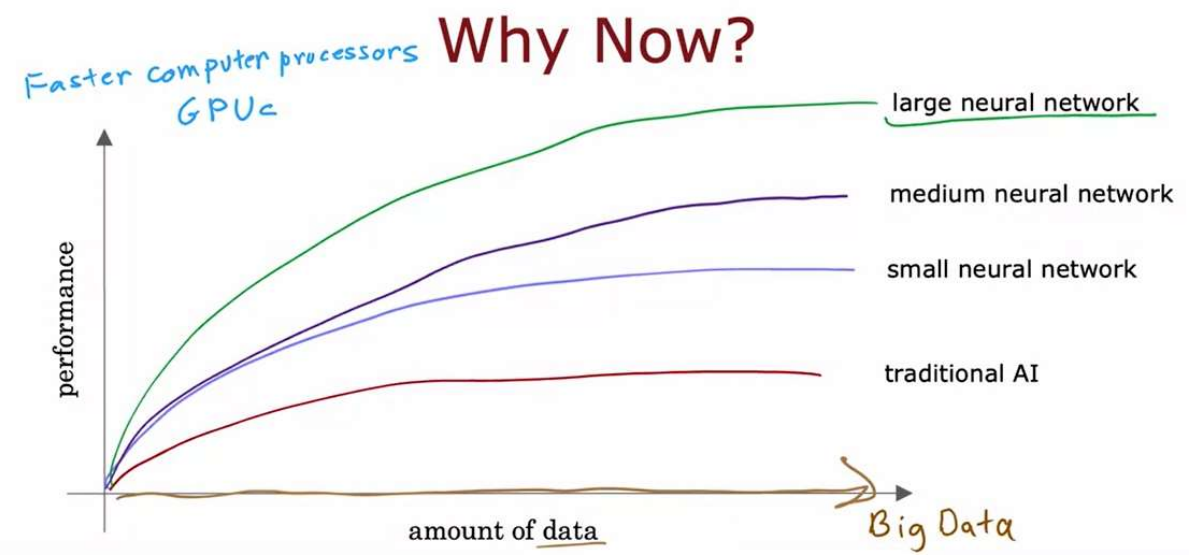

Was ein Neuron tut, ist, dass es einige Eingaben entgegennimmt, eine oder mehrere Eingaben, bei denen es sich nur um Zahlen handelt. Es führt einige Berechnungen durch und gibt eine andere Zahl aus, die dann als Eingabe für ein zweites Neuron dienen könnte, wie hier rechts gezeigt. Wenn Sie ein künstliches neuronales Netzwerk oder einen Deep-Learning-Algorithmus aufbauen, möchten Sie oft nicht nur ein Neuron nach dem anderen aufbauen, sondern oft viele solcher Neuronen gleichzeitig simulieren. In diesem Diagramm zeichne ich drei Neuronen. Was diese Neuronen gemeinsam tun, ist, einige Zahlen einzugeben, einige Berechnungen durchzuführen und einige andere Zahlen auszugeben.

An dieser Stelle möchte ich einen großen Vorbehalt anbringen: Obwohl ich eine lockere Analogie zwischen biologischen Neuronen und künstlichen Neuronen gezogen habe, denke ich, dass wir heute fast keine Ahnung haben, wie das menschliche Gehirn funktioniert. Tatsächlich erzielen Neurowissenschaftler alle paar Jahre einen grundlegenden Durchbruch bei der Funktionsweise des Gehirns. Ich denke, wir werden dies auch in absehbarer Zukunft tun. Das ist für mich ein Zeichen dafür, dass es noch viele Durchbrüche gibt, die darüber entdeckt werden müssen, wie das Gehirn tatsächlich funktioniert, und dass daher Versuche, blind das nachzuahmen, was wir heute über das menschliche Gehirn wissen, was ehrlich gesagt sehr wenig ist, wahrscheinlich nicht gelingen werden uns so weit beim Aufbau roher Intelligenz gebracht.

Sicherlich nicht mit unserem aktuellen Wissensstand in den Neurowissenschaften. Allerdings werden wir selbst mit diesen extrem vereinfachten Modellen eines Neurons, über die wir sprechen werden, in der Lage sein, wirklich leistungsstarke Deep-Learning-Algorithmen zu entwickeln. Wenn Sie sich also tiefer mit neuronalen Netzen und Deep Learning befassen, sollten Sie die biologische Motivation nicht zu ernst nehmen, auch wenn die Ursprünge biologisch motiviert sind. Tatsächlich haben sich diejenigen von uns, die sich mit Deep Learning befassen, von der biologischen Motivation abgewandt. Stattdessen nutzen sie lediglich technische Prinzipien, um herauszufinden, wie sich effektivere Algorithmen entwickeln lassen. Aber ich denke, es könnte immer noch Spaß machen, hin und wieder darüber zu spekulieren und darüber nachzudenken, wie biologische Neuronen funktionieren.

Die Ideen neuronaler Netze gibt es schon seit vielen Jahrzehnten. Ein paar Leute haben mich gefragt: „Hey Andrew, warum jetzt? Warum haben neuronale Netze erst in den letzten paar Jahren so richtig Fahrt aufgenommen?“ Dies ist ein Bild, das ich für sie zeichne, wenn mir diese Frage gestellt wird, und das Sie vielleicht auch für andere zeichnen könnten, wenn sie Ihnen diese Frage stellen. Lassen Sie mich auf der horizontalen Achse die Datenmenge darstellen, die Sie für ein Problem haben, und auf der vertikalen Achse die Leistung oder Genauigkeit eines auf dieses Problem angewendeten Lernalgorithmus. In den letzten Jahrzehnten, mit dem Aufkommen des Internets, der Verbreitung von Mobiltelefonen und der Digitalisierung unserer Gesellschaft, ist die Datenmenge, die wir für viele Anwendungen haben, stetig gestiegen. Bei vielen Aufzeichnungen, in denen P auf Papier verwendet wird, z. B. wenn Sie etwas bestellen und nicht auf einem Blatt Papier, ist es viel wahrscheinlicher, dass es sich um eine digitale Aufzeichnung handelt.

Wenn Sie einen Arzt aufsuchen, ist Ihre Gesundheitsakte jetzt viel wahrscheinlicher digital als auf Papier. So ist in vielen Anwendungsbereichen die Menge an digitalen Daten explodiert. Was wir gesehen haben, war, dass es bei herkömmlichen Algorithmen für maschinelles Lernen, wie der logistischen Regression und der linearen Regression, sehr schwierig war, die Leistung weiter zu steigern, selbst wenn man diesen Algorithmen mehr Daten zuführte. Es war also so, als ob die traditionellen Lernalgorithmen wie die lineare Regression und die logistische Regression einfach nicht in der Lage wären, mit der Datenmenge zu skalieren, die wir ihnen jetzt zur Verfügung stellen konnten, und dass sie nicht in der Lage wären, all diese Daten, die wir hatten, effektiv zu nutzen für unterschiedliche Anwendungen.

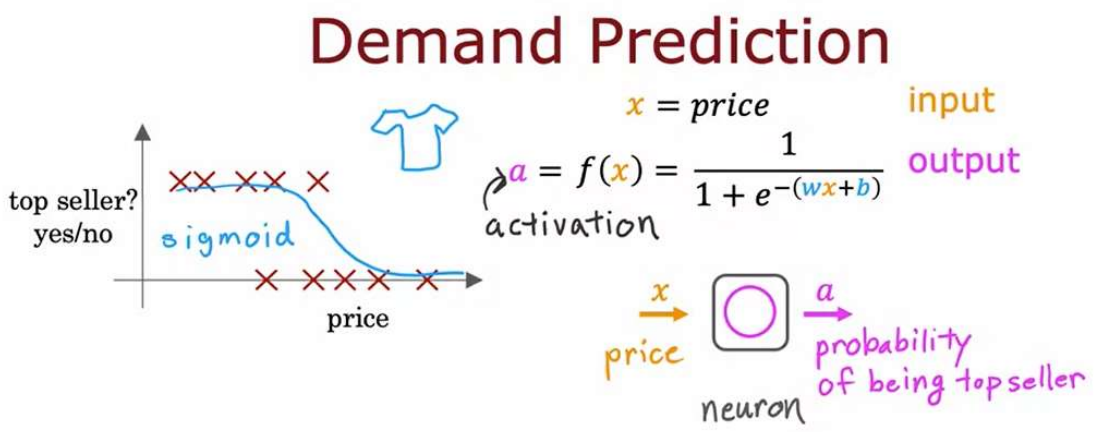



KI-Forscher begannen zu beobachten, dass die Leistung möglicherweise so aussieht, wenn man ein kleines neuronales Netzwerk anhand dieses Datensatzes trainiert. Wenn Sie ein mittelgroßes neuronales Netzwerk trainieren würden, also eines mit mehr Neuronen, könnte seine Leistung so aussehen. Wenn man ein sehr großes neuronales Netzwerk trainiert, also eines mit vielen dieser künstlichen Neuronen, dann wird die Leistung bei manchen Anwendungen einfach immer weiter steigen. Das bedeutete also zwei Dinge: Es bedeutete, dass man für eine bestimmte Klasse von Anwendungen, bei denen man sehr viele Daten hat, manchmal den Begriff „Big Data“ im Umlauf hört, wenn man in der Lage ist, ein sehr großes neuronales Netzwerk zu trainieren, um davon zu profitieren Mit der riesigen Datenmenge, über die Sie verfügen, könnten Sie Leistung in allen Bereichen erzielen, von der Spracherkennung über die Bilderkennung bis hin zu Anwendungen zur Verarbeitung natürlicher Sprache und vielem mehr.

Mit früheren Generationen von Lernalgorithmen war dies einfach nicht möglich. Dies hat dazu geführt, dass sich Deep-Learning-Algorithmen durchgesetzt haben, und dies ist auch der Grund für schnellere Computerprozessoren, einschließlich des Aufstiegs von GPUs oder Grafikprozessoreinheiten. Dabei handelt es sich um Hardware, die ursprünglich für die Generierung ansprechender Computergrafiken entwickelt wurde, sich aber auch für Deep Learning als äußerst leistungsstark erwies. Dies war auch ein wesentlicher Faktor dafür, dass Deep-Learning-Algorithmen zu dem wurden, was sie heute sind. Das ist der Grund, warum neuronale Netze entstanden sind und warum sie sich in den letzten Jahren so schnell durchgesetzt haben. Lassen Sie uns nun tiefer in die Details eintauchen, wie ein neuronales Netzwerk tatsächlich funktioniert. Bitte fahren Sie mit dem nächsten Video fort.

## Nachfragevorhersage

Um die Funktionsweise neuronaler Netze zu veranschaulichen, beginnen wir mit einem Beispiel. Wir verwenden ein Beispiel aus der Nachfragevorhersage, bei dem Sie sich das Produkt ansehen und versuchen vorherzusagen, ob dieses Produkt ein Verkaufsschlager sein wird oder nicht. Lass uns einen Blick darauf werfen. In diesem Beispiel verkaufen Sie T-Shirts und möchten wissen, ob ein bestimmtes T-Shirt ein Verkaufsschlager wird, ja oder nein, und Sie haben Daten zu verschiedenen T-Shirts gesammelt, die zu unterschiedlichen Preisen verkauft wurden. und welche davon zum Verkaufsschlager wurden.



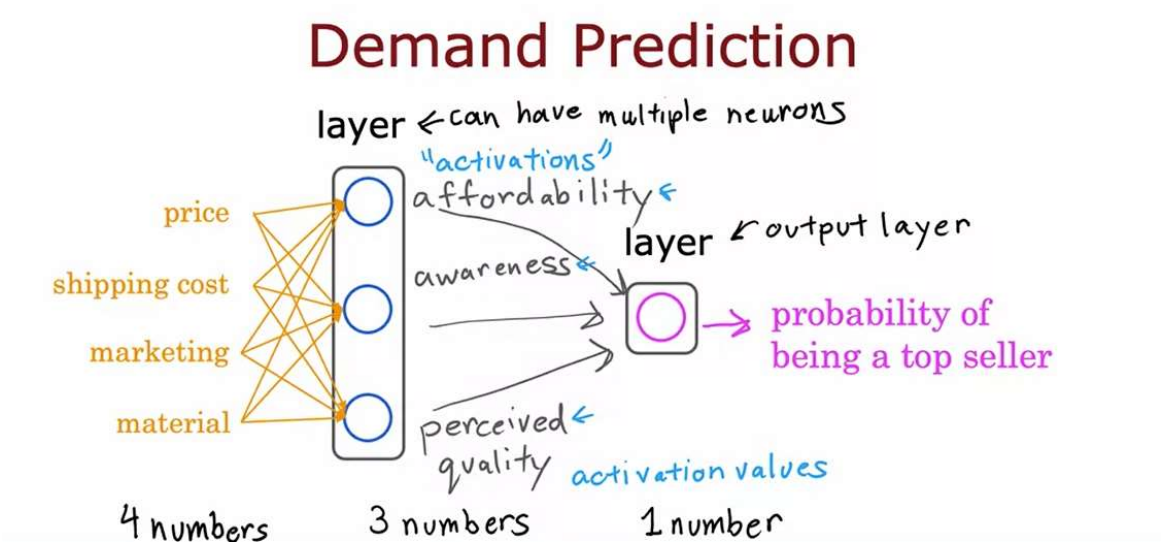

Diese Art von Anwendung wird heute von Einzelhändlern genutzt, um bessere Lagerbestände sowie Marketingkampagnen zu planen. Wenn Sie wissen, was wahrscheinlich ein Verkaufsschlager wird, würden Sie beispielsweise planen, einfach im Voraus mehr von dieser Aktie zu kaufen. In diesem Beispiel ist das Eingabemerkmal x der Preis des T-Shirts und somit die Eingabe für den Lernalgorithmus.

Wenn Sie eine logistische Regression anwenden, um eine Sigmoidfunktion an die Daten anzupassen, die so aussehen könnte, könnten die Ergebnisse Ihrer Vorhersage so aussehen: 1/1 plus e zum negativen wx plus b. Zuvor hatten wir dies als f von x als Ausgabe des Lernalgorithmus geschrieben. Um uns auf den Aufbau eines neuronalen Netzwerks vorzubereiten, werde ich die Terminologie ein wenig ändern und das Alphabet a verwenden, um die Ausgabe dieses logistischen Regressionsalgorithmus zu bezeichnen.

Der Begriff a steht für Aktivierung und ist eigentlich ein Begriff aus der Neurowissenschaft. Er bezieht sich darauf, wie stark ein Neuron eine hohe Leistung an andere ihm nachgeschaltete Neuronen sendet. Es stellt sich heraus, dass diese logistischen Regressionseinheiten oder dieser kleine logistische Regressionsalgorithmus als ein sehr vereinfachtes Modell eines einzelnen Neurons im Gehirn betrachtet werden kann.

Was das Neuron tut, ist, dass wir den Preis x eingeben, dann diese Formel darüber berechnet und die Zahl a ausgibt, die durch diese Formel berechnet wird, und die Wahrscheinlichkeit ausgibt, dass dieses T-Shirt ein Oberteil ist Verkäufer. Man kann sich ein Neuron auch wie einen winzig kleinen Computer vorstellen, dessen einzige Aufgabe darin besteht, eine oder mehrere Zahlen einzugeben, beispielsweise einen Preis, und dann eine Zahl oder vielleicht ein paar andere Zahlen auszugeben, in diesem Fall die Wahrscheinlichkeit dass das T-Shirt ein Topseller ist. Wie ich im vorherigen Video angedeutet habe, ist ein Algorithmus für die logistische Regression viel einfacher als das, was jedes biologische Neuron in Ihrem oder meinem Gehirn tut. Deshalb ist das künstliche neuronale Netzwerk ein so stark vereinfachtes Modell des menschlichen Gehirns. Auch wenn Deep-Learning-Algorithmen in der Praxis, wie Sie wissen, sehr gut funktionieren.

---




Angesichts dieser Beschreibung eines einzelnen Neurons muss zum Aufbau eines neuronalen Netzwerks nur noch ein Bündel dieser Neuronen genommen und miteinander verbunden oder zusammengesetzt werden. Schauen wir uns nun ein komplexeres Beispiel einer Nachfrageprognose an. In diesem Beispiel verfügen wir über vier Funktionen, um vorherzusagen, ob ein T-Shirt ein Verkaufsschlager ist oder nicht. Zu den Merkmalen gehören der Preis des T-Shirts, die Versandkosten, der Umfang der Vermarktung dieses bestimmten T-Shirts sowie die Materialqualität. Handelt es sich um hochwertige, dicke Baumwolle oder vielleicht um ein minderwertiges Material? Nun könnte man vermuten, dass es von einigen Faktoren abhängt, ob ein T-Shirt zum Verkaufsschlager wird oder nicht. Erstens ist eines davon die Erschwinglichkeit dieses T-Shirts. Zweitens: Wie groß ist der Bekanntheitsgrad dieses T-Shirts bei potenziellen Käufern?

Drittens ist die wahrgenommene Qualität voreingenommen oder potenziell voreingenommen und besagt, dass es sich um ein hochwertiges T- Shirt handelt. Ich werde ein künstliches Neuron erschaffen, um die Wahrscheinlichkeit abzuschätzen, dass dieses T-Shirt als äußerst erschwinglich wahrgenommen wird. Die Erschwinglichkeit hängt hauptsächlich vom Preis und den Versandkosten ab, da sich der Gesamtbetrag der Bezahlung aus einem Teil des Preises und der Versandkosten zusammensetzt. Wir werden hier ein kleines Neuron verwenden, eine logistische Regressionseinheit, um Preise und Versandkosten einzugeben und vorherzusagen, ob die Leute denken, dass dies

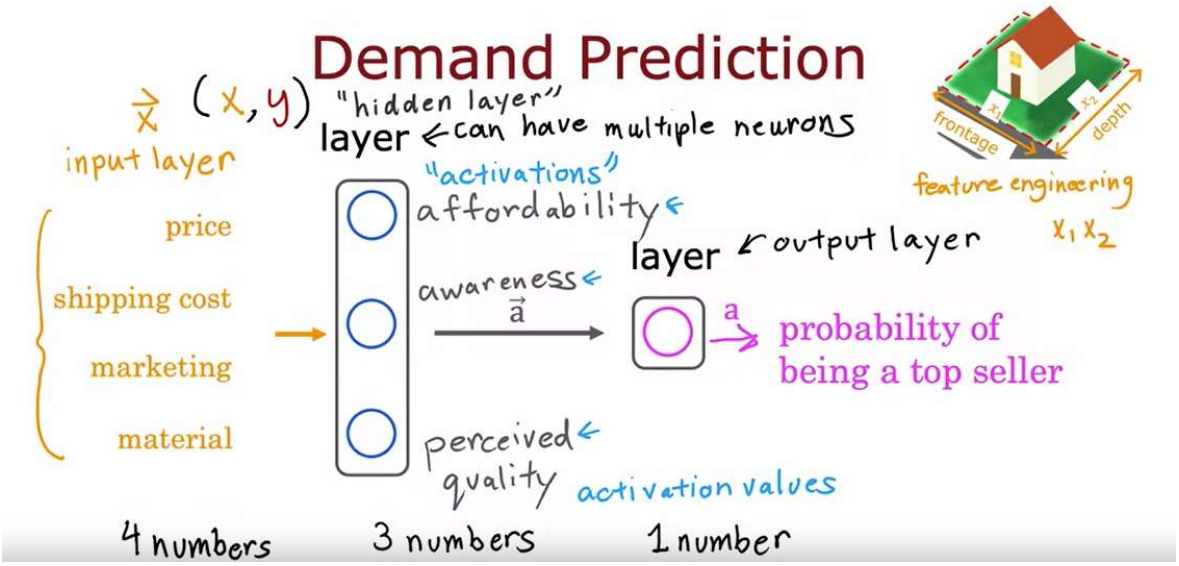
erschwinglich ist? Zweitens werde ich hier ein weiteres künstliches Neuron erstellen, um abzuschätzen, ob das Bewusstsein dafür groß ist. Bekanntheit ist in diesem Fall vor allem eine Funktion der Vermarktung des T-Shirts. Abschließend möchte ich ein weiteres Neuron erstellen, um abzuschätzen, ob die Leute dies als hochwertig empfinden. Dies hängt möglicherweise hauptsächlich vom Preis des T-Shirts und der Materialqualität ab. Der Preis ist hier ein Faktor, denn wenn es ein sehr teures T-Shirt gibt, wird es glücklicherweise oder leider manchmal als qualitativ hochwertig wahrgenommen, weil es sehr teuer ist, als die Leute vielleicht denken, dass es von hoher Qualität sein wird. Basierend auf diesen Schätzungen zu Erschwinglichkeit, Bekanntheit und wahrgenommener Qualität verknüpfen wir dann die Ausgänge dieser drei Neuronen mit einem anderen Neuron hier rechts, sodass es dann eine weitere logistische Regressionseinheit gibt. Damit werden schließlich diese drei Zahlen eingegeben und die Wahrscheinlichkeit ausgegeben, dass dieses T-Shirt ein Top-Seller wird.

In der Terminologie neuronaler Netze werden wir diese drei Neuronen in einer sogenannten Schicht zusammenfassen. Eine Schicht ist eine Gruppierung von Neuronen, die als Eingabe dieselben oder ähnliche Merkmale verwendet und wiederum einige Zahlen zusammen ausgibt. Diese drei Neuronen auf der linken Seite bilden eine Ebene, weshalb ich sie übereinander gezeichnet habe, und dieses einzelne Neuron auf der rechten Seite ist ebenfalls eine Ebene. Die Schicht auf der linken Seite hat drei Neuronen, sodass eine Schicht mehrere Neuronen haben kann, oder sie kann auch ein einzelnes Neuron haben, wie im Fall dieser Schicht auf der rechten Seite. Diese Schicht auf der rechten Seite wird auch als Ausgabeschicht bezeichnet, da die Ausgaben dieses letzten Neurons der vom neuronalen Netzwerk vorhergesagten Ausgabewahrscheinlichkeit entsprechen. In der Terminologie neuronaler Netze werden wir Erschwinglichkeitsbewusstsein und Wahrnehmungsqualität auch als Aktivierungen bezeichnen. Der Begriff Aktivierungen stammt von biologischen Neuronen und bezieht sich auf das Ausmaß, in dem das biologische Neuron einen hohen Ausgangswert sendet oder viele elektrische Impulse an andere Neuronen sendet, die ihm nachgeschaltet sind. Diese Zahlen zu Erschwinglichkeit, Bekanntheit und wahrgenommener Qualität sind die Aktivierungen dieser drei Neuronen in dieser Schicht, und auch diese Ausgabewahrscheinlichkeit ist die Aktivierung dieses hier rechts gezeigten Neurons.

Dieses spezielle neuronale Netzwerk führt daher Berechnungen wie folgt durch. Es gibt vier Zahlen ein, dann verwendet diese Schicht des neuronalen Netzwerks diese vier Zahlen, um die neuen Zahlen, auch Aktivierungswerte genannt, zu berechnen. Dann verwendete die letzte Schicht, die Ausgabeschicht des neuronalen Netzwerks, diese drei Zahlen, um eine Zahl zu berechnen. In einem neuronalen Netzwerk wird diese Liste aus vier Zahlen auch Eingabeschicht genannt, und das ist einfach eine Liste aus vier Zahlen. Nun gibt es eine Vereinfachung, die ich an diesem neuronalen Netzwerk vornehmen möchte. So wie ich es bisher beschrieben habe, mussten wir die Neuronen einzeln durchgehen und entscheiden, welche Eingaben sie von der vorherigen Schicht übernehmen würden. Wir haben zum Beispiel gesagt, dass die Erschwinglichkeit nur eine Funktion des Preises und der Versandkosten ist und die Bekanntheit nur eine Funktion des Marketings usw. ist, aber wenn Sie ein großes neuronales Netzwerk aufbauen, wäre das eine Menge manueller Arbeit Entscheiden Sie, welche Neuronen welche Merkmale als Eingaben verwenden sollen. Die Art und Weise, wie ein neuronales Netzwerk in der Praxis implementiert wird, ist jedes Neuron in einer bestimmten Schicht; Nehmen wir an, diese Ebene in der Mitte hat Zugriff auf jedes Feature, auf jeden Wert der vorherigen Ebene und auf die Eingabeebene, weshalb ich jetzt Pfeile von jedem Eingabemerkmal zu jedem dieser hier in der Mitte gezeigten Neuronen zeichne . Sie können sich vorstellen, dass Sie, wenn Sie versuchen, die Erschwinglichkeit vorherzusagen und wissen, wie hoch die Versandkosten für Marketing und Material sind, vielleicht lernen werden, Marketing und Material zu ignorieren und einfach durch die entsprechende Einstellung der Parameter herauszufinden, dass Sie sich nur auf die Teilmenge konzentrieren der Funktionen, die für die Erschwinglichkeit am relevantesten sind. Um die

Notation und Beschreibung dieses neuronalen Netzwerks weiter zu vereinfachen, werde ich diese vier Eingabemerkmale nehmen und sie als Vektor x schreiben, und wir werden das neuronale Netzwerk so betrachten, dass es vier Merkmale aufweist, die diesen Merkmalsvektor x umfassen.

Dieser Merkmalsvektor wird dieser Schicht in der Mitte zugeführt, die dann drei Aktivierungswerte berechnet. Das heißt, diese Zahlen und diese drei Aktivierungswerte werden wiederum zu einem weiteren Vektor, der dieser letzten Ausgabeschicht zugeführt wird, die schließlich die Wahrscheinlichkeit ausgibt, dass dieses T-Shirt ein Topseller wird. Das ist alles, was ein neuronales Netzwerk ist. Es besteht aus mehreren Schichten, wobei jede Schicht einen Vektor eingibt und einen anderen Zahlenvektor ausgibt. Diese Ebene in der Mitte gibt beispielsweise vier Zahlen x ein und gibt drei Zahlen aus, die der Erschwinglichkeit, dem Bekanntheitsgrad und der wahrgenommenen Qualität entsprechen. Um etwas mehr Terminologie hinzuzufügen: Sie haben gesehen, dass diese Ebene als Ausgabeebene und diese Ebene als Eingabeebene bezeichnet wird. Um der Ebene in der Mitte auch einen Namen zu geben, wird diese Ebene in der Mitte als versteckte Ebene bezeichnet. Ich weiß, dass dies vielleicht nicht der beste oder intuitivste Name ist, aber diese Terminologie kommt daher, wenn man einen Trainingssatz hat.

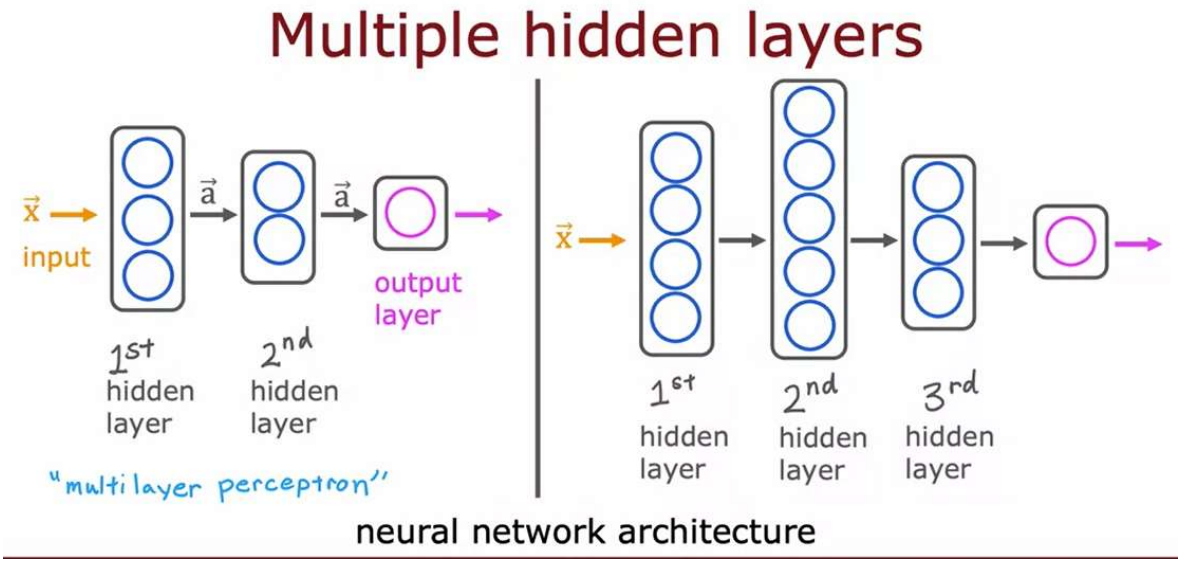



In einem Trainingssatz können Sie sowohl x als auch y beobachten. Ihr Datensatz sagt Ihnen, was x und was y ist, und so erhalten Sie Daten, die Ihnen sagen, was die richtigen Eingaben und die richtigen Ausgaben sind. Ihr Datensatz sagt Ihnen jedoch nicht, welche Werte für Erschwinglichkeit, Bekanntheit und wahrgenommene Qualität korrekt sind. Die korrekten Werte dafür sind ausgeblendet. Sie sind im Trainingssatz nicht zu sehen, weshalb diese Ebene in der Mitte als verborgene Ebene bezeichnet wird. Ich möchte mit Ihnen eine andere Denkweise über neuronale Netze teilen, die ich für nützlich befunden habe, um meine Intuition darüber zu entwickeln. Lassen Sie mich einfach die linke Hälfte dieses Diagramms abdecken und sehen, was übrig bleibt.

Was Sie hier sehen, ist, dass es einen logistischen Regressionsalgorithmus oder eine logistische Regressionseinheit gibt, die als Eingabe Erschwinglichkeit, Bekanntheit und wahrgenommene Qualität eines T-Shirts verwendet und diese drei Merkmale verwendet, um die Wahrscheinlichkeit der Existenz des T-Shirts abzuschätzen ein Topseller. Das ist nur eine logistische Regression. Aber das Coole daran ist, dass statt der Originalfunktionen, des Preises, der Versandkosten, des Marketings usw. möglicherweise bessere Funktionen, Erschwinglichkeit, Bekanntheit und wahrgenommene Qualität verwendet werden, die hoffentlich besser vorhersagen können, ob oder nicht Dieses T-Shirt wird ein Verkaufsschlager. Eine Möglichkeit, sich dieses neuronale Netzwerk vorzustellen, ist einfach

die logistische Regression. Aber als eine Version der logistischen Regression können sie ihre eigenen Funktionen erlernen, die es einfacher machen, genaue Vorhersagen zu treffen. Vielleicht erinnern Sie sich vielleicht noch an das Wohnbeispiel aus dem vorherigen Kurs, in dem wir gesagt haben, dass Sie, wenn Sie den Preis des Hauses vorhersagen möchten, die Fassade oder die Breite der Grundstücke nehmen und diese mit der Tiefe des zu bauenden Grundstücks multiplizieren können ein komplexeres Merkmal, x_1 mal x_2, was der Größe des Rasens entspricht. Dort führten wir manuelles Feature-Engineering durch, bei dem wir uns die Features x_1 und x_2 ansehen und manuell entscheiden mussten, wie wir sie miteinander kombinieren, um bessere Features zu erhalten. Was das neuronale Netzwerk tut, ist, dass Sie die Funktionen nicht manuell entwickeln müssen, sondern es kann, wie Sie später sehen werden, seine eigenen Funktionen lernen, um sich das Lernproblem zu erleichtern. Dies macht neuronale Netze heute zu einem der leistungsstärksten Lernalgorithmen der Welt. Zusammenfassend lässt sich sagen, dass ein neuronales Netzwerk dies tut. Die Eingabeschicht verfügt über einen Merkmalsvektor, in diesem Beispiel vier Zahlen, und wird in die verborgene Schicht eingegeben, die drei Zahlen ausgibt.

Ich werde einen Vektor verwenden, um diesen Aktivierungsvektor zu bezeichnen, den diese verborgene Ebene ausgibt. Dann nimmt die Ausgabeschicht ihre Eingabe in drei Zahlen um und gibt eine Zahl aus, die die endgültige Aktivierung oder die endgültige Vorhersage des neuronalen Netzwerks wäre. Eine Anmerkung: Obwohl ich dieses neuronale Netzwerk zuvor als Berechnung von Erschwinglichkeit, Bekanntheit und wahrgenommener Qualität beschrieben habe, besteht eine der wirklich schönen Eigenschaften eines neuronalen Netzwerks darin, dass man, wenn man es anhand von Daten trainiert, nicht explizit darauf eingehen muss welche anderen Merkmale, wie Erschwinglichkeit usw., das neuronale Netzwerk stattdessen berechnen oder selbst herausfinden sollte, welche Merkmale es in dieser verborgenen Schicht verwenden möchte. Das macht ihn zu einem so leistungsstarken Lernalgorithmus. Sie haben hier ein Beispiel eines neuronalen Netzwerks gesehen und dieses neuronale Netzwerk hat eine einzelne Schicht, die eine verborgene Schicht ist. Schauen wir uns einige andere Beispiele neuronaler Netze an, insbesondere Beispiele mit mehr als einer verborgenen Schicht.




Hier ist ein Beispiel. Dieses neuronale Netzwerk verfügt über einen Eingabemerkmalsvektor X, der einer verborgenen Schicht zugeführt wird. Ich werde dies die erste verborgene Ebene nennen. Wenn diese verborgene Schicht drei Neuronen hat, gibt sie einen Vektor mit drei Aktivierungswerten aus. Diese drei Zahlen können dann in die zweite verborgene Ebene eingegeben werden. Wenn die zweite verborgene Schicht über zwei Neuronen zu logistischen Einheiten verfügt, dann gibt diese zweite

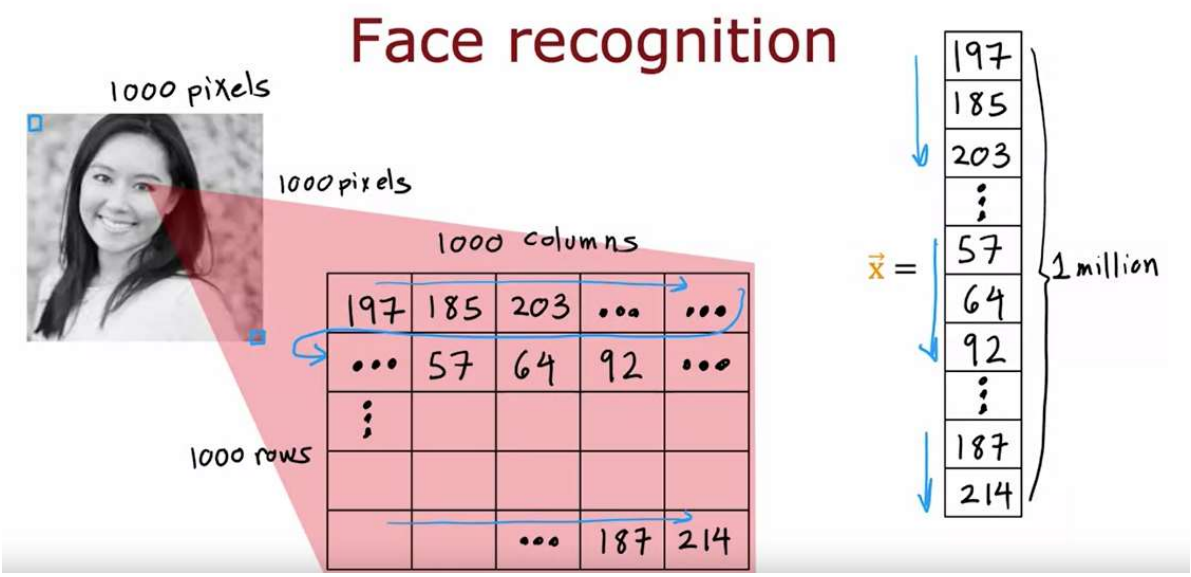
verborgene Schicht einen weiteren Vektor mit nun zwei Aktivierungswerten aus, der möglicherweise an die Ausgabeschicht geht, die dann die endgültige Vorhersage des neuronalen Netzwerks ausgibt.

Hier ist ein weiteres Beispiel. Hier ist ein neuronales Netzwerk, dessen Eingabe an die erste verborgene Schicht geht, die Ausgabe der ersten verborgenen Schicht an die zweite verborgene Schicht, an die dritte verborgene Schicht und schließlich an die Ausgabeschicht. Wenn Sie Ihr eigenes neuronales Netzwerk aufbauen, müssen Sie unter anderem entscheiden, wie viele verborgene Schichten Sie möchten und wie viele Neuronen jede verborgene Schicht haben soll. Diese Frage, wie viele verborgene Schichten und wie viele Neuronen pro verborgener Schicht vorhanden sind, ist eine Frage der Architektur des neuronalen Netzwerks. Später in diesem Kurs erfahren Sie einige Tipps zur Auswahl einer geeigneten Architektur für ein neuronales Netzwerk. Aber auch die Auswahl der richtigen Anzahl verborgener Schichten und der Anzahl verborgener Einheiten pro Schicht kann sich auf die Leistung eines Lernalgorithmus auswirken. Später in diesem Kurs erfahren Sie auch, wie Sie eine gute Architektur für Ihr neuronales Netzwerk auswählen.

Übrigens sieht man in mancher Literatur diese Art von neuronalen Netzwerken mit mehreren Schichten, die als mehrschichtiges Perzeptron bezeichnet werden. Wenn Sie das sehen, bezieht sich das lediglich auf ein neuronales Netzwerk, das so aussieht, wie Sie es hier auf der Folie sehen. Das ist ein neuronales Netzwerk.

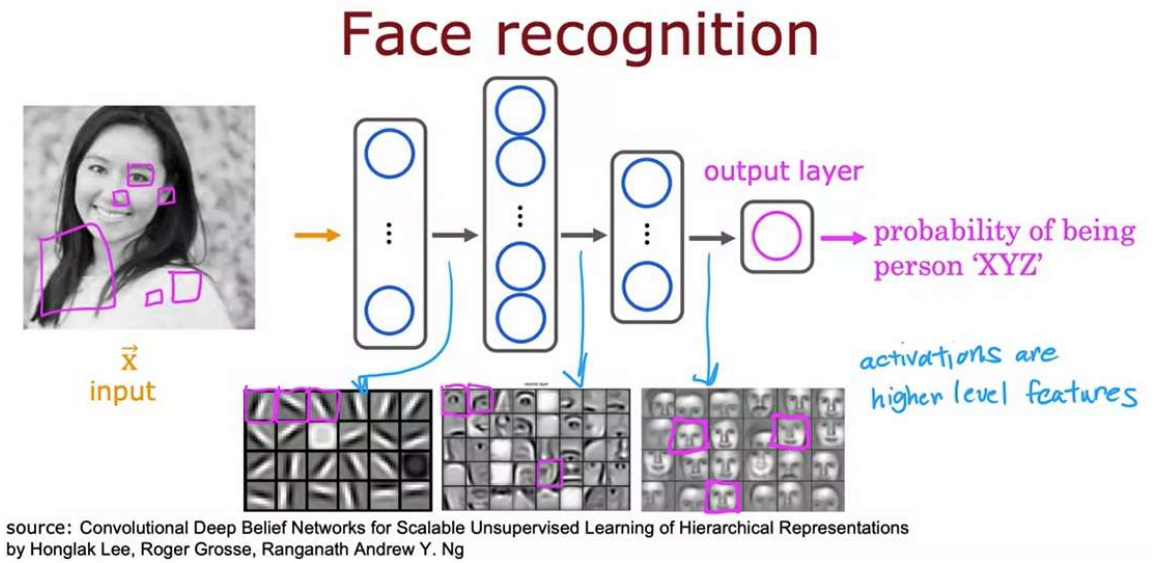
## Example: Recognizing Images

Im letzten Video haben Sie anhand eines Beispiels zur Nachfragevorhersage gesehen, wie ein neuronales Netzwerk funktioniert. Schauen wir uns an, wie Sie eine ähnliche Idee auf Computer-Vision-Anwendungen anwenden können.




Wenn Sie eine Gesichtserkennungsanwendung erstellen, möchten Sie möglicherweise ein neuronales Netzwerk trainieren, das ein Bild wie dieses als Eingabe verwendet und die Identität der Person auf dem Bild ausgibt. Dieses Bild ist 1.000 x 1.000 Pixel groß. Seine Darstellung im Computer erfolgt tatsächlich als 1.000 x 1.000 Raster oder wird auch 1.000 x 1.000 Matrix von Pixelintensitätswerten genannt. In diesem Beispiel reichen meine Pixelintensitätswerte oder Pixelhelligkeitswerte von 0 bis 255, sodass 197 hier die Helligkeit des Pixels ganz oben links im Bild wäre, 185 die Helligkeit des Pixels, ein Pixel darüber , und also weiter bis hinunter zu 214 wäre die untere rechte Ecke dieses Bildes. Wenn Sie diese Pixelintensitätswerte in einen Vektor aufrollen, erhalten Sie am Ende eine Liste oder einen Vektor mit einer Million Pixelintensitätswerten.

Eine Million, weil 1.000 mal 1.000 im Quadrat eine Million Zahlen ergibt. Das Problem der Gesichtserkennung besteht darin, ein neuronales Netzwerk zu trainieren, das als Eingabe einen Merkmalsvektor mit einer Million Pixelhelligkeitswerten verwendet und die Identität der Person auf dem Bild ausgibt. So könnten Sie ein neuronales Netzwerk aufbauen, um diese Aufgabe auszuführen. Das Eingabebild X wird dieser Neuronenschicht zugeführt.



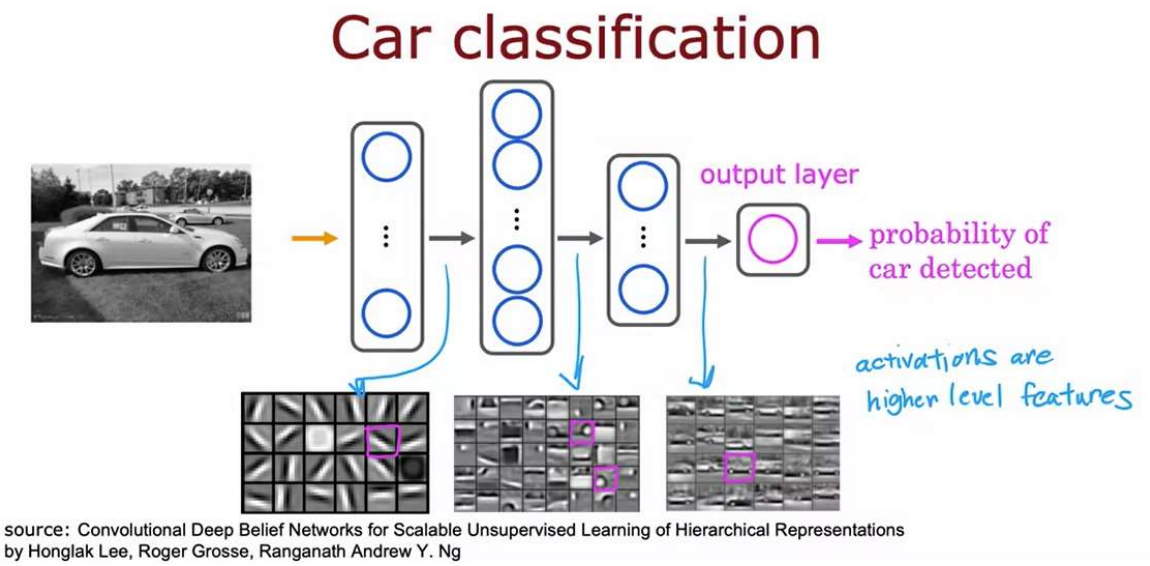

Dies ist die erste verborgene Ebene, die dann einige Features extrahiert. Die Ausgabe dieser ersten verborgenen Schicht wird einer zweiten verborgenen Schicht zugeführt, und diese Ausgabe wird einer dritten Schicht und schließlich der Ausgabeschicht zugeführt, die dann beispielsweise die Wahrscheinlichkeit schätzt, dass es sich dabei um eine bestimmte Person handelt.

Eine interessante Sache wäre, wenn man sich ein neuronales Netzwerk anschaut, das auf vielen Bildern von Gesichtern trainiert wurde, und versucht, diese verborgenen Schichten zu visualisieren und zu berechnen. Es stellt sich heraus, dass Sie möglicherweise Folgendes finden, wenn Sie ein System wie dieses auf vielen Bildern von Gesichtern trainieren und auf die verschiedenen Neuronen in den verborgenen Schichten blicken, um herauszufinden, was sie möglicherweise berechnen.

In der ersten verborgenen Ebene finden Sie möglicherweise ein Neuron, das nach der niedrigen vertikalen Linie oder einer solchen vertikalen Kante sucht. Ein zweites Neuron sucht nach einer solchen orientierten Linie oder orientierten Kante. Das dritte Neuron sucht nach einer Linie in dieser Ausrichtung und so weiter. In den frühesten Schichten eines neuronalen Netzwerks stellen Sie möglicherweise fest, dass die Neuronen nach sehr kurzen Linien oder sehr kurzen Kanten im Bild suchen. Wenn Sie sich die nächste verborgene Ebene ansehen, stellen Sie fest, dass diese Neuronen möglicherweise lernen, viele kleine kurze Linien und kleine kurze Kantensegmente zu gruppieren, um nach Teilen von Gesichtern zu suchen.

Jedes dieser kleinen quadratischen Kästchen ist beispielsweise eine Visualisierung dessen, was dieses Neuron zu erkennen versucht. Dieses erste Neuron sieht aus, als würde es versuchen, das Vorhandensein oder Fehlen eines Auges an einer bestimmten Position im Bild zu erkennen. Das zweite Neuron sieht aus, als würde es versuchen, einen Nasenwinkel zu erkennen, und vielleicht versucht dieses Neuron hier, die Unterseite eines Ohrs zu erkennen. Wenn Sie sich dann die nächste verborgene Ebene in diesem Beispiel ansehen, aggregiert das neuronale Netzwerk verschiedene Teile von Gesichtern, um dann zu versuchen, das Vorhandensein oder Fehlen größerer, gröberer Gesichtsformen zu erkennen. Schließlich wird durch die Erkennung, wie sehr das Gesicht verschiedenen Gesichtsformen entspricht, ein umfangreicher Satz an Merkmalen erstellt, der der Ausgabeebene dann dabei hilft, die Identität des Personenbilds zu bestimmen.

Das Bemerkenswerte am neuronalen Netzwerk ist, dass Sie diese Merkmalsdetektoren auf den verschiedenen verborgenen Schichten ganz von selbst erlernen können. In diesem Beispiel wurde ihm nie gesagt, dass es in der ersten Ebene nach kurzen kleinen Kanten suchen soll, in der zweiten Ebene nach Augen, Nasen und Gesichtsteilen und dann in der dritten Ebene nach vollständigeren Gesichtsformen. Das neuronale Netzwerk ist in der Lage, diese Dinge selbstständig aus Daten herauszufinden. Nur eine Anmerkung: In dieser Visualisierung werden die Neuronen in der ersten verborgenen Schicht gezeigt, wie sie auf relativ kleine Fenster schauen, um nach diesen Kanten zu suchen. In der zweiten verborgenen Ebene wird ein größeres Fenster angezeigt, und in der dritten verborgenen Ebene wird ein noch größeres Fenster angezeigt. Diese kleinen Neuronen-Visualisierungen entsprechen tatsächlich unterschiedlich großen Regionen im Bild. Mal zum Spaß sehen wir uns an, was passiert, wenn Sie dieses neuronale Netzwerk mit einem anderen Datensatz trainieren würden, beispielsweise mit vielen Bildern von Autos, Bild nebenbei.



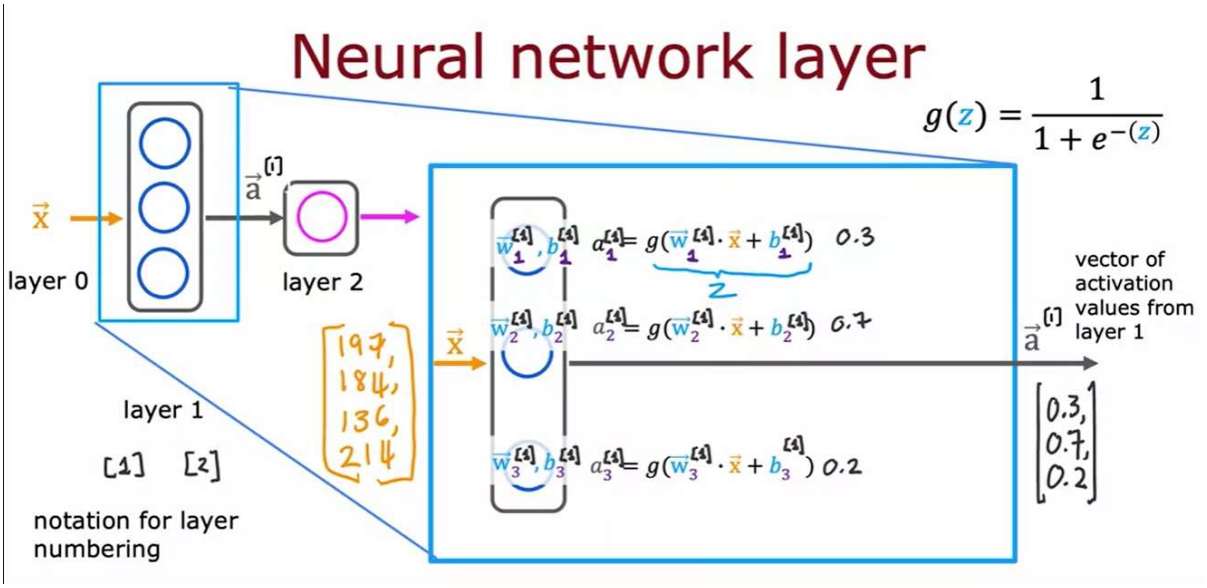
source: Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations by Honglak Lee, Roger Grosse, Ranganath Andrew Y. Ng

Derselbe Lernalgorithmus soll Autos erkennen und dann die Kanten in der ersten Ebene lernen. Ziemlich ähnlich, aber dann lernen sie, Teile von Autos in der zweiten verborgenen Ebene und dann vollständigere Autoformen in der dritten verborgenen Ebene zu erkennen. Allein durch die Eingabe unterschiedlicher Daten lernt das neuronale Netzwerk automatisch, sehr unterschiedliche Merkmale zu erkennen, um Vorhersagen über die Erkennung von Autos oder Personen zu treffen oder festzustellen, ob eine bestimmte Aufgabe trainiert wird. So funktioniert ein neuronales Netzwerk für Computer-Vision-Anwendungen. Tatsächlich werden Sie später in dieser Woche sehen, wie Sie selbst ein neuronales Netzwerk aufbauen und es auf eine handschriftliche Ziffernerkennungsanwendung anwenden können. Bisher haben wir die Intuitionen neuronaler Netze beschrieben, um Ihnen ein Gefühl dafür zu vermitteln, wie sie funktionieren. Im nächsten Video werfen wir einen genaueren Blick auf die konkrete Mathematik und eine konkrete Implementierung von Details, wie Sie tatsächlich eine oder mehrere Schichten eines neuronalen Netzwerks aufbauen und wie Sie daher eines dieser Dinge selbst implementieren können. Kommen wir zum nächsten Video.

## Neuronale Netzwerkschicht

Der Grundbaustein der meisten modernen neuronalen Netze ist eine Schicht aus Neuronen. In diesem Video erfahren Sie, wie Sie eine Neuronenschicht aufbauen. Sobald Sie diese aufgebaut haben, können Sie diese Bausteine zu einem großen neuronalen Netzwerk zusammenfügen. Werfen wir einen Blick darauf, wie eine Schicht von Neuronen funktioniert. Hier ist das Beispiel, das wir aus

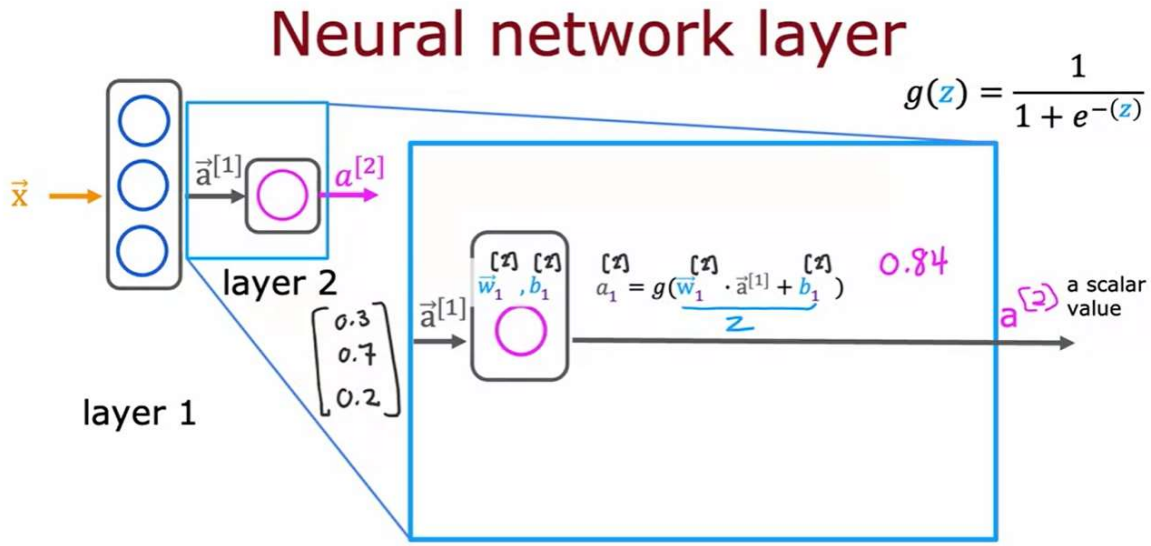
dem Beispiel der Bedarfsvorhersage hatten, bei dem wir vier Eingabemerkmale hatten, die auf diese Schicht mit drei Neuronen in der verborgenen Schicht festgelegt wurden, die dann ihre Ausgabe mit nur einem Neuron an diese Ausgabeschicht sendet. Vergrößern wir die verborgene Ebene, um uns ihre Berechnungen anzusehen. Diese verborgene Schicht gibt vier Zahlen ein und diese vier Zahlen sind Eingaben für jedes der drei Neuronen.




Jedes dieser drei Neuronen implementiert lediglich eine kleine logistische Regressionseinheit oder eine kleine logistische Regressionsfunktion. Nehmen Sie dieses erste Neuron. Es hat zwei Parameter, w und b. Um zu verdeutlichen, dass dies die erste versteckte Einheit ist, werde ich sie als w_1, b_1 tiefstellen. Dadurch gebe ich einen Aktivierungswert a aus, der g von w_1 in einem Produkt mit x plus b_1 ist, wobei dies der bekannte z-Wert ist, den Sie im vorherigen Kurs in der logistischen Regression kennengelernt haben, und g von z ist die bekannte logistische Funktion, 1 über 1 plus e zum negativen z. Vielleicht ist das am Ende eine Zahl von 0,3 und das ist der Aktivierungswert a des ersten Neurons. Um anzuzeigen, dass dies das erste Neuron ist, füge ich hier auch einen Index a_1 hinzu, sodass a_1 eine Zahl wie 0,3 sein kann. Basierend auf den Eingabefunktionen besteht eine Wahrscheinlichkeit von 0,3, dass dies sehr erschwinglich ist. Schauen wir uns nun das zweite Neuron an. Das zweite Neuron hat die Parameter w_2 und b_2, und diese w, b oder w_2, b_2 sind die Parameter der zweiten Logistikeinheit. Es wird berechnet, dass a_2 der logistischen Funktion g entspricht, die auf w_2 Skalarprodukt x plus b_2 angewendet wird, und dies kann eine andere Zahl sein, beispielsweise 0,7. Denn in diesem Beispiel liegt die Wahrscheinlichkeit, dass potenzielle Käufer auf dieses T-Shirt aufmerksam werden, bei 0,7. Ebenso verfügt das dritte Neuron über einen dritten Parametersatz w_3, b_3. In ähnlicher Weise berechnet es einen Aktivierungswert a_3 gleich g des w_3-Skalarprodukts x plus b_3, also beispielsweise 0,2.

In diesem Beispiel geben diese drei Neuronen 0,3, 0,7 und 0,2 aus, und dieser Vektor aus drei Zahlen wird zum Vektor der Aktivierungswerte a, der dann an die letzte Ausgabeschicht dieses neuronalen Netzwerks übergeben wird. Wenn Sie nun neuronale Netze mit mehreren Schichten aufbauen, ist es sinnvoll, den Schichten unterschiedliche Nummern zu geben. Konventionell wird diese Schicht als Schicht 1 des neuronalen Netzwerks und diese Schicht als Schicht 2 des neuronalen Netzwerks bezeichnet. Die Eingabeschicht wird manchmal auch Schicht 0 genannt und heutzutage gibt es neuronale Netze, die Dutzende oder sogar Hunderte von Schichten haben können. Aber um die Notation einzuführen, die uns hilft, zwischen den verschiedenen Ebenen zu unterscheiden, werde ich die hochgestellte eckige Klammer 1 verwenden, um in verschiedene Ebenen zu indizieren. Insbesondere werde ich eine hochgestellte 1 in eckigen Klammern verwenden, das ist eine Notation,

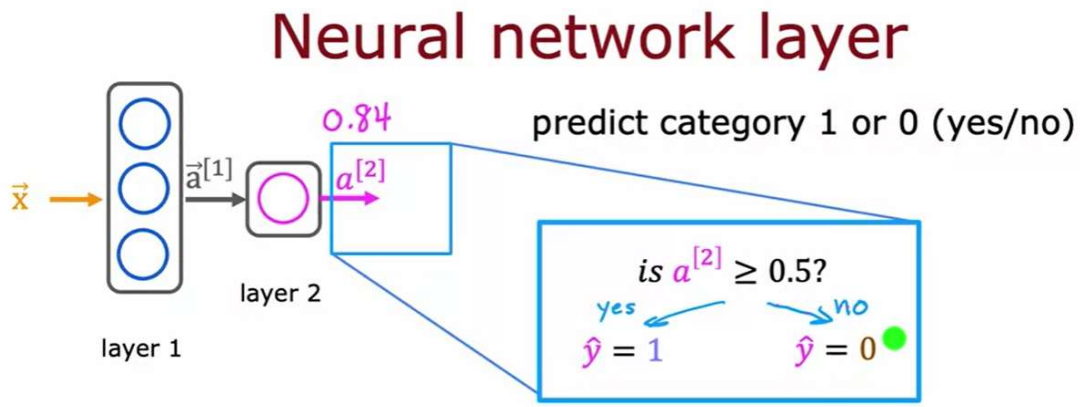
um die Ausgabe von Schicht 1 dieser verborgenen Schicht dieses neuronalen Netzwerks zu bezeichnen, und in ähnlicher Weise sind w_1, b_1 hier die Parameter der ersten Einheit in Schicht 1 des neuronalen Netzwerks, daher füge ich hier auch eine hochgestellte 1 in eckigen Klammern hinzu, und w_2, b_2 sind die Parameter der zweiten verborgenen Einheit bzw. des zweiten verborgenen Neurons in Schicht 1. Seine Parameter werden ebenfalls bezeichnet hier w ^[ 1] so.




Ebenso kann ich hochgestellte eckige Klammern hinzufügen, um anzuzeigen, dass dies die Aktivierungswerte der verborgenen Einheiten der Schicht 1 dieses neuronalen Netzwerks sind. Ich weiß, dass diese Notation vielleicht etwas unübersichtlich wird. Beachten Sie jedoch Folgendes: Wenn Sie diese hochgestellte eckige Klammer 1 sehen, bezieht sich diese lediglich auf eine Größe, die mit Schicht 1 des neuronalen Netzwerks verknüpft ist. Wenn Sie die hochgestellte eckige Klammer 2 sehen, bezieht sich das auf eine Größe, die mit Schicht 2 des neuronalen Netzwerks verknüpft ist, und das Gleiche gilt auch für andere Schichten, einschließlich Schicht 3, Schicht 4 usw. für neuronale Netzwerke mit mehr Schichten. Das ist die Berechnung der Schicht 1 dieses neuronalen Netzwerks. Seine Ausgabe ist dieser Aktivierungsvektor, a ^[ 1], und ich werde ihn hierher kopieren, weil diese Ausgabe a_1 zur Eingabe für Schicht 2 wird. Schauen wir uns nun die Berechnung von Schicht 2 dieses neuronalen Netzwerks genauer an, was ebenfalls der Fall ist die Ausgabeschicht. Die Eingabe für Schicht 2 ist die Ausgabe von Schicht 1, also ist a_1 dieser Vektor 0,3, 0,7, 0,2, den wir gerade im vorherigen Teil dieser Folie berechnet haben.

Da die Ausgabeschicht nur ein einziges Neuron hat, berechnet sie lediglich a_1, das die Ausgabe dieses ersten und einzigen Neurons ist, als g, die Sigmoidfunktion, die in einem Produkt mit a^[1] auf w _1 angewendet wird, also dies ist die Eingabe in diese Ebene und dann plus b_1. Hier handelt es sich um die Größe z, die Sie kennen, und g ist wie zuvor die Sigmoidfunktion, die Sie darauf anwenden. Wenn dies eine Zahl ergibt, beispielsweise 0,84, wird diese zur Ausgabeschicht des neuronalen Netzwerks. Da die Ausgabeschicht in diesem Beispiel nur über ein einzelnes Neuron verfügt, ist diese Ausgabe nur ein Skalar, eine einzelne Zahl und kein Zahlenvektor. Wir bleiben bei unserer Notationskonvention von zuvor und verwenden eine hochgestellte Zahl in eckigen Klammern 2, um die mit Schicht 2 dieses neuronalen Netzwerks verbundenen Größen zu kennzeichnen, sodass a^[2] die Ausgabe dieser Schicht ist und somit I Ich werde dies hier auch als endgültige Ausgabe des neuronalen Netzwerks kopieren. Um die Notation konsistent zu machen, können Sie diesen hochgestellten eckigen Klammern auch Zweien hinzufügen, um anzuzeigen, dass es sich um die Parameter und Aktivierungswerte handelt, die der Schicht 2 des neuronalen Netzwerks zugeordnet sind. Sobald das neuronale Netzwerk a_2 berechnet hat, gibt es einen letzten optionalen Schritt, den

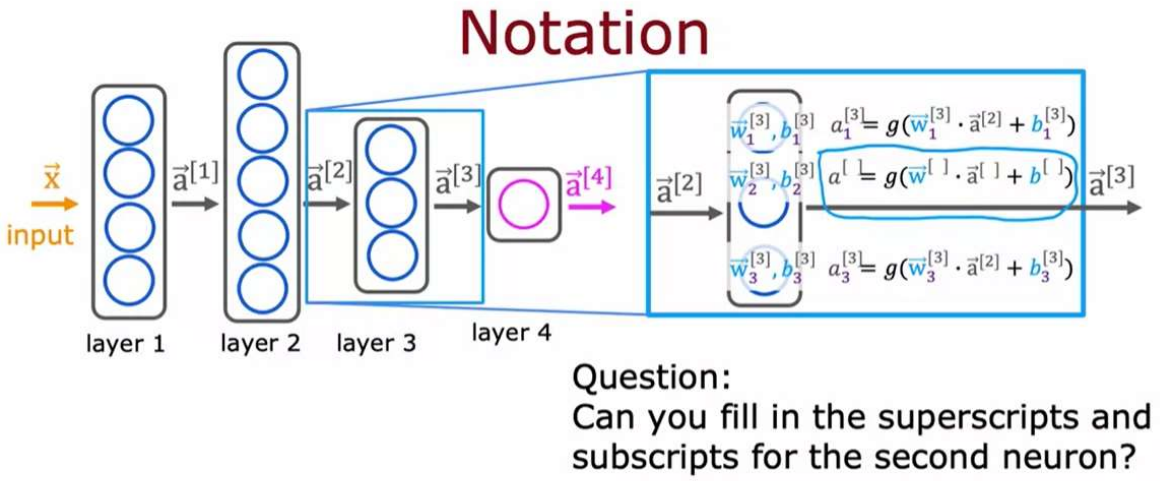
Sie implementieren können oder nicht: Wenn Sie eine binäre Vorhersage wünschen, 1 oder 0, ist das ein Verkaufsschlager? Ja oder nein? Da Sie die Zahl a hochgestellte eckige Klammern 2 tiefgestellte 1 nehmen können, ist dies die Zahl 0,84, die wir berechnet haben, und diesen auf 0,5 beschränken. Wenn er größer als 0,5 ist, können Sie y hat gleich 1 vorhersagen, und wenn er kleiner als 0,5 ist, dann vorhersagen, dass y hat gleich 0 ist.




Diese Schwellenwertbildung haben wir auch gesehen, als Sie im ersten Kurs der Spezialisierung etwas über logistische Regression gelernt haben. Wenn Sie möchten, erhalten Sie dann die endgültige Vorhersage, die entweder Eins oder Null ist, wenn Sie nicht nur die Wahrscheinlichkeit, dass es sich um einen Top-Seller handelt, wollen. So funktioniert also ein neuronales Netzwerk. Jede Schicht gibt einen Zahlenvektor ein und wendet eine Reihe logistischer Regressionseinheiten darauf an und berechnet dann einen weiteren Zahlenvektor, der dann von Schicht zu Schicht weitergegeben wird, bis Sie zur endgültigen Berechnung der Ausgabeschicht gelangen, bei der es sich um die Vorhersage des Neuronen handelt Netzwerk. Dann können Sie entweder einen Schwellenwert von 0,5 festlegen oder nicht, um die endgültige Vorhersage zu treffen. Lassen Sie uns nun auf dieser Grundlage, die wir jetzt aufgebaut haben, einige noch komplexere, noch größere Modelle neuronaler Netzwerke betrachten. Ich hoffe, dass das Konzept der Schichten und wie man sie zum Aufbau eines neuronalen Netzwerks zusammenfügt, durch die Betrachtung weiterer Beispiele noch klarer wird. Kommen wir also zum nächsten Video.

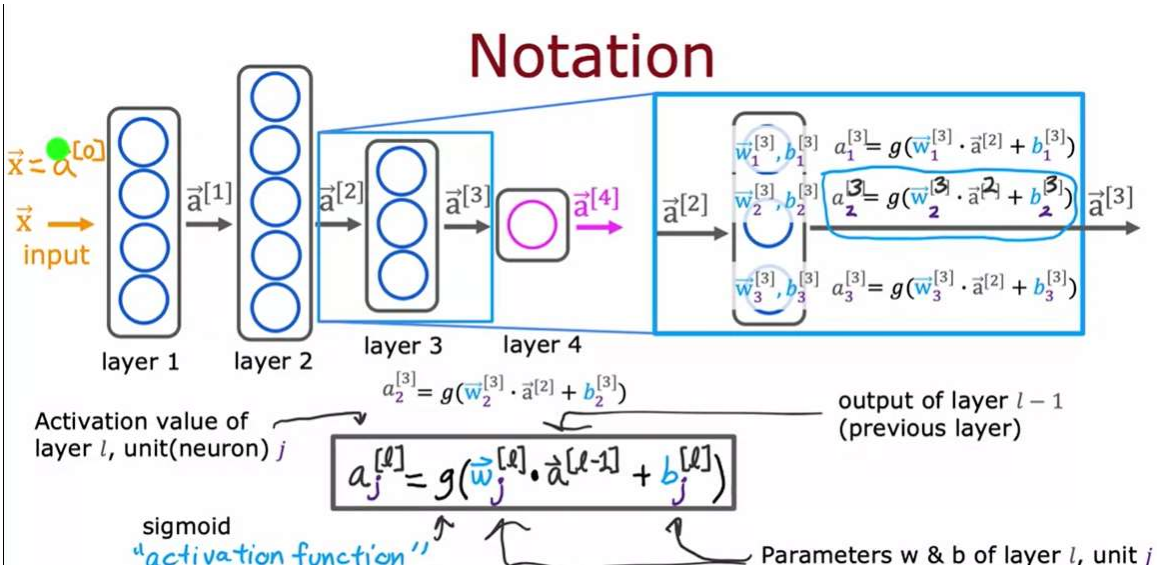
## Komplexere neuronale Netze

Im letzten Video haben Sie etwas über die neuronale Netzwerkschicht erfahren und erfahren, wie diese einen Zahlenvektor eingibt und wiederum einen anderen Zahlenvektor ausgibt. In diesem Video verwenden wir diese Ebene, um ein komplexeres neuronales Netzwerk aufzubauen. Dadurch hoffe ich, dass auch die Notation, die wir für neuronale Netze verwenden, klarer und konkreter wird. Lass uns einen Blick darauf werfen. Dies ist das laufende Beispiel, das ich in diesem Video als Beispiel für ein komplexeres neuronales Netzwerk verwenden werde. Dieses Netzwerk hat vier Schichten, die Eingabeschicht nicht mitgerechnet, die auch Schicht 0 genannt wird, wobei die Schichten 1, 2 und 3 verborgene Schichten sind und Schicht 4 die Ausgabeschicht ist und Schicht 0 wie üblich die Eingabeschicht ist . Wenn wir sagen, dass ein neuronales Netzwerk vier Schichten hat, schließt das konventionell alle verborgenen Schichten in der Ausgabeschicht ein, die Eingabeschicht wird jedoch nicht mitgezählt. Hierbei handelt es sich um ein neuronales Netzwerk mit vier Schichten in der herkömmlichen Art und Weise, Schichten im Netzwerk zu zählen.




Vergrößern wir die Ebene 3, die dritte und letzte verborgene Ebene, um uns die Berechnungen dieser Ebene anzusehen. Schicht 3 gibt einen Vektor ein, eine hochgestellte eckige Klammer 2, die von der vorherigen Schicht berechnet wurde, und gibt a_3 aus, was ein weiterer Vektor ist. Welche Berechnung führt Schicht 3 durch, um von a_2 zu a_3 zu gelangen? Wenn es drei Neuronen hat oder wir es drei versteckte Einheiten nennen, dann hat es die Parameter w_1, b_1, w_2, b_2 und w_3, b_3 und berechnet, dass a_1 gleich dem Sigmoid von w_1 ist. Produkt mit dieser Eingabe in die Ebene plus b_1, und es berechnet a_2 gleich Sigmoid von w_2. Produkt mit wieder a_2, der Eingabe in die Ebene plus b_2 usw., um a_3 zu erhalten. Dann ist die Ausgabe dieser Ebene ein Vektor, der a_1, a_2 und a_3 umfasst. Auch hier gilt: Wenn wir expliziter angeben wollen, dass es sich bei all diesen Größen um Größen handelt, die der Schicht 3 zugeordnet sind, dann fügen wir hier allen hochgestellten eckigen Klammern 3 hinzu, um anzugeben, dass diese Parameter w und b die Parameter sind, die Neuronen zugeordnet sind in Schicht 3 und dass diese Aktivierungen Aktivierungen mit Schicht 3 sind. Beachten Sie, dass dieser Begriff hier w_1 hochgestellte eckige Klammer 3 ist, was die mit Schicht 3 verknüpften Parameter bedeutet. Produkt mit hochgestellter eckiger Klammer 2, das die Ausgabe von Schicht 2 war, die wurde zur Eingabe für Schicht 3. Deshalb gibt es hier a_3, weil es ein Parameter-Assoziator von Schicht 3 ist. Produkt mit, und es gibt dort a_2, weil es die Ausgabe von Schicht 2 ist. Lassen Sie uns nun noch einmal kurz überprüfen, ob wir es verstanden haben Das. Ich werde die hochgestellten und tiefgestellten Zeichen ausblenden, die mit dem zweiten Neuron verbunden sind, und ohne dieses Video zurückzuspulen, spulen Sie zurück, wenn Sie möchten, aber lieber nicht. Aber können Sie, ohne dieses Video zurückzuspulen, die fehlenden hoch- und tiefgestellten Indizes in dieser Gleichung durchdenken und sie selbst ergänzen? Werfen Sie einen Blick auf das abschließende Videoquiz und sehen Sie, ob Sie herausfinden können, welche hochgestellten und tiefgestellten Zeichen für diese Gleichung hier angemessen sind. Wenn Sie sich für die 1. Option entschieden haben, dann haben Sie es richtig gemacht! Die Aktivierung des 2. Neurons auf Schicht 3 wird mit „a“ drei zwei bezeichnet. Um die Aktivierungsfunktion g anzuwenden, verwenden wir die Parameter desselben Neurons. Also haben w und b den gleichen tiefgestellten Index 2 und die hochgestellte eckige Klammer 3. Die Eingabemerkmale sind der Ausgabevektor der vorherigen Ebene, also Ebene 2. Das ist also der Vektor „a“, hochgestellte 2. Die zweite Option ist Verwenden des Vektors 'a' 3, der nicht der Ausgabevektor der vorherigen Ebene ist. Die Eingabe für diese Ebene ist „a“ zwei. Und die dritte Option hat zwei zwei als Eingabe, was eine einzelne Zahl und nicht der Vektor ist. Denn denken Sie daran, dass die richtige Eingabe ein Vektor ist, eine Zwei, mit dem kleinen Pfeil oben, und keine einzelne Zahl. Um es noch einmal zusammenzufassen: a_3 ist die Aktivierung, die der Schicht 3 für das zweite Neuron zugeordnet ist,

daher ist dieses a_2 ein Parameter, der der dritten Schicht zugeordnet ist. Für das zweite Neuron ist dies a_2, wie oben und dann noch zusätzlich b_3. Hoffentlich macht das Sinn. Nur die allgemeinere Form dieser Gleichung für eine beliebige Schicht 0 und für eine beliebige Einheit j, die besagt, dass eine Deaktivierung von Schicht l, Einheit j, wie a32, die auf diesen Term angewendete Sigmoidfunktion ist, d. h der Wellenvektor der Schicht l, z. B. Schicht 3 für die j-te Einheit, also gibt es im obigen Beispiel wieder a_2, und das wird mit einem Deaktivierungswert punktproduktiert. Beachten Sie, dass dies nicht l ist, sondern l minus 1, wie oben a_2, da Sie mit der Ausgabe der vorherigen Ebene punktproduktiv arbeiten und dann b, den Parameter für diese Ebene für diese Einheit j, addieren. Dadurch erhalten Sie die Aktivierung der Schicht l Einheit j, wobei der hochgestellte Index in eckigen Klammern l die Schicht l bezeichnet und ein tiefgestellter j die Einheit j bezeichnet. Beim Aufbau neuronaler Netze bezieht sich Einheit j auf das j-te Neuron, daher verwenden wir diese Begriffe ein wenig austauschbar, wobei jede Einheit ein einzelnes Neuron in der Schicht ist. G ist hier die Sigmoidfunktion. Im Kontext eines neuronalen Netzes hat g einen anderen Namen, der auch Aktivierungsfunktion genannt wird, da g diesen Aktivierungswert ausgibt. Wenn ich Aktivierungsfunktion sage, meine ich diese Funktion g hier.

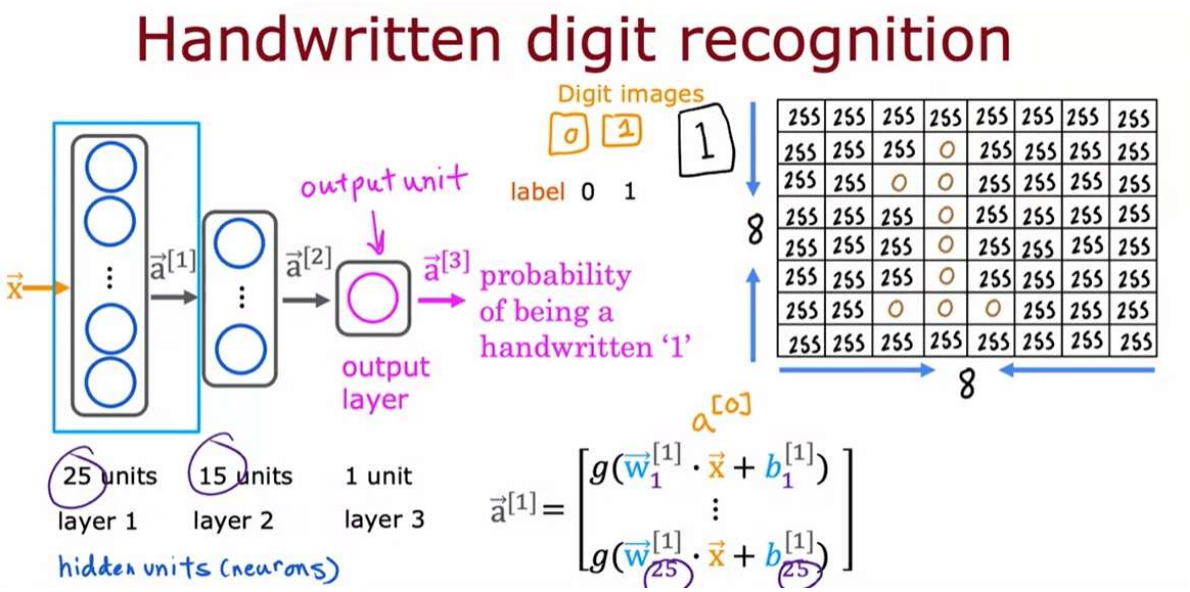



Bisher ist die einzige Aktivierungsfunktion, die Sie gesehen haben, eine Sigmoidfunktion, aber nächste Woche werden wir uns die anderen Funktionen ansehen, dann kann die Sigmoidfunktion auch anstelle von g eingefügt werden. Die Aktivierungsfunktion ist einfach die Funktion, die diese Aktivierungswerte ausgibt. Nur noch ein letztes Stück Notation. Um diese ganze Notation konsistent zu machen, werde ich auch dem Eingabevektor Aktivierungen der ersten Ebene, also a_1, wären das Sigmoid mal dem Punktprodukt der Gewichte mit a_0, was genau dieser Eingabe-Feature-Vektor X ist. Mit dieser Notation wissen Sie jetzt, wie Sie die Aktivierungswerte jeder Ebene in a berechnen neuronales Netzwerk als Funktion der Parameter sowie der Aktivierungen der vorherigen Schicht. Sie wissen jetzt, wie Sie die Aktivierungen einer beliebigen Ebene unter Berücksichtigung der Aktivierungen der vorherigen Ebene berechnen. Lassen Sie uns dies in einen Inferenzalgorithmus für ein neuronales Netzwerk integrieren. Mit anderen Worten, wie man ein neuronales Netzwerk dazu bringt, Vorhersagen zu treffen. Schauen wir uns das im nächsten Video an.

## Schlussfolgerung: Vorhersagen treffen (Vorwärtsausbreitung)

Nehmen wir das Gelernte und fügen wir es in einen Algorithmus ein, damit Ihr neuronales Netzwerk Schlussfolgerungen ziehen oder Vorhersagen treffen kann. Dies wird ein Algorithmus sein, der Vorwärtsausbreitung genannt wird. Lass uns einen Blick darauf werfen. Als motivierendes Beispiel

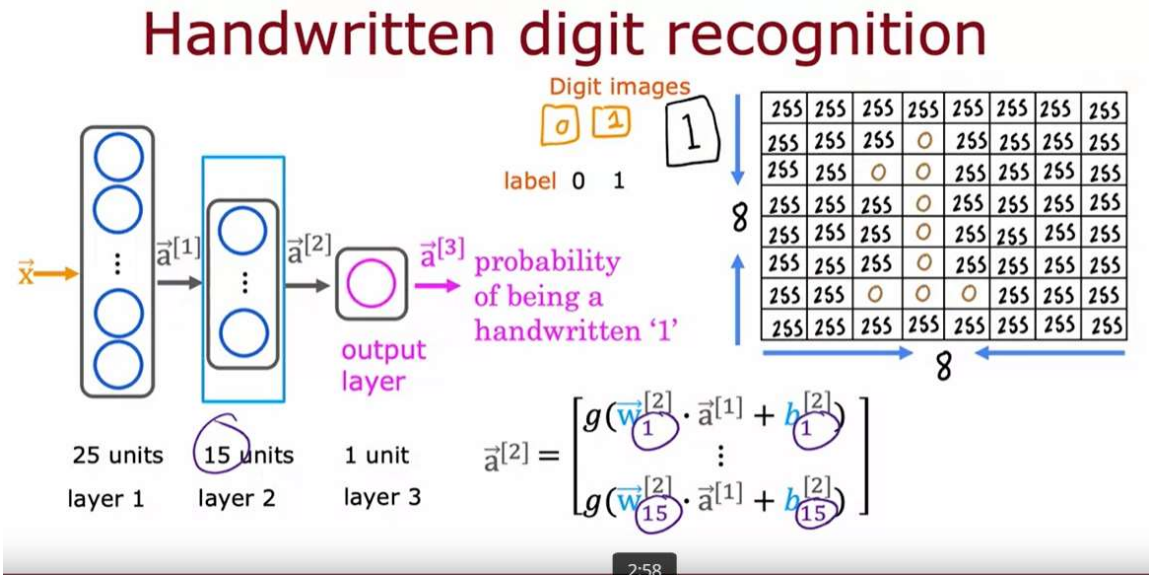
verwende ich die handschriftliche Ziffernerkennung. Und der Einfachheit halber unterscheiden wir nur zwischen den handgeschriebenen Ziffern Null und Eins.



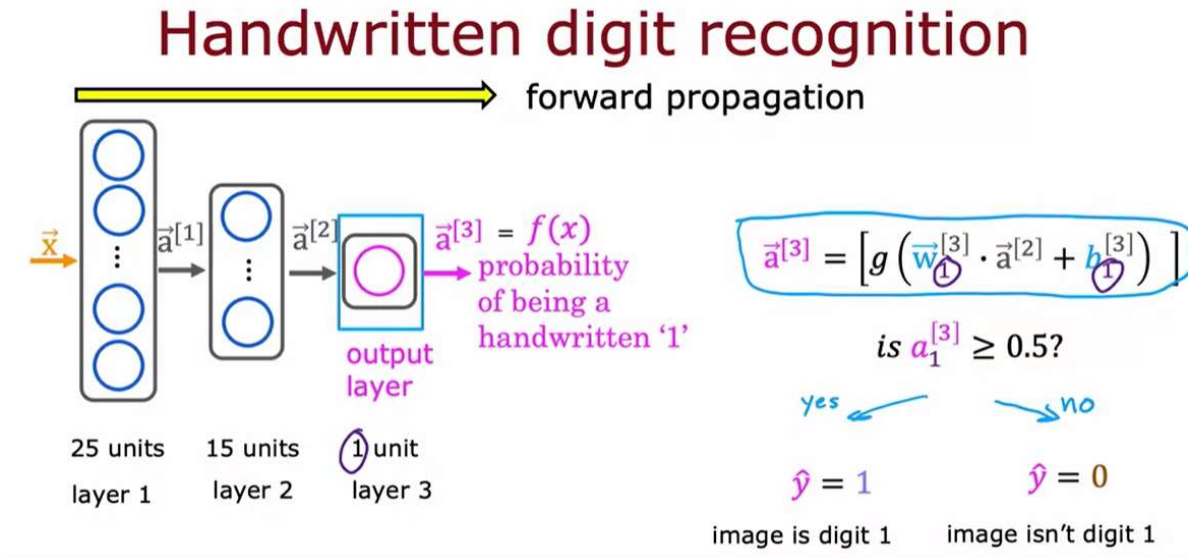

Es handelt sich also nur um ein binäres Klassifizierungsproblem, bei dem wir ein Bild eingeben und klassifizieren. Ist dies die Ziffer Null oder die Ziffer Eins? Und später in dieser Woche können Sie im Übungslabor auch selbst damit spielen. Für das Beispiel der Folie verwende ich ein Bild im Format 8 x 8. Und so ist dieses Bild einer Eins dieses Gitter oder diese Matrix aus acht mal acht oder 64 Pixel-Intensitätswerten, wobei 255 ein helles weißes Pixel und Null ein schwarzes Pixel bezeichnen würde.

Und unterschiedliche Zahlen sind unterschiedliche Grautöne zwischen den Schwarz- und Weißtönen. Angesichts dieser 64 Eingabemerkmale werden wir das neuronale Netzwerk mit zwei verborgenen Schichten verwenden. Wobei die erste verborgene Schicht 25 Neuronen oder 25 Einheiten hat. Die zweite verborgene Schicht besteht aus 15 Neuronen oder 15 Einheiten. Und schließlich die Ausgabeebene oder Ausgabeeinheit. Wie groß ist die Wahrscheinlichkeit, dass diese 1 gegenüber 0 ist?

Lassen Sie uns also die Reihenfolge der Berechnungen durchgehen, die Ihr neuronales Netzwerk durchführen muss, um von der Eingabe X, diesen acht mal acht oder 64 Zahlen, zur vorhergesagten Wahrscheinlichkeit a3 zu gelangen. Die erste Berechnung besteht darin, von X nach a1 zu gelangen, und genau das macht die erste Schicht der ersten verborgenen Schicht. Es führt eine Berechnung eines Superstreifens mit eckiger Klammer 1 durch, der dieser Formel auf der rechten Seite entspricht. Beachten Sie, dass eine Eins 25 Zahlen hat, weil diese verborgene Schicht 25 Einheiten hat. Aus diesem Grund reichen die Parameter von w1 bis w25 sowie von b1 bis b25. Und ich habe hier x geschrieben, aber ich hätte hier auch a0 schreiben können, weil vereinbarungsgemäß die Aktivierung der Ebene Null, also a0, gleich dem Eingabemerkmalswert x ist. Berechnen wir also einfach a1.




Der nächste Schritt besteht darin, a2 zu berechnen. Betrachtet man die zweite verborgene Ebene, führt sie dann diese Berechnung durch, wobei a2 eine Funktion von a1 ist und als sichere Punktaktivierungsfunktion berechnet wird, die auf w Skalarprodukt a1 plus dem entsprechenden Wert von b angewendet wird. Beachten Sie, dass Schicht zwei 15 Neuronen oder 15 Einheiten hat, weshalb die Parameter hier von w1 bis w15 und von b1 bis b15 reichen. Jetzt haben wir a2 berechnet. Der letzte Schritt besteht dann darin, a3 zu berechnen, und wir verwenden hierfür eine sehr ähnliche Berechnung. Nur hat diese dritte Schicht, die Ausgabeschicht, jetzt nur noch eine Einheit, weshalb es hier nur eine Ausgabe gibt. Also ist a3 nur ein Skalar. Und schließlich können Sie optional den Index a3 nehmen und ihn auf 4,5 beschränken, um eine binäre Klassifizierungsbezeichnung zu erhalten. Ist das die Ziffer 1? Ja oder nein? Die Berechnungsfolge nimmt also zuerst x und berechnet dann a1, dann a2 und dann a3, was auch die Ausgabe der neuronalen Netze ist. Sie können das auch als f(x) schreiben. Denken Sie also daran, als wir etwas über lineare Regression und logistische Regression lernten, verwendeten wir f(x), um die Ausgabe der linearen Regression oder der logistischen Regression zu bezeichnen. Wir können also auch f(x) verwenden, um die vom neuronalen Netzwerk berechnete Funktion als Funktion von x zu bezeichnen. Da diese Berechnung von links nach rechts erfolgt, beginnen Sie bei x und berechnen a1, dann a2 und dann a3.



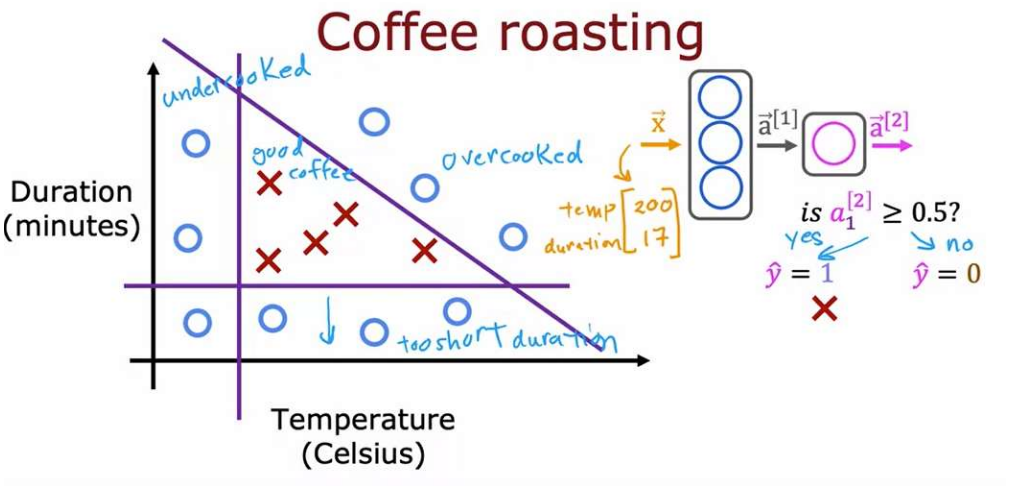
Dieses Album wird auch Vorwärtsausbreitung genannt, weil Sie die Aktivierungen der Neuronen verbreiten. Sie führen diese Berechnungen also vorwärts von links nach rechts durch. Und dies steht im Gegensatz zu einem anderen Algorithmus namens Rückwärtsausbreitung oder Backpropagation, der zum Lernen verwendet wird. Und das erfahren Sie nächste Woche. Und übrigens, diese Art von neuronaler Netzwerkarchitektur, bei der es zunächst mehr verborgene Einheiten gibt und die Anzahl der verborgenen Einheiten dann abnimmt, je näher man der Ausgabeschicht kommt. Bei der Auswahl neuronaler Netzwerkarchitekturen gibt es auch eine ziemlich typische Auswahl. Und noch mehr Beispiele dafür sehen Sie im Praxislabor.

Das ist also die Inferenz eines neuronalen Netzwerks unter Verwendung des Vorwärtsausbreitungs-algorithmus. Und damit könnten Sie die Parameter eines neuronalen Netzwerks herunterladen, das jemand anderes trainiert und im Internet veröffentlicht hat. Und Sie könnten mithilfe ihres neuronalen Netzwerks Rückschlüsse auf Ihre neuen Daten ziehen. Nachdem Sie nun die Mathematik und den Algorithmus kennengelernt haben, werfen wir einen Blick darauf, wie Sie dies tatsächlich in Tensorflow implementieren können. Schauen wir uns das konkret im nächsten Video an.

## Schlussfolgerung im Code

TensorFlow ist eines der führenden Frameworks zur Implementierung von Deep-Learning-Algorithmen. Wenn ich Projekte erstelle, ist TensorFlow tatsächlich das Tool, das ich am häufigsten verwende. Das andere beliebte Tool ist PyTorch. Aber wir werden uns in dieser Spezialisierung auf TensorFlow konzentrieren.

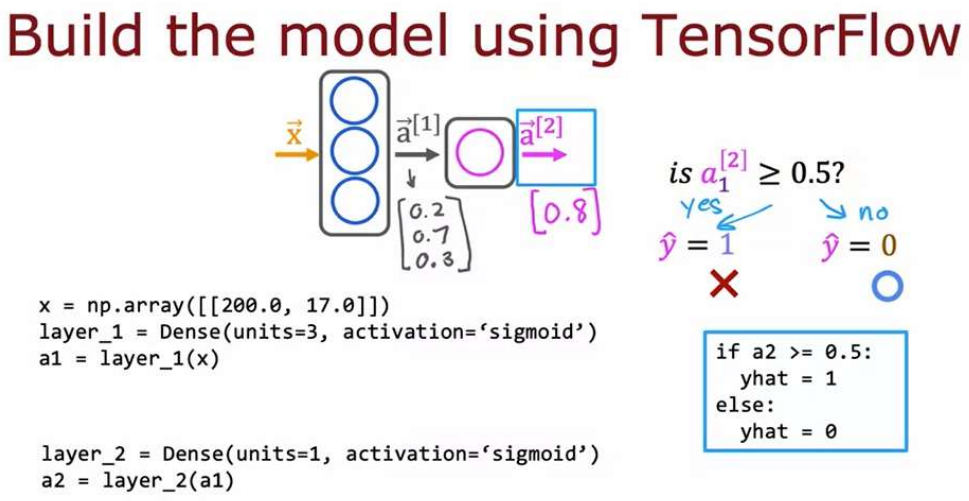
Schauen wir uns in diesem Video an, wie Sie Inferenzcode mit TensorFlow implementieren können. Eines der bemerkenswerten Dinge an neuronalen Netzen ist, dass derselbe Algorithmus auf so viele verschiedene Anwendungen angewendet werden kann. In diesem Video und in einigen Laboren, damit Sie sehen können, was das neuronale Netzwerk tut, werde ich ein weiteres Beispiel verwenden, um die Schlussfolgerung zu veranschaulichen. Manchmal röste ich Kaffeebohnen auch gerne selbst zu Hause. Mein Favorit sind eigentlich kolumbianische Kaffeebohnen. Kann der Lernalgorithmus dabei helfen, die Qualität der Bohnen zu optimieren, die Sie bei einem solchen Röstprozess erhalten?




Wenn Sie Kaffee rösten, können Sie zwei Parameter steuern: die Temperatur, auf die Sie die Rohkaffeebohnen erhitzen, um sie in schön geröstete Kaffeebohnen zu verwandeln, sowie die Dauer bzw. die Dauer des Röstens der Bohnen. In diesem leicht vereinfachten Beispiel haben wir Datensätze mit unterschiedlichen Temperaturen und unterschiedlicher Dauer sowie Etiketten erstellt, die zeigen, ob es sich bei dem von Ihnen gerösteten Kaffee um wohlschmeckenden Kaffee

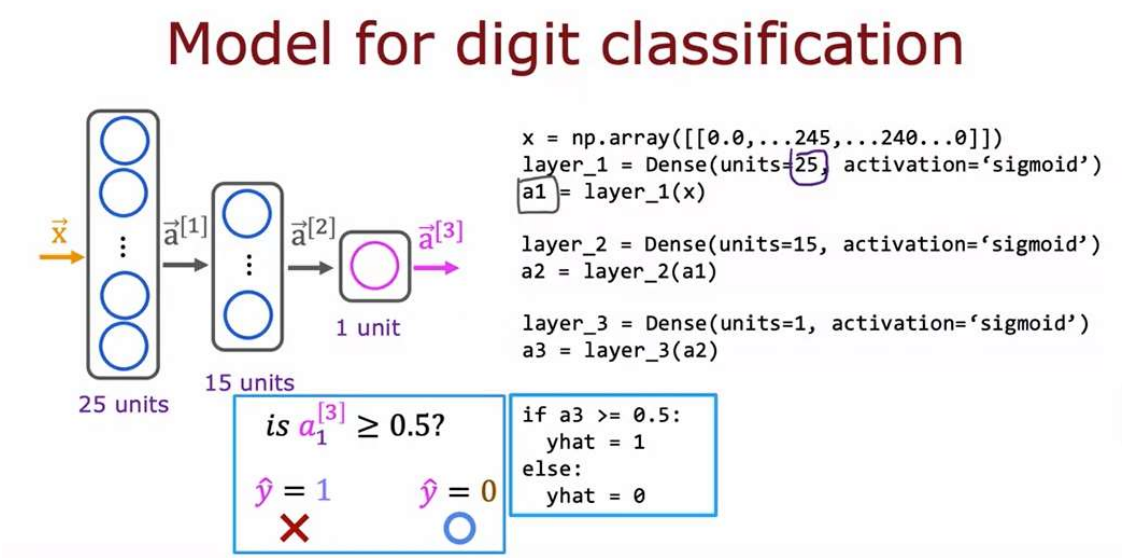
handelt. Wo Kreuz hier ist, entspricht das positive Kreuz y gleich 1 gutem Kaffee, und alle negativen Kreuze entsprechen schlechtem Kaffee.

Man kann sich diesen Datensatz durchaus so vorstellen, dass er nicht geröstet wird und nicht ausreichend gegart ist, wenn man ihn bei einer zu niedrigen Temperatur kocht. Wenn man es nicht lange genug kocht, ist die Dauer zu kurz und es sind auch keine schön gerösteten Bohnen. Wenn Sie die Bohnen schließlich zu lange oder bei einer zu hohen Temperatur kochen, sind die Bohnen verkocht. Es sind ein wenig verbrannte Bohnen. Es gibt auch keinen guten Kaffee. Nur die Punkte innerhalb dieses kleinen Dreiecks entsprechen einem guten Kaffee. Dieses Beispiel ist etwas vom tatsächlichen Kaffeerösten vereinfacht. Auch wenn dieses Beispiel zur Veranschaulichung vereinfacht ist, gab es tatsächlich ernsthafte Projekte, die maschinelles Lernen auch zur Optimierung des Kaffeeröstens nutzten. Der Aufgabe wird ein Merkmalsvektor x mit Temperatur und Dauer zugewiesen, beispielsweise 200 Grad Celsius für 17 Minuten. Wie können wir in einem neuronalen Netzwerk Rückschlüsse ziehen, um uns mitzuteilen, ob diese Temperatur- und Dauereinstellung zu gutem Kaffee führt oder nicht? oder nicht? Es sieht aus wie das. Wir werden x als Array aus zwei Zahlen festlegen. Die Eingabe umfasst 200 Grad Celsius und 17 Minuten. Dann erstellen Sie Schicht 1 als diese erste verborgene Schicht, das neuronale Netzwerk, als dichte offene Klammereinheiten 3, das heißt drei Einheiten oder drei verborgene Einheiten in dieser Schicht, wobei als Aktivierungsfunktion die Sigmoidfunktion verwendet wird. Dense ist ein anderer Name für die Schichten eines neuronalen Netzwerks, die wir bisher kennengelernt haben. Wenn Sie mehr über neuronale Netze erfahren, lernen Sie auch andere Arten von Schichten kennen. Aber vorerst verwenden wir für alle unsere Beispiele nur die dichte Ebene, den Ebenentyp, den Sie in den letzten Videos kennengelernt haben. Als nächstes berechnen Sie a1, indem Sie Schicht 1, die eigentlich eine Funktion ist, nehmen und diese Funktionsschicht 1 auf die Werte von x anwenden. Auf diese Weise erhalten Sie a1, eine Liste mit drei Zahlen, da Schicht 1 drei Einheiten hatte. Zur Veranschaulichung kann a1 hier also 0,2, 0,7, 0,3 sein . Als nächstes wäre die zweite verborgene Schicht, Schicht 2, dicht. Diesmal hat es nun eine Einheit und wieder eine Sigmoid-Aktivierungsfunktion, und Sie können dann a2 berechnen, indem Sie diese Schicht-2-Funktion auf die Aktivierungswerte von Schicht 1 bis a1 anwenden. Das ergibt den Wert von a2, der zur Veranschaulichung vielleicht 0,8 beträgt. Wenn Sie den Schwellenwert schließlich auf 0,5 festlegen möchten, können Sie einfach testen, ob a2 größer und gleich 0,5 ist, und y-hat gleich eins oder null positives oder negatives Kreuz entsprechend festlegen.




So führen Sie mithilfe von TensorFlow Inferenzen im neuronalen Netzwerk durch. Es gibt einige zusätzliche Details, auf die ich hier nicht eingegangen bin, z. B. das Laden der TensorFlow-Bibliothek

und das Laden der Parameter w und b des neuronalen Netzwerks. Aber wir werden das im Labor noch einmal durchgehen. Schauen Sie sich unbedingt das Labor an. Dies sind jedoch die wichtigsten Schritte für die Vorwärtsausbreitung bei der Berechnung von a1 und a2 und optional des Schwellenwerts a2. Schauen wir uns ein weiteres Beispiel an und kehren wir zum Problem der handschriftlichen Ziffernklassifizierung zurück.

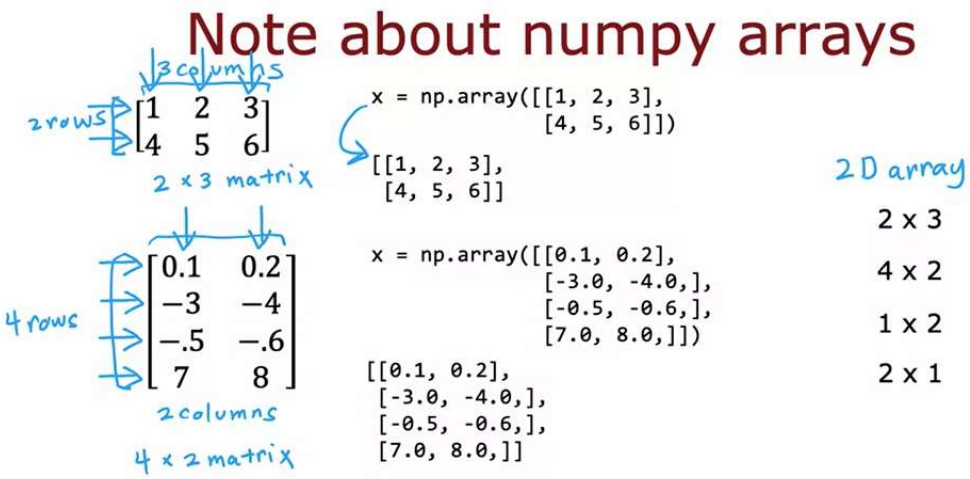



In diesem Beispiel ist x eine Liste der Pixelintensitätswerte. x ist also gleich einem Numpy-Array dieser Liste von Pixelintensitätswerten. Um dann einen Schritt der Vorwärtsausbreitung zu initialisieren und durchzuführen, ist Schicht 1 eine dichte Schicht mit 25 Einheiten und der Sigmoid-Aktivierungsfunktion. Anschließend berechnen Sie, dass a1 der auf x angewendeten Layer-1-Funktion entspricht. Um eine Schlussfolgerung über die zweite Ebene zu erstellen und durchzuführen, richten Sie auf ähnliche Weise Ebene 2 wie folgt ein und berechnen dann a2 als Ebene 2, angewendet auf a1. Schicht 3 schließlich ist die dritte und letzte dichte Schicht. Schließlich können Sie optional den Schwellenwert a3 festlegen, um eine binäre Vorhersage für y-hat zu erstellen. Das ist die Syntax zum Ausführen von Inferenzen in TensorFlow. Eine Sache, auf die ich kurz angespielt habe, ist die Struktur der Numpy-Arrays. TensorFlow behandelt Daten auf eine bestimmte Art und Weise, die wichtig ist, um richtig zu sein. Schauen wir uns im nächsten Video an, wie TensorFlow mit Daten umgeht.

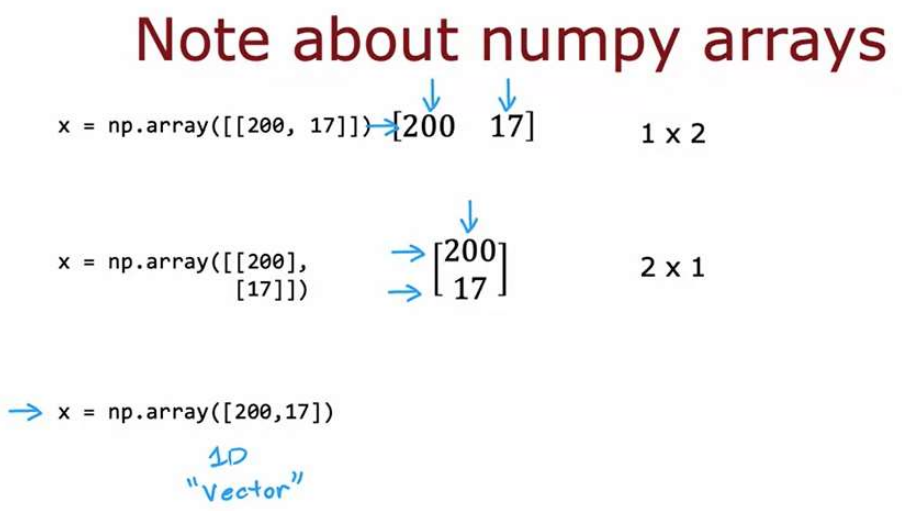
## Daten in TensorFlow

In diesem Video möchte ich mit Ihnen Schritt für Schritt erläutern, wie Daten in NumPy und in TensorFlow dargestellt werden. Damit Sie bei der Implementierung neuer neuronaler Netze über einen konsistenten Rahmen verfügen, um darüber nachzudenken, wie Sie Ihre Daten darstellen. Eines der bedauerlichen Dinge an der heutigen Art und Weise, wie Dinge im Code erledigt werden, ist, dass NumPy vor vielen, vielen Jahren zum ersten Mal erstellt wurde und sich zu einer Standardbibliothek für lineare Algebra und Python entwickelte. Und viel später entwickelte das Google-Gehirnteam, das Team, das ich gegründet und einst geleitet hatte, TensorFlow. Daher gibt es leider einige Inkonsistenzen zwischen der Datendarstellung in NumPy und in TensorFlow. Daher ist es gut, diese Konventionen zu kennen, damit Sie den richtigen Code implementieren und hoffentlich die Dinge in Ihren neuronalen Netzwerken zum Laufen bringen können. Schauen wir uns zunächst an, wie TensorFlow Daten darstellt. Sehen wir uns an, dass Sie einen Datensatz wie diesen aus dem Kaffeebeispiel haben. Ich habe erwähnt, dass Sie x wie folgt schreiben würden. Warum haben Sie hier also diese doppelte eckige Klammer? Werfen wir einen Blick darauf, wie NumPy Vektoren und Matrizen speichert. Wenn Sie der Meinung sind, dass Matrizen und Vektoren komplizierte mathematische Konzepte sind, machen Sie sich darüber keine Sorgen. Wir gehen einige konkrete

Beispiele durch und Sie können alles tun, was Sie mit Matrizen und Vektoren tun müssen, um Ihre Netzwerke umzusetzen. Beginnen wir mit einem Beispiel einer Matrix. Hier ist eine Matrix mit 2 Zeilen und 3 Spalten.



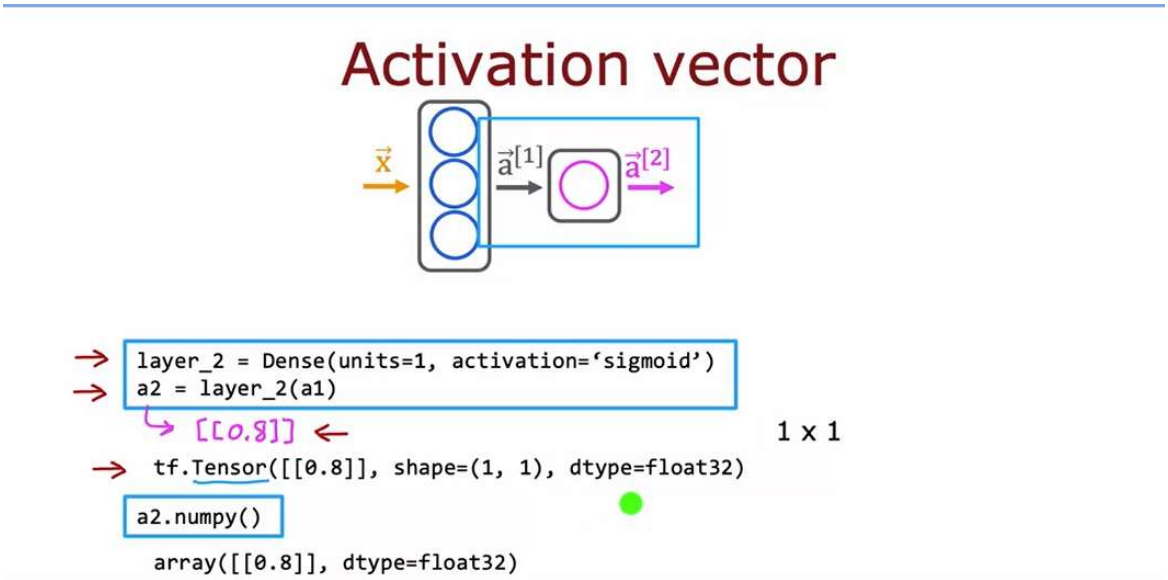

Beachten Sie, dass es eine, zwei Zeilen und 1, 2, 3 Spalten gibt. Wir nennen dies also eine 2 x 3-Matrix. Daher besteht die Konvention darin, dass die Dimension der Matrix als Anzahl der Zeilen mal Anzahl der Spalten geschrieben wird. Im Code zum Speichern dieser Matrix, dieser 2 x 3-Matrix, schreiben Sie einfach x = np.array dieser Zahlen wie folgt. Wo Sie bemerken, dass die eckige Klammer Ihnen sagt, dass 1, 2, 3 die erste Zeile dieser Matrix und 4, 5, 6 die zweite Zeile dieser Matrix ist. Und dann gruppiert diese offene eckige Klammer die erste und die zweite Zeile. Dies setzt x also auf das Zahlenarray. Die Matrix ist also nur ein zweidimensionales Array von Zahlen. Schauen wir uns noch ein Beispiel an, hier habe ich eine weitere Matrix geschrieben. Wie viele Rollen und wie viele Spalten hat dies? Nun, Sie können dies als eine, zwei, drei, vier Zeilen zählen und es hat eine, zwei Spalten. Das ist also eine Matrix aus der Anzahl der Zeilen mal der Anzahl der Spalten, also eine 4 x 2-Matrix. Um dies im Code zu speichern, schreiben Sie x equal np.array und dann diese Syntax hier, um diese vier Matrixzeilen in der Variablen x zu speichern. Dadurch entsteht ein 2D-Array dieser acht Zahlen. Matrizen können unterschiedliche Dimensionen haben. Sie haben ein Beispiel für eine 2 x 3-Matrix und die 4 x 2-Matrix gesehen. Eine Matrix kann auch andere Dimensionen wie 1 x 2 oder 2 x 1 haben. Beispiele dafür sehen wir auf der nächsten Folie. Was wir also zuvor gemacht haben, als wir x als Eingabe-Feature-Vektoren festgelegt haben, war x gleich np.array mit zwei eckigen Klammern, 200, 17. Und was das bewirkt, ist, dass dadurch eine 1 x 2-Matrix erstellt wird, die nur eine ist Zeile und zwei Spalten. Schauen wir uns ein anderes Beispiel an: Wenn Sie x als np.array definieren würden, es aber jetzt so schreiben, entsteht eine 2 x 1-Matrix mit zwei Zeilen und einer Spalte. Weil die erste Zeile nur die Zahl 200 und die zweite Zeile nur die Zahl 17 ist. Das hat also die gleichen Zahlen, aber in einer 2 x 1- statt einer 1 x 2-Matrix. Genug dieses Beispiel oben wird auch als Zeilenvektor bezeichnet, es handelt sich um einen Vektor, der nur aus einer einzelnen Zeile besteht. Und dieses Beispiel wird auch Spaltenvektor genannt, weil dieser Vektor nur eine einzige Spalte hat.




Und der Unterschied zwischen der Verwendung doppelter eckiger Klammern wie dieser und einer einzelnen, eckigen Klammer wie dieser besteht darin, dass bei den beiden Beispielen über 2D-Arrays eine der Dimensionen zufällig 1 ist. Dieses Beispiel führt zu einem 1D-Vektor. Dies ist also nur ein 1D-Array ohne Zeilen oder Spalten, obwohl wir x vereinbarungsgemäß als Spalte wie diese festlegen können. Vergleichen wir dies also mit dem, was wir zuvor im ersten Kurs gemacht hatten, nämlich x so mit einer einzelnen eckigen Klammer zu schreiben. Und das führte zu dem, was in Python ein 1D-Vektor anstelle einer 2D-Matrix genannt wird. Und das ist technisch gesehen nicht 1 x 2 oder 2 x 1, sondern nur ein lineares Array ohne Zeilen oder Spalten, sondern nur eine Liste von Zahlen. Während wir im ersten Kurs mit linearer Regression und logistischer Regression arbeiten, verwenden wir diese 1D-Vektoren, um die Eingabemerkmale x darzustellen. Bei TensorFlow besteht die Konvention darin, Matrizen zur Darstellung der Daten zu verwenden. Und warum gibt es diese Umschaltkonventionen? Nun stellt sich heraus, dass TensorFlow für die Verarbeitung sehr großer Datenmengen konzipiert wurde und durch die Darstellung der Daten in Matrizen anstelle von 1D-Arrays TensorFlow intern etwas recheneffizienter macht. Kehren wir also zu unserem ursprünglichen Beispiel für das erste Training zurück, Beispiel in diesem Datensatz mit Merkmalen von 200 °C in 17 Minuten, wir wurden so dargestellt. Das ist also tatsächlich eine 1 x 2-Matrix, die zufällig eine Zeile und zwei Spalten zum Speichern der Zahlen 217 hat. Und falls Ihnen das nach vielen Details und wirklich komplizierten Konventionen vorkommt, machen Sie sich darüber keine Sorgen klarer werden. Und in den optionalen Laboren und in den Übungslaboren können Sie sich die konkreten Umsetzungen des Codes selbst anschauen. Zurück zum Code zur Ausführung zur Ausbreitung oder Beeinflussung im neuronalen Netzwerk. Wenn Sie berechnen, dass a1 gleich der auf x angewendeten Ebene 1 ist, was ist dann a1? Nun, a1 wird tatsächlich so sein, weil die drei Zahlen tatsächlich eine 1 x 3-Matrix sein werden. Und wenn Sie a1 ausdrucken, erhalten Sie etwa Folgendes: tf.tensor 0,2, 0,7, 0,3 als Form von 1 x 3, 1, 3 bedeutet, dass dies eine 1 x 3-Matrix ist. Und das ist TensorFlows Art zu sagen, dass es sich um eine Gleitkommazahl handelt, was bedeutet, dass es sich um eine Zahl handelt, die einen Dezimalpunkt haben kann, der 32 Bit Speicher in Ihrem Computer verwendet. Dort befindet sich die Gleitkommazahl 32. Und was ist der Tensor? Ein Tensor ist hier ein Datentyp, den das TensorFlow-Team erstellt hat, um Berechnungen auf Matrizen effizient zu speichern und durchzuführen. Wenn Sie also einen Tensor sehen, denken Sie einfach an die Matrix auf diesen wenigen Folien. Technisch gesehen ist ein Tensor etwas allgemeiner als die Matrix, aber für die Zwecke dieses Kurses stellen Sie sich den Tensor lediglich als eine Möglichkeit zur Darstellung von Matrizen vor. Denken Sie also daran, dass ich zu Beginn dieses Videos gesagt habe, dass es die TensorFlow-Methode zur Darstellung der Matrix und die NumPy-Methode zur Darstellung der Matrix gibt. Dies ist ein Artefakt der Entstehungsgeschichte von NumPy und TensorFlow, und leider gibt es zwei Möglichkeiten, eine

Matrix darzustellen, die in diese Systeme integriert wurden. Und tatsächlich, wenn Sie a1, einen Tensor, nehmen und ihn zurück in ein NumPy-Array konvertieren möchten, können Sie dies mit dieser Funktion a1.numpy tun. Und es nimmt dieselben Daten und gibt sie in Form eines NumPy-Arrays und nicht in Form eines TensorFlow-Arrays oder einer TensorFlow-Matrix zurück. Schauen wir uns nun an, wie die Aktivierungsausgabe der zweiten Ebene aussehen würde. Hier ist der Code, den wir zuvor hatten: Schicht 2 ist eine dichte Schicht mit einer Einheit und Sigmoidaktivierung und a2 wird berechnet, indem Schicht 2 genommen und auf a1 angewendet wird. Was ist also a2? A2, vielleicht eine Zahl wie 0,8 und technisch gesehen ist dies eine 1 x 1-Matrix, ein 2D-Array mit einer Zeile und einer Spalte und entspricht daher dieser Zahl 0,8. Und wenn Sie a2 ausdrucken, sehen Sie, dass es sich um einen TensorFlow-Tensor mit nur einem Element und einer Zahl von 0,8 handelt und dass es sich um eine 1 x 1-Matrix handelt. Und wieder handelt es sich um eine Float32-Dezimalzahl, die 32 Bit im Computerspeicher belegt. Auch hier können Sie mit a2.numpy von einem Tensorflow-Tensor in eine NumPy-Matrix konvertieren und daraus wieder ein NumPy-Array machen, das so aussieht.

---

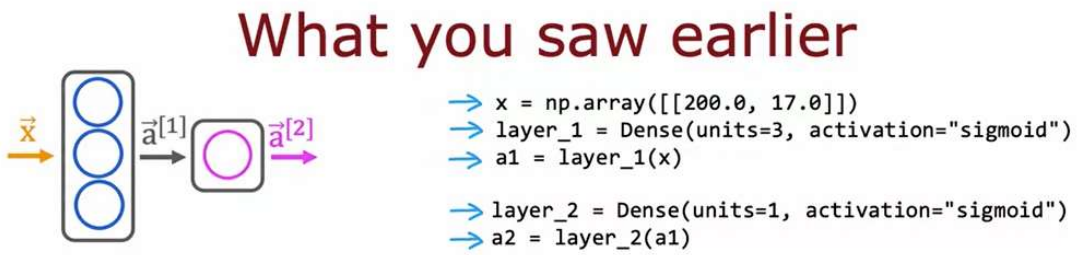



Das gibt Ihnen hoffentlich einen Eindruck davon, wie Daten in TensorFlow und NumPy dargestellt werden. Ich bin es gewohnt, Daten in NumPy zu laden und zu manipulieren, aber wenn Sie ein NumPy-Array an TensorFlow übergeben, konvertiert TensorFlow es gerne in sein eigenes internes Format. Der Tensor arbeitet dann effizient mit Tensoren. Und wenn Sie die Daten wieder auslesen, können Sie sie als Tensor behalten oder zurück in ein NumPy-Array konvertieren. Ich finde es etwas bedauerlich, dass wir aufgrund der Entwicklungsgeschichte dieser Bibliotheken diese zusätzliche Konvertierungsarbeit durchführen mussten, obwohl die beiden Bibliotheken eigentlich recht gut zusammenarbeiten können. Aber wenn Sie hin und her konvertieren, egal ob Sie ein NumPy-Array oder einen Tensor verwenden, sollten Sie beim Schreiben von Code darauf achten. Als nächstes nehmen wir das Gelernte und setzen es zusammen, um tatsächlich ein neuronales Netzwerk aufzubauen. Schauen wir uns das im nächsten Video an.

## Aufbau eines neuronalen Netzwerks

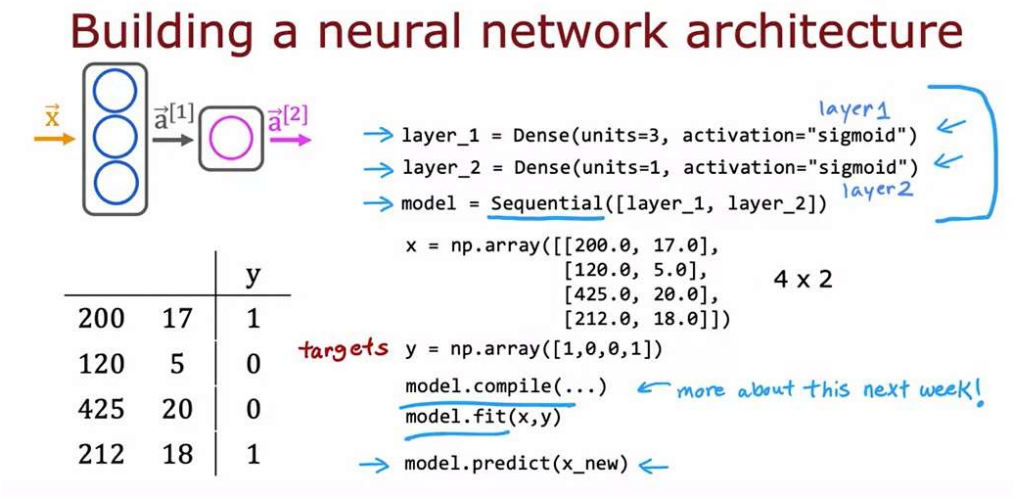
Sie haben also eine Menge Tensor-Flow-Code gesehen und gelernt, wie man im Tensor-Flow eine Ebene aufbaut und wie man im Tensor-Flow Vorwärtsbewegungen durch eine einzelne Ebene durchführt. Und habe auch etwas über Daten in TensorFlow gelernt. Lassen Sie uns alles zusammenfassen und darüber sprechen, wie man in TensorFlow ein neuronales Netzwerk aufbaut.

Dies ist auch das letzte Video zum Tensorfluss für diese Woche. Und in diesem Video lernen Sie auch eine andere Art des Aufbaus eines neuronalen Netzwerks kennen, die sogar noch ein bisschen einfacher ist als das, was Sie bisher gesehen haben. Lassen Sie uns also in das eintauchen, was Sie zuvor gesehen haben.

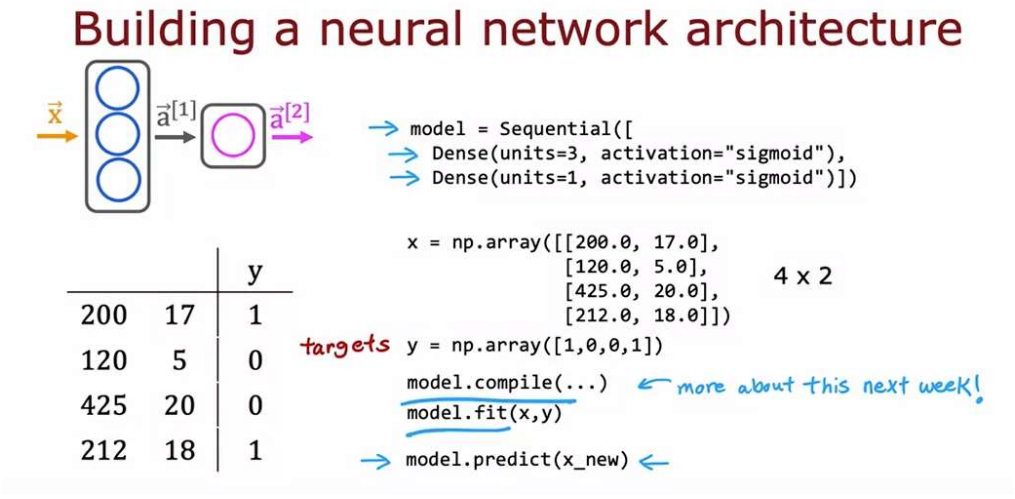



Wenn Sie eine Vorwärtsstütze ausführen möchten, initialisieren Sie die Daten X, erstellen Schicht eins, berechnen dann eine Eins, erstellen dann Schicht zwei und berechnen eine Zwei. Dies war also zu dieser Zeit eine explizite Methode, um eine Vorwärtsberechnung auf einer Ebene durchzuführen. Es stellt sich heraus, dass der Tensorfluss eine andere Art der Implementierung von Vorwärtsprops als auch des Lernens bietet. Lassen Sie mich Ihnen eine andere Art des Aufbaus eines neuronalen Netzwerks in TensorFlow zeigen, die dieselbe ist wie zuvor, als Sie Schicht eins und Schicht zwei erstellten. Aber jetzt müssen Sie die Daten nicht manuell erfassen und an Schicht eins übergeben und dann die Aktivierungen von Schicht eins übernehmen und an Schicht zwei übergeben. Stattdessen können wir dem Tensorfluss mitteilen, dass er Schicht eins und Schicht zwei nehmen und aneinanderreihen soll, um ein neuronales Netzwerk zu bilden. Das ist es, was die sequentielle Funktion in TensorFlow macht, nämlich: „Lieber TensorFlow, bitte erstellen Sie ein neuronales Netzwerk für mich, indem Sie diese beiden Ebenen, die ich gerade erstellt habe, nacheinander aneinanderreihen.“ Es stellt sich heraus, dass der Tensorflow mit dem sequentiellen Framework eine Menge Arbeit für Sie erledigen kann. Nehmen wir an, Sie haben links ein Trainingsset wie dieses. Dies ist für das Kaffeebeispiel. Anschließend können Sie die Trainingsdaten als Eingaben X verwenden und in ein Numpy-Array einfügen. Dies hier ist eine Vier-mal-Zwei-Matrix und die Zielbezeichnungen. Y kann dann wie folgt geschrieben werden. Und dies ist nur ein eindimensionales Array der Länge vier Y. Dieser Satz von Zielen kann dann als ein eindimensionales Array wie dieses 1001 gespeichert werden, das vier Zugbeispielen entspricht. Und es stellt sich heraus, dass angesichts der in dieser Matrix X und diesem Array gespeicherten Daten X und Y Y erforderlich sind. Wenn Sie dieses neuronale Netzwerk trainieren möchten, müssen Sie nur die Funktionen aufrufen, die Sie zum Aufrufen des Modells benötigen, um sie mit einigen Punkten zu kompilieren Parameter. Wir werden nächste Woche mehr darüber besprechen, also machen Sie sich vorerst keine Sorgen. Und dann müssen Sie das Modell Dot Fit XY aufrufen, das den Tensorfluss anweist, dieses neuronale Netzwerk, das durch sequentielles Aneinanderreihen der Schichten eins und zwei erstellt wird, zu nehmen und es anhand der Daten X und Y zu trainieren. Aber wir werden lernen, wie Aber wir werden nächste Woche die Details darüber erfahren, wie das geht, und schließlich: Wie führt man Rückschlüsse auf dieses neuronale Netzwerk aus? Wie machen Sie Forward Prop, wenn Sie ein neues Beispiel haben, sagen wir X new, bei dem es sich um ein NP-Array mit diesen beiden Funktionen handelt, als Forward Prop auszuführen, anstatt es Schicht für Schicht selbst ausführen zu müssen? Sie müssen nur „Model“ aufrufen Vorhersage auf X neu und dies gibt den entsprechenden Wert einer Zwei für Sie aus, wenn dieser Eingabewert von Jetzt möchte ich diese drei Codezeilen ergänzen und sie noch ein wenig weiter vereinfachen, nämlich beim Codieren in Tensorflow. Konventionell weisen wir die beiden Schichten nicht explizit zwei Variablen zu, Schicht eins und Schicht zwei, wie folgt. Aber vereinbarungsgemäß würde ich normalerweise einen Code wie diesen schreiben, wenn wir sagen,

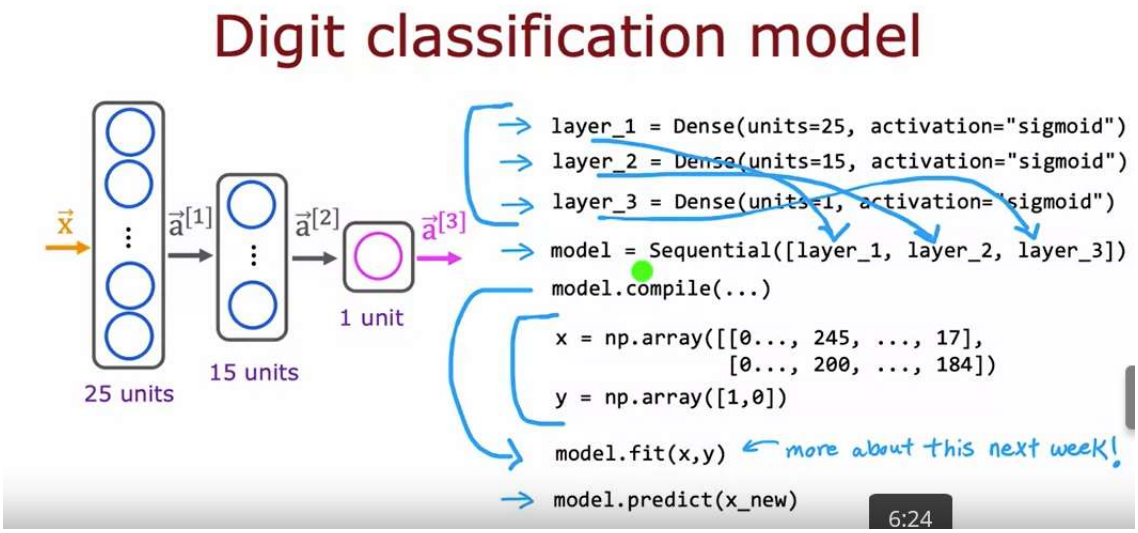
dass das Modell ein sequentielles Modell aus einigen aneinandergereihten Schichten ist. In der Reihenfolge ist die erste Schicht eine dichte Schicht mit drei Einheiten und einer Sigmoid-Aktivierung und die zweite Schicht eine dichte Schicht mit einer Einheit und wiederum einer Sigmoid-Aktivierungsfunktion. Wenn Sie sich also den Tensorflusscode anderer ansehen, sehen Sie oft, dass er eher so aussieht, als dass es eine explizite Zuweisung zu diesen Variablen der ersten und zweiten Ebene gibt. Und das ist es. Dies ist im Wesentlichen der Code, den Sie benötigen, um in TensorFlow ein neuronales Netzwerk zu trainieren und daraus Rückschlüsse zu ziehen. Nächste Woche werden wir noch einmal mehr über die Trainingsteile dieser beiden kombinierten Compiler- und Fit-Funktionen sprechen. Wiederholen wir dies auch für das Beispiel der Ziffernklassifizierung. Vorher hatten wir also X, in dieser Eingabeebene ist eins eine Ebene, die einer Eins entspricht. Sie möchten auf und weisen Sie Tensor Flow an, die Schichten für Sie zu einem neuen Netzwerk zusammenzufassen, und zwar wie zuvor. Anschließend können Sie die Daten in der Matrix speichern, die Kompilierungsfunktion ausführen und das Modell wie folgt anpassen. Mehr dazu nächste Woche. Um schließlich Rückschlüsse zu ziehen oder Vorhersagen zu treffen, können Sie das Modell „Prediction on Im Allgemeinen nehmen Sie einfach diese Ebenen und fügen sie direkt in die sequentielle Funktion ein.




Am Ende erhalten Sie diesen kompakteren Code, der nur den Tensorfluss angibt.



Erstellen Sie für mich ein Modell, das diese drei Schichten nacheinander aneinanderreiht, und dann funktioniert der Rest des Codes genauso wie zuvor. Auf diese Weise haben Sie in TensorFlow ein neuronales Netzwerk aufgebaut. Jetzt weiß ich, dass Sie beim Erlernen dieser Techniken manchmal von jemandem gebeten werden, diese fünf Codezeilen zu implementieren, und dann geben Sie fünf Codezeilen ein, und dann sagt jemand Glückwünsche mit nur fünf Codezeilen. Sie haben dieses verrückt komplizierte neuronale Netzwerk auf dem neuesten Stand der Technik aufgebaut und manchmal fragen Sie sich: Was genau habe ich mit nur diesen fünf Codezeilen gemacht? Eine Sache, die ich Ihnen von der Spezialisierung auf maschinelles Lernen mitnehmen möchte, ist die Möglichkeit, modernste Bibliotheken wie Tensor Flow zu verwenden, um Ihre Arbeit effizient zu erledigen. Aber ich möchte nicht wirklich, dass Sie nur fünf Codezeilen aufrufen und nicht wirklich wissen, was der Code tatsächlich unter der Haube tut. Im nächsten Video lasse ich Sie also noch einmal zurück und teile mit Ihnen, wie Sie die Umsetzung von Grund auf selbst durchführen können. Vorwärtspropagierung in Python, damit Sie das Ganze in der Praxis selbst nachvollziehen können. Die meisten Ingenieure für maschinelles Lernen implementieren die Vorwärtsausbreitung in Python nicht wirklich, so dass wir oft nur Bibliotheken wie Tensorflow und Pytorch verwenden, aber weil ich möchte, dass Sie selbst verstehen, wie diese Algorithmen funktionieren, damit Sie selbst darüber nachdenken können, wenn etwas schief geht, Was Sie möglicherweise ändern müssen, was wahrscheinlich funktioniert, was weniger wahrscheinlich funktioniert.

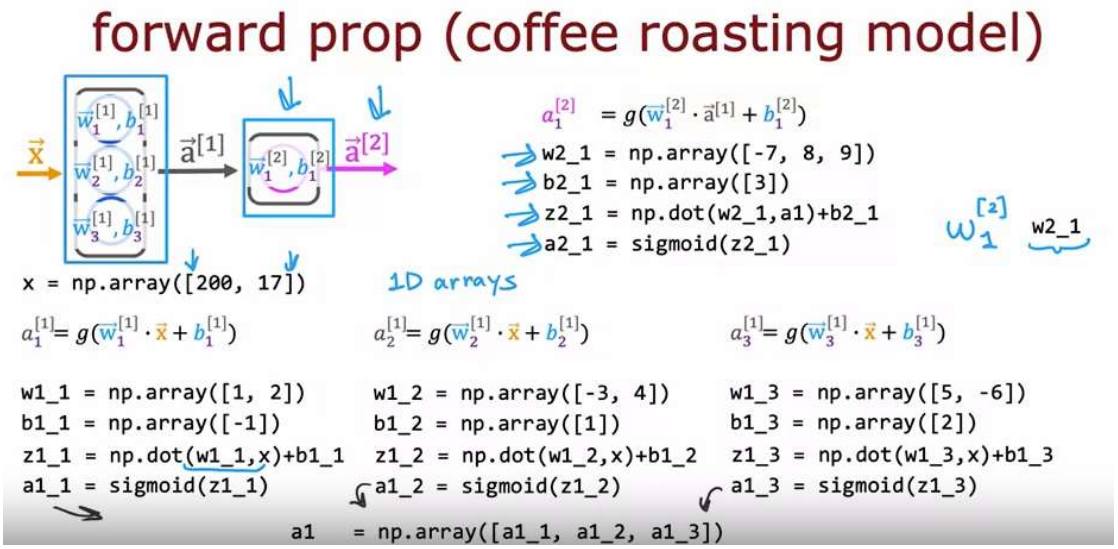



Lassen Sie uns auch durchgehen, was Sie für die Weitergabe von Grund auf implementieren müssten, denn auf diese Weise möchte ich Sie im Hinterkopf haben, selbst wenn Sie eine Bibliothek aufrufen und dafür sorgen, dass sie effizient läuft und großartige Dinge in Ihrer Anwendung bewirkt um auch ein tieferes Verständnis dafür zu bekommen, was Ihr Code tatsächlich tut, also fahren wir mit dem nächsten Video fort.

## Forward Prop in einer einzigen Schicht

Wenn Sie die Vorwärtspropagierung selbst von Grund auf in Python implementieren müssten, wie würden Sie dabei vorgehen und zusätzlich ein Gespür dafür gewinnen, was in Bibliotheken wie TensorFlow und PyTorch wirklich vor sich geht? Wenn Sie eines Tages beschließen, etwas noch Besseres als TensorFlow und PyTorch zu bauen, vielleicht haben Sie jetzt eine bessere Idee, dann empfehle ich den meisten Menschen nicht wirklich, dies zu tun. Aber vielleicht wird eines Tages jemand ein noch besseres Framework als TensorFlow und PyTorch entwickeln, und wer auch immer das tut, muss diese Dinge am Ende möglicherweise selbst von Grund auf implementieren. Werfen wir also einen Blick darauf. Auf dieser Folie werde ich ziemlich viel Code durchgehen, und Sie sehen den gesamten Code später im optionalen Labor noch einmal, genau wie im Übungslabor. Machen Sie sich

also keine Sorgen darüber, dass Sie sich zu jeder Codezeile Notizen machen oder jede Codezeile auswendig lernen müssen. Sie sehen diesen Code im Jupiter-Notizbuch im Labor aufgeschrieben und das Ziel dieses Videos besteht lediglich darin, Ihnen den Code zu zeigen, um sicherzustellen, dass Sie verstehen, was er tut. Wenn Sie also zum optionalen Labor und zum Übungslabor gehen und den Code dort sehen, wissen Sie, was zu tun ist, und müssen sich keine Gedanken darüber machen, zu jeder Zeile detaillierte Notizen zu machen.

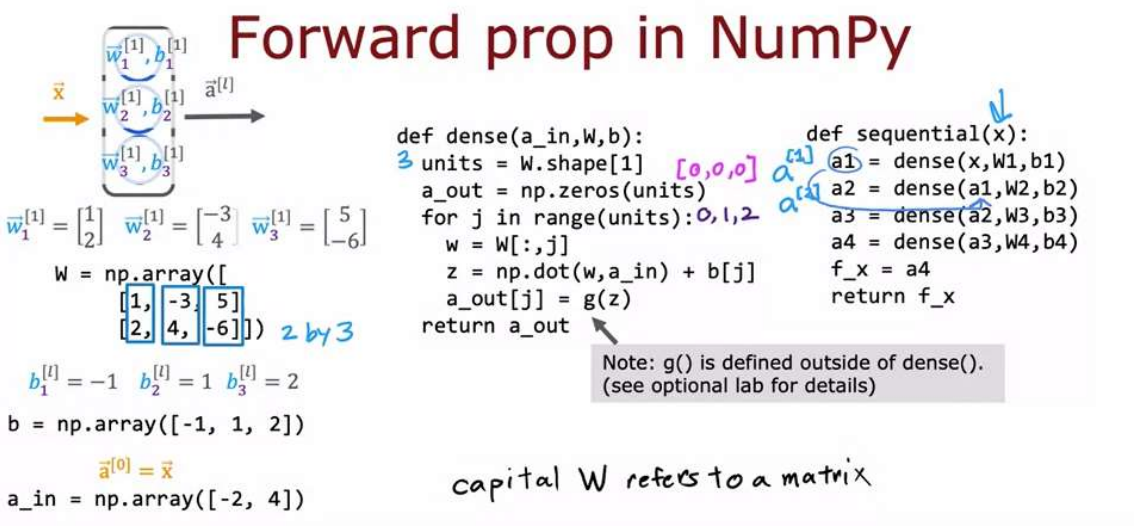



Wenn Sie den Code auf dieser Folie durchlesen und verstehen können, was er tut, ist das alles, was Sie brauchen. Schauen wir uns also an, wie Sie die Vorwärtsstütze in einer einzelnen Ebene implementieren. Wir werden weiterhin das hier gezeigte Kaffeeröstmodell verwenden. Und schauen wir uns an, wie Sie einen Eingabe-Feature-Vektor x nehmen und eine Vorwärtsstütze implementieren würden, um diese Ausgabe a2 zu erhalten. In dieser Python-Implementierung werde ich 1D-Arrays verwenden, um alle diese Vektoren und Parameter darzustellen, weshalb es hier nur eine einzige eckige Klammer gibt. Dies ist ein 1D-Array in Python und nicht eine 2D-Matrix, die wir hatten, als wir doppelte eckige Klammern hatten. Der erste Wert, den Sie also berechnen müssen, ist ein Super-Streifen mit eckiger Klammer 1, Index 1, der der erste Aktivierungswert von a1 ist, und das ist g dieses Ausdrucks hier. Deshalb werde ich auf dieser Folie die Konvention verwenden, dass ich bei einem Begriff wie w2, 1, als Variable w2 und dann den Index 1 darstellen werde. Dieser Unterstrich eins bezeichnet den Index eins, bezeichnet den Index eins, also bedeutet w2 w In eckigen Klammern wird 2 hochgestellt und dann 1 tiefgestellt. Um also a1_1 zu berechnen, haben wir die Parameter w1_1 und b1_1, also beispielsweise 1_2 und -1. Sie würden dann z1_1 als Skalarprodukt zwischen diesem Parameter w1_1 und der Eingabe x berechnen und zu b1_1 addieren, und schließlich ist a1_1 gleich g, der Sigmoidfunktion, die auf z1_1 angewendet wird. Als nächstes berechnen wir a1_2, was wiederum gemäß der hier beschriebenen Konvention a1_2 sein wird, so geschrieben. So ähnlich wie das, was wir links gemacht haben, w1_2 besteht aus zwei Parametern -3, 4, b1_2 ist der Term, b 1, 2 dort drüben, also berechnen Sie z als diesen Term in der Mitte und wenden dann die Sigmoidfunktion an und dann Sie Am Ende erhältst du eine 1_2, und schließlich machst du dasselbe, um a1_3 zu berechnen. Jetzt haben Sie diese drei Werte a1_1, a1_2 und a1_3 berechnet, und wir möchten diese drei Zahlen nehmen und sie in einem Array zusammenfassen, um hier oben a1 zu erhalten, was die Ausgabe der ersten Ebene ist. Und das tun Sie, indem Sie sie mithilfe eines np-Arrays wie folgt gruppieren . Nachdem Sie also a_1 berechnet haben, implementieren wir auch die zweite Ebene. Sie berechnen also die Ausgabe a2, also wird a2 mit diesem Ausdruck berechnet und wir hätten die Parameter w2_1 und b2_1, die diesen Parametern entsprechen. Und dann würden Sie

z als Skalarprodukt zwischen w2_1 und a1 berechnen, b2_1 hinzufügen und dann die Sigmoidfunktion anwenden, um a2_1 zu erhalten, und das war's, so implementieren Sie Forward Prop nur mit Python und NP. Nun gibt es auf dieser Codeseite, die Sie gerade gesehen haben, viele Ausdrücke. Schauen wir uns im nächsten Video an, wie Sie dies vereinfachen können, um Forward Prop für ein allgemeineres neuronales Netzwerk zu implementieren, anstatt es für jedes einzelne Neuron hart zu codieren wie wir es gerade getan haben. Sehen wir uns das also im nächsten Video an.

## Allgemeine Implementierung der Vorwärtsausbreitung

Im letzten Video haben Sie gesehen, wie man Forward Prop in Python implementiert, allerdings durch harte Codierung von Codezeilen für jedes einzelne Neuron. Werfen wir nun einen Blick auf die allgemeinere Implementierung von Forward Prop in Python. Ähnlich wie im vorherigen Video besteht mein Ziel in diesem Video darin, Ihnen den Code zu zeigen, damit Sie wissen, wie Sie ihn interpretieren, wenn Sie ihn in ihrem Übungslabor, in den optionalen Laboren, noch einmal sehen. Machen Sie sich beim Durchgehen dieses Beispiels keine Sorgen darüber, sich zu jeder einzelnen Codezeile Notizen zu machen. Wenn Sie den Code durchlesen und verstehen können, reicht das auf jeden Fall. Sie können eine Funktion schreiben, um eine dichte Schicht zu implementieren, also eine einzelne Schicht eines neuronalen Netzwerks. Ich werde die Dichtefunktion definieren, die als Eingabe die Aktivierung aus der vorherigen Schicht sowie die Parameter w und b für die Neuronen in einer bestimmten Schicht verwendet. Nehmen wir das Beispiel aus dem vorherigen Video: Wenn Schicht 1 drei Neuronen hat und w_1, w_2 und w_3 diese sind, dann stapeln wir alle diese Wellenvektoren in einer Matrix.




Dies wird eine Zwei-mal-Drei-Matrix sein, wobei die erste Spalte der Parameter w_1,1 , die zweite Spalte der Parameter w_1,2 und die dritte Spalte der Parameter w_1,3 ist. Wenn dann auf ähnliche Weise die Parameter b_1,1 gleich minus eins, b_1,2 gleich eins usw. sind, stapeln wir diese drei Zahlen wie folgt in einem 1D-Array b: minus eins, eins zwei. Was die Dense-Funktion tun wird, ist, als Eingaben die Aktivierung aus der vorherigen Ebene zu verwenden, und a könnte hier a_0 sein, was gleich x ist, oder die Aktivierung aus einer späteren Ebene sowie die in Spalten gestapelten w-Parameter, wie gezeigt auf der rechten Seite sowie die b-Parameter, die ebenfalls in einem 1D-Array gestapelt sind, wie dort links gezeigt. Diese Funktion würde eine Aktivierung aus der vorherigen Ebene eingeben und die Aktivierungen aus der aktuellen Ebene ausgeben. Lassen Sie uns den Code dazu Schritt für Schritt durchgehen. Hier ist der Code. Erstens entsprechen die Einheiten W.shape ,1 . Da es sich hier um eine Zwei-mal-Drei-Matrix handelt, beträgt die Anzahl der Spalten drei. Das

entspricht der Anzahl der Einheiten in dieser Ebene. Hier wären die Einheiten gleich drei. Die Betrachtung der Form von w ist nur eine Möglichkeit, die Anzahl der verborgenen Einheiten oder die Anzahl der Einheiten in dieser Ebene herauszufinden. Als nächstes legen wir fest, dass a ein Array aus Nullen mit so vielen Elementen ist, wie es Einheiten gibt. In diesem Beispiel müssen wir drei Aktivierungswerte ausgeben, also initialisiert dies einfach a als Null, Null, Null, ein Array aus drei Nullen. Als nächstes durchlaufen wir eine for- Schleife, um das erste, zweite und dritte Element von a zu berechnen. Für j in Bereichseinheiten geht j von Null zu Einheiten minus eins. Es geht von 0, 1, 2 aus, die Indizierung beginnt bei Null und Python wie üblich. Dieser Befehl w entspricht W Doppelpunkt Komma j, so ziehen Sie die j-te Spalte einer Matrix in Python heraus. Beim ersten Durchlauf dieser Schleife wird die erste Spalte von w und somit w_1 ,1 herausgezogen . Beim zweiten Durchlauf dieser Schleife, wenn Sie die Aktivierung der zweiten Einheit berechnen, wird die zweite Spalte, die w_1, 2 usw. entspricht, zum dritten Mal durch diese Schleife herausgezogen. Dann berechnen Sie z mit der üblichen Formel. Es ist ein Skalarprodukt zwischen diesem Parameter w und der Aktivierung, die Sie erhalten haben, plus b, j. Und dann berechnen Sie die Aktivierungs-Sigmoidfunktion a, j, gleich g, angewendet auf z. Wenn Sie diese Schleife dreimal durchlaufen und sie berechnen, sind die Werte für alle drei Werte dieses Aktivierungsvektors a. Dann kehrst du endlich zurück. Die Dense-Funktion gibt die Aktivierungen der vorherigen Ebene ein und gibt angesichts der Parameter für die aktuelle Ebene die Aktivierungen für die nächste Ebene zurück. Angesichts der dichten Funktion können Sie wie folgt einige dichte Schichten nacheinander aneinanderreihen, um eine Vorwärtsstütze im neuronalen Netzwerk zu implementieren. Wenn die Eingabemerkmale x gegeben sind, können Sie dann die Aktivierungen a_1 so berechnen, dass a_1 gleich der Dichte von x, w_1, b_1 ist, wobei w_1, b_1 hier die Parameter sind, manchmal auch als Gewichte der ersten verborgenen Schicht bezeichnet. Dann können Sie a_2 als Dichte von jetzt a_1 berechnen, das Sie gerade oben berechnet haben. W_2, b-2, die die Parameter oder Gewichte dieser zweiten verborgenen Schicht sind. Berechnen Sie dann a_3 und a_4. Wenn es sich um ein neuronales Netzwerk mit vier Schichten handelt, definieren Sie, dass die Ausgabe f von x gerade gleich a_4 ist, und Sie geben f von x zurück. Beachten Sie, dass ich hier W verwende, da gemäß den Notationskonventionen der linearen Algebra die Verwendung von Großbuchstaben bzw. Großbuchstaben für eine Matrix und Kleinbuchstaben für Vektoren und Skalare erforderlich ist. Da es sich also um eine Matrix handelt, ist dies W. Das ist es. Sie wissen jetzt, wie Sie Forward Prop von Grund auf selbst implementieren können. Sie können diesen gesamten Code sehen, ihn ausführen und selbst im Übungslabor üben, das auch dazu führt. Ich denke, selbst wenn Sie leistungsstarke Bibliotheken wie TensorFlow verwenden, ist es hilfreich zu wissen, wie es unter der Haube funktioniert. Denn für den Fall, dass etwas schief geht, etwas sehr langsam läuft, Sie ein seltsames Ergebnis erhalten oder es so aussieht, als ob ein Fehler vorliegt, werden Sie beim Debuggen Ihres Codes viel effektiver sein, wenn Sie verstehen, was tatsächlich vor sich geht. Wenn ich häufig maschinelle Lernalgorithmen verwende, funktioniert das ehrlich gesagt nicht. Sophie, nicht das erste Mal. Ich finde, dass meine Fähigkeit, meinen Code so zu debuggen, dass er ein TensorFlow-Code oder etwas anderes ist, wirklich wichtig ist, um ein effektiver Ingenieur für maschinelles Lernen zu sein. Selbst wenn Sie TensorFlow oder ein anderes Framework verwenden, hoffe ich, dass Sie dieses tiefere Verständnis für Ihre eigenen Anwendungen und auch für das Debuggen Ihrer eigenen Algorithmen für maschinelles Lernen nützlich finden. Das ist es. Das ist das letzte erforderliche Video dieser Woche mit Code. Im nächsten Video möchte ich auf ein meiner Meinung nach unterhaltsames und faszinierendes Thema eingehen: Welche Beziehung besteht zwischen neuronalen Netzen und KI oder AGI, künstlicher allgemeiner Intelligenz? Dies ist ein kontroverses Thema, aber weil es so ausführlich diskutiert wurde, möchte ich Ihnen einige Gedanken dazu mitteilen. Wenn Sie gefragt werden: Sind neuronale Netze überhaupt auf dem Weg zur Intelligenz auf menschlicher Ebene? Sie haben einen Rahmen, um über diese Frage nachzudenken. Werfen wir einen Blick auf dieses lustige Thema, denke ich, im nächsten Video.

## Gibt es einen Weg zur AGI?

Seit ich als Teenager anfing, mit neuronalen Netzen herumzuspielen, hatte ich einfach das Gefühl, dass der Traum, vielleicht eines Tages ein KI-System zu bauen, das so intelligent ist wie ich selbst oder so intelligent wie ein normaler Mensch, einer der inspirierendsten Träume war KI. Ich halte diesen Traum auch heute noch am Leben. Aber ich denke, dass der Weg dorthin nicht klar ist und sehr schwierig sein könnte. Ich weiß nicht, ob es nur Jahrzehnte dauern würde und ob wir noch zu unseren Lebzeiten Durchbrüche erleben werden, oder ob es Jahrhunderte oder sogar länger dauern könnte, bis wir dorthin gelangen. Werfen wir einen Blick darauf, wie dieser Traum von AGI, der künstlichen allgemeinen Intelligenz, aussieht, und spekulieren wir ein wenig darüber, welche möglichen Wege, unklaren Wege und schwierigen Wege es sein könnte, um eines Tages dorthin zu gelangen. Ich denke, dass es einen unnötigen Hype um AGI oder künstliche allgemeine Intelligenz gegeben hat. Vielleicht liegt ein Grund dafür darin, dass KI tatsächlich zwei sehr unterschiedliche Dinge umfasst. Eine davon ist ANI, was für künstliche enge Intelligenz steht. Hierbei handelt es sich um ein KI-System, das eine Sache, eine begrenzte Aufgabe, manchmal wirklich gut erledigt und unglaublich wertvoll sein kann, wie zum Beispiel den intelligenten Lautsprecher oder selbstfahrende Autos oder die Websuche, oder KI, die auf bestimmte Anwendungen wie die Landwirtschaft oder Fabriken angewendet wird. In den letzten Jahren hat ANI enorme Fortschritte gemacht und schafft, wie Sie wissen, einen enormen Mehrwert in der heutigen Welt. Da es sich bei ANI um eine Teilmenge der KI handelt, ist es aufgrund des schnellen Fortschritts bei ANI logisch, dass auch die KI im letzten Jahrzehnt enorme Fortschritte gemacht hat. Bei der KI gibt es eine andere Idee, nämlich AGI, künstliche allgemeine Intelligenz. Es besteht die Hoffnung, KI-Systeme zu bauen, die alles können, was ein typischer Mensch tun kann. Trotz aller Fortschritte bei ANI und damit enormer Fortschritte bei KI bin ich mir nicht sicher, wie viel Fortschritt wir in Richtung AGI wirklich machen, wenn überhaupt. Ich denke, dass alle Fortschritte in der ANI die Menschen zu dem richtigen Schluss geführt haben, dass es enorme Fortschritte in der KI gibt. Aber das hat einige Leute zu dem Schluss gebracht, dass große Fortschritte in der KI zwangsläufig bedeuten, dass es große Fortschritte in Richtung AGI gibt. Wenn Sie sich sonst noch mit KI und AGI auskennen, kann es manchmal hilfreich sein, dieses Bild zu zeichnen, um einige der Dinge zu erklären, die auch in der KI vor sich gehen, und einige der Ursachen für unnötigen Hype um AGI. Mit dem Aufkommen des modernen Deep Learning haben wir begonnen, Neuronen zu simulieren, und mit immer schnelleren Computern und sogar GPUs können wir noch mehr Neuronen simulieren. Ich glaube, vor vielen Jahren gab es die große Hoffnung, dass, Junge, wenn wir nur viele Neuronen simulieren könnten, wir dann auch das menschliche Gehirn oder so etwas wie ein menschliches Gehirn simulieren könnten und wir wirklich intelligente Systeme hätten. Leider ist es nicht ganz so einfach. Ich denke, dafür gibt es zwei Gründe: Erstens: Wenn man sich die künstlichen neuronalen Netze ansieht, die wir aufbauen, sind sie so einfach, dass eine logistische Regressionseinheit wirklich nichts mit dem zu tun hat, was ein biologisches Neuron tut, sondern so viel einfacher ist als das, was jedes Neuron darin tut Dein oder mein Gehirn tut es. Zweitens glaube ich, dass wir bis heute kaum eine Ahnung davon haben, wie das Gehirn funktioniert. Es gibt immer noch grundlegende Fragen darüber, wie ein Neuron genau die Eingaben auf die Ausgaben abbildet , die wir heute einfach nicht kennen. Der Versuch, dies in einem Computer zu simulieren, geschweige denn eine einzelne logistische Funktion, ist weit von einem genauen Modell dessen entfernt, was das menschliche Gehirn tatsächlich tut. Angesichts unseres derzeit und wahrscheinlich auch in naher Zukunft sehr begrenzten Verständnisses der Funktionsweise des menschlichen Gehirns denke ich, dass allein der Versuch, das menschliche Gehirn als Weg zur AGI zu simulieren, ein unglaublich schwieriger Weg sein wird. Gibt es dennoch Hoffnung, noch zu unseren Lebzeiten einen Durchbruch in der AGI zu erleben? Lassen Sie mich einige Beweise mit Ihnen teilen, die mir helfen, diese Hoffnung zumindest für mich selbst am Leben zu erhalten. Es wurden einige faszinierende Experimente an Tieren durchgeführt, die zeigen oder stark darauf hindeuten, dass dasselbe Stück biologisches Gehirngewebe ein überraschend breites Spektrum an Aufgaben erfüllen

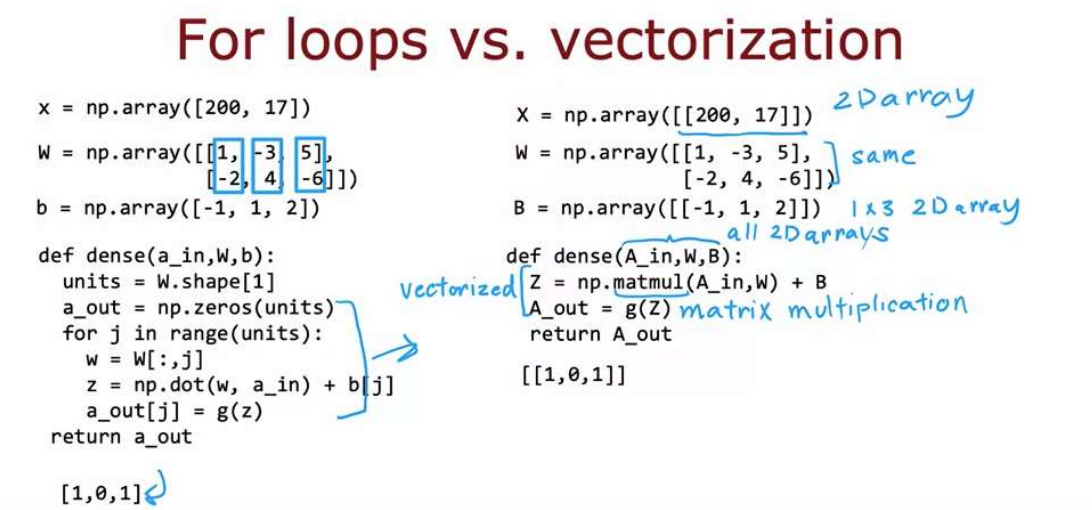
kann. Dies hat zu der Hypothese eines einzigen Lernalgorithmus geführt, dass möglicherweise viel Intelligenz auf einen oder eine kleine Handvoll Lernalgorithmen zurückzuführen sein könnte. Wenn wir nur herausfinden könnten, was dieser eine oder eine kleine Handvoll Algorithmen ist, könnten wir ihn vielleicht eines Tages in einem Computer implementieren. Lassen Sie mich einige Details dieser Experimente mit Ihnen teilen. Dies ist ein Ergebnis von Roe et al. von vor vielen Jahrzehnten. Der hier gezeigte Teil Ihres Gehirns ist Ihre Hörrinde, und Ihr Gehirn ist so verdrahtet, dass es Signale von Ihren Ohren in Form von elektrischen Impulsen an diese Hörrinde weiterleitet, je nachdem, welches Geräusch Ihr Ohr wahrnimmt. Es stellt sich heraus, dass, wenn man das Gehirn eines Tieres neu verkabelt , indem man den Draht zwischen dem Ohr und der Hörrinde durchschneidet und stattdessen Bilder in die Hörrinde einspeist, die Hörrinde das Sehen lernt. Auditiv bezieht sich auf Geräusche, und so lernt dieser Teil des Gehirns, der bei den meisten Menschen das Sehen lernt, wenn ihm verschiedene Daten zugeführt werden, stattdessen das Sehen. Hier ist ein weiteres Beispiel. Dieser Teil Ihres Gehirns ist Ihr somatosensorischer Kortex. Somatosensorisch bezieht sich auf die Berührungsverarbeitung. Wenn Sie das Gehirn auf ähnliche Weise neu verdrahten würden, um die Verbindung von den Berührungssensoren zu diesem Teil des Gehirns zu unterbrechen und stattdessen das Gehirn neu zu verdrahten, um Bilder einzuspeisen, dann lernt der somatosensorische Kortex das Sehen. Es gab eine Reihe solcher Experimente, die zeigten, dass viele verschiedene Teile des Gehirns, je nachdem, welche Daten gegeben werden, sehen, fühlen oder hören lernen können, als ob es vielleicht einen Algorithmus gäbe, der nur davon abhängt Welche Daten oder welche gegeben werden, lernt, diese Eingaben entsprechend zu verarbeiten. Es gibt Systeme, die eine an der Stirn einer Person angebrachte Kamera aufnehmen und diese auf ein Muster von Spannungen in einem Gitter auf der Zunge einer anderen Person abbilden. Durch die Zuordnung eines Graustufenbildes zu einem Muster von Spannungen auf Ihrer Zunge kann dies Menschen helfen, die nicht als Individuen bezeichnet werden, das Sehen mit Ihrer Zunge zu lernen, oder sie haben faszinierende Experimente mit menschlicher Echoortung oder menschlichem Sonar durchgeführt, also Tiere wie Delfine und Fledermäuse verwenden Sonar, um zu sehen, und Forscher haben herausgefunden, dass Menschen manchmal ein gewisses Maß an menschlicher Echoortung erlernen können, wenn man Menschen trainiert, Klickgeräusche zu erzeugen und darauf zu hören, wie diese von der Umgebung reflektiert werden. Oder das ist ein haptischer Gürtel, und mein Forschungslabor in Stanford hat so etwas schon einmal gebaut, aber wenn man einen Ring aus Summern um die Taille legt und ihn mit einem Magnetkompass programmiert, dann sind das die Summer im Norden Die meisten Richtungen schwingen immer langsam, dann bekommt man irgendwie einen Richtungssinn, den manche Tiere haben, aber Menschen nicht. Dann fühlt es sich einfach so an, als würde man herumlaufen und einfach wissen, wo der Norden ist, es fühlt sich nicht so an, als würde dieser Teil meiner Taille summen, es fühlt sich an, als würde man sagen: „Oh, ich weiß, wo der Norden ist.“ Oder es wird einem Frosch ein drittes Auge implantiert, und das Gehirn lernt mit diesem Input einfach mit. Es gab eine Vielzahl von Experimenten wie diese, die nur zeigten, dass das menschliche Gehirn erstaunlich anpassungsfähig ist. Neurowissenschaftler sagen, es sei erstaunlich plastisch, sie meinen nur anpassungsfähig an eine verwirrende Bandbreite an Sensoreingaben, und daher stellt sich die Frage, ob dasselbe Stück Gehirngewebe dazu in der Lage ist Lernen Sie, zu sehen, zu berühren, zu fühlen oder sogar andere Dinge, was ist der Durchschnitt der Benutzer, und können wir diesen Algorithmus replizieren und in einem Computer implementieren? Ich habe Mitleid mit dem Frosch und den anderen Tieren oder denen, die diese Experimente durchgeführt haben, obwohl ich denke, dass die Schlussfolgerungen auch ziemlich faszinierend sind. Bis heute halte ich die Arbeit an AGI für eines der faszinierendsten wissenschaftlichen und technischen Probleme aller Zeiten, und vielleicht werden Sie sich eines Tages dazu entschließen, darüber zu forschen. Allerdings denke ich, dass es wichtig ist, zu viel Hype zu vermeiden. Ich weiß nicht, ob das Gehirn wirklich aus einem oder einer kleinen Handvoll Algorithmen besteht, und selbst wenn es so wäre, habe ich keine Ahnung, und ich glaube nicht, dass irgendjemand weiß, was Der Algorithmus ist es, aber ich habe immer noch

diese Hoffnung, und vielleicht ist es das, und vielleicht könnten wir durch viel harte Arbeit eines Tages eine Annäherung daran finden. Ich finde das immer noch eines der faszinierendsten Themen, ich denke in meiner Freizeit wirklich darüber nach und vielleicht sind Sie eines Tages derjenige, der einen Beitrag zur Lösung dieses Problems leistet. Kurzfristig denke ich, dass maschinelles Lernen und neuronale Netze auch ohne AGI ein sehr leistungsfähiges Werkzeug sind, und selbst ohne den Versuch, Intelligenz auf menschlicher Ebene aufzubauen, sind neuronale Netze meiner Meinung nach ein unglaublich leistungsfähiges Werkzeug. und nützliche Tools für Anwendungen, die Sie möglicherweise erstellen. Das waren die erforderlichen Videos dieser Woche. Herzlichen Glückwunsch, dass Sie in den Lektionen an diesen Punkt gelangt sind. Danach werden wir auch ein paar optionale Videos haben, um etwas tiefer in die effiziente Implementierung neuronaler Netze einzutauchen. Insbesondere in den kommenden optionalen Videos möchte ich Ihnen einige Details zur Vektorisierung von Implementierungen neuronaler Netze mitteilen. Ich hoffe, dass Sie sich diese Videos auch ansehen.

## Wie neuronale Netze effizient implementiert werden

Einer der Gründe dafür, dass Deep-Learning-Forscher im letzten Jahrzehnt in der Lage waren, neuronale Netze zu vergrößern und wirklich große neuronale Netze zu denken, liegt darin, dass neuronale Netze vektorisiert werden können. Sie lassen sich sehr effizient durch Matrizenmultiplikationen umsetzen. Es stellt sich heraus, dass parallele Rechenhardware, einschließlich GPUs, aber auch einige CPU-Funktionen, sehr gut darin ist, sehr große Matrixmultiplikationen durchzuführen.

In diesem Video werfen wir einen Blick darauf, wie diese vektorisierten Implementierungen neuronaler Netze funktionieren.

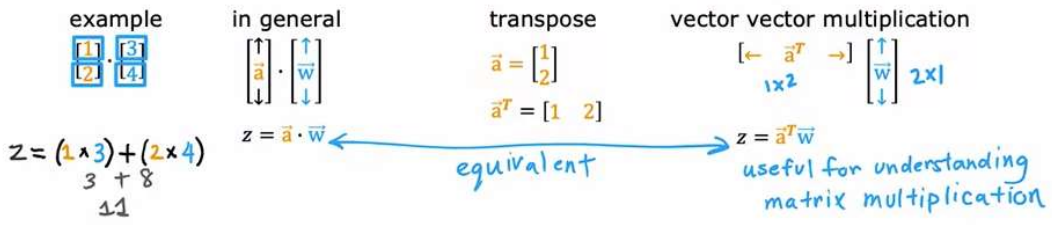



Ohne diese Ideen glaube ich nicht, dass Deep Learning heute auch nur annähernd ein Erfolg und eine Größenordnung wäre. Hier auf der linken Seite sehen Sie den Code, den Sie zuvor gesehen haben und der zeigt, wie Sie Forward Prop oder Forward Propagation in einer einzelnen Ebene implementieren würden. X ist hier die Eingabe, W, die Gewichte des ersten, zweiten und dritten Neurons, sagen wir, Parameter B, und dann ist dies derselbe Code, den wir zuvor gesehen haben. Dadurch werden beispielsweise drei Zahlen ausgegeben. Wenn Sie diese Berechnung tatsächlich durchführen, erhalten Sie 1, 0, 1 . Es stellt sich heraus, dass Sie wie folgt eine vektorisierte Implementierung dieser Funktion entwickeln können. Setzen Sie X gleich diesem Wert. Beachten Sie die doppelten eckigen Klammern. Dies ist jetzt ein 2D-Array, wie in TensorFlow. W ist dasselbe wie

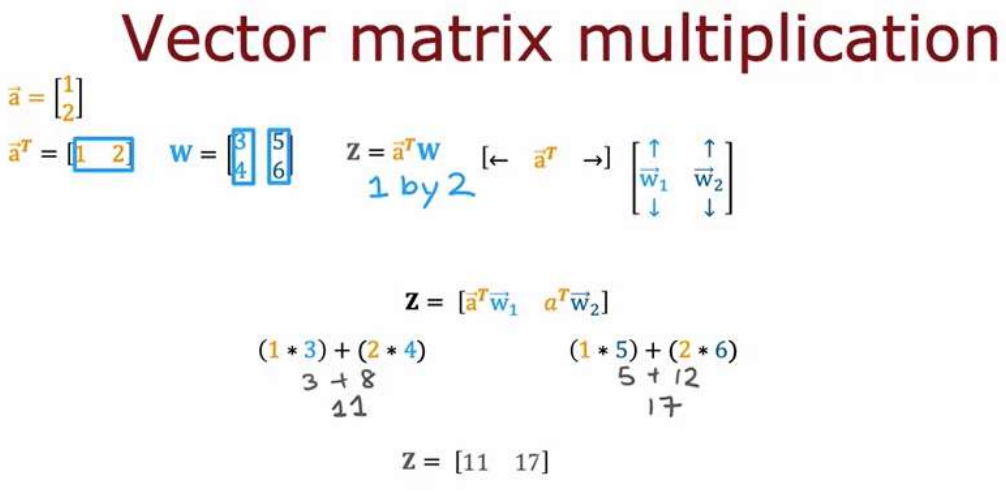
zuvor, und B, ich verwende jetzt B, ist ebenfalls ein eins-mal-drei-2D-Array. Dann stellt sich heraus, dass alle diese Schritte, diese for-Schleife darin, durch nur ein paar Codezeilen ersetzt werden können, Z ist gleich np.matmul. Mit Matmul führt NumPy die Matrixmultiplikation durch. Wobei jetzt X und W beide Matrizen sind und Sie sie einfach miteinander multiplizieren. Es stellt sich heraus, dass in dieser for-Schleife alle diese Codezeilen durch nur ein paar Codezeilen ersetzt werden können, was eine vektorisierte Implementierung dieser Funktion ergibt. Sie berechnen Z, das jetzt wieder eine Matrix ist, als numpy.matmul zwischen A in und W, wobei A in und W hier beide Matrizen sind und matmul ist, wie NumPy eine Matrixmultiplikation durchführt. Es multipliziert zwei Matrizen miteinander und fügt dann die Matrix B hinzu. Dann ist A out gleich der Aktivierungsfunktion g, also der Sigmoidfunktion, elementweise angewendet auf diese Matrix Z, und dann gibt man schließlich A out zurück. So sieht der Code aus. Beachten Sie, dass in der vektorisierten Implementierung alle diese Größen x, die in den Wert von A in sowie W, B sowie Z und A out eingespeist werden, jetzt allesamt 2D-Arrays sind. All dies sind Matrizen. Dies erweist sich als eine sehr effiziente Implementierung eines Schritts der Vorwärtsausbreitung durch eine dichte Schicht im neuronalen Netzwerk. Dies ist Code für eine vektorisierte Implementierung von Forward Prop in einem neuronalen Netzwerk. Aber was macht dieser Code und wie funktioniert er eigentlich? Was macht dieser Matmul eigentlich? In den nächsten beiden Videos, beide ebenfalls optional, gehen wir auf die Matrixmultiplikation und deren Funktionsweise ein. Wenn Sie mit linearer Algebra vertraut sind, wenn Sie mit Vektoren, Matrizen, Transponierungen und Matrixmultiplikationen vertraut sind, können Sie diese beiden Videos getrost überfliegen und zum letzten Video dieser Woche springen. Im letzten Video dieser Woche, ebenfalls optional, gehen wir dann detaillierter darauf ein, wie Matmul Ihnen diese vektorisierte Implementierung ermöglicht. Kommen wir zum nächsten Video, in dem wir einen Blick darauf werfen, was eine Matrixmultiplikation ist.

## Matrix-Multiplikation

Sie wissen, dass eine Matrix nur ein Block oder ein zweidimensionales Array von Zahlen ist. Was bedeutet es, zwei Matrizen zu multiplizieren? Lass uns einen Blick darauf werfen. Um Multiplikationsmatrizen aufzubauen, schauen wir uns zunächst an, wie wir Skalarprodukte zwischen Vektoren bilden. Nehmen wir das Beispiel der Bildung des Skalarprodukts zwischen diesem Vektor 1, 2 und diesem Vektor 3, 4. Wenn z das Skalarprodukt zwischen diesen beiden Vektoren ist, dann berechnen Sie z, indem Sie hier das erste Element mit dem ersten Element multiplizieren, es ist 1 mal 3, plus das zweite Element mal das zweite Element plus 2 mal 4, also ist das nur 3 plus 8, was gleich 11 ist. Im allgemeineren Fall, wenn z das Skalarprodukt zwischen einem Vektor a und einem Vektor w ist, Dann berechnen Sie z, indem Sie das erste Element miteinander multiplizieren, dann die zweiten Elemente miteinander und das dritte usw. und dann alle diese Produkte addieren. Das ist der Vektor, Vektorskalarprodukt. Es stellt sich heraus, dass es eine andere äquivalente Möglichkeit gibt, ein Skalarprodukt zu schreiben, das einen Vektor a, also 1, 2, als Spalte geschrieben hat. Sie können daraus eine Reihe machen.

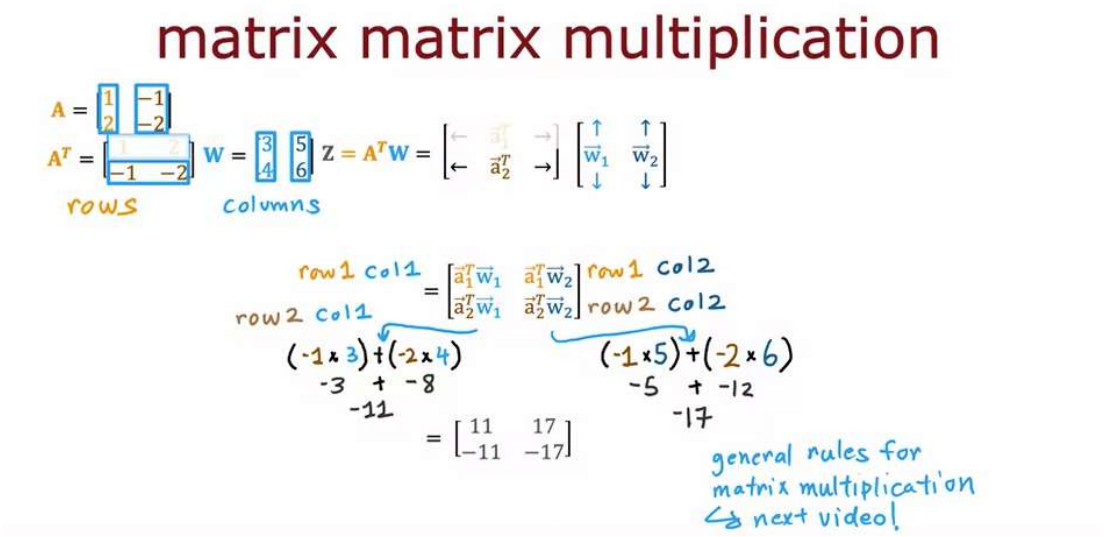



Das heißt, Sie können ihn von einem sogenannten Spaltenvektor in einen Zeilenvektor umwandeln, indem Sie die Transponierte von a nehmen. Die Transponierung des Vektors a bedeutet, dass Sie diesen Vektor nehmen und seine Elemente wie folgt auf die Seite legen. Es stellt sich heraus, dass wenn Sie eine Transponierte multiplizieren, dies ein Zeilenvektor ist, oder Sie können sich dies als eine Eins-mal-Zwei-Matrix mit w vorstellen, die Sie sich jetzt als Zwei-mal-Eins-Matrix vorstellen können. Dann ist z gleich a transponiert mal w und das ist dasselbe wie die Bildung des Skalarprodukts zwischen a und w. Um es noch einmal zusammenzufassen: z entspricht dem Skalarprodukt zwischen a und w und z entspricht einer Transponierten, also einem auf die Seite gelegten Wert, multipliziert mit w, und dies wird für das Verständnis der Matrixmultiplikation nützlich sein. Dass dies nur zwei Möglichkeiten sind, genau dieselbe Berechnung zu schreiben, um zu z zu gelangen. Schauen wir uns nun die Vektormatrixmultiplikation an, bei der man einen Vektor nimmt und einen Vektor mit einer Matrix multipliziert. Auch hier ist der Vektor a 1, 2 und eine Transponierte ist ein auf die Seite gelegtes a. Stellen Sie sich dies also nicht wie eine Zwei-mal-eins-Matrix vor, sondern wird zu einer Eins-mal-zwei-Matrix. Lassen Sie mich nun eine Zwei-mal-Zwei-Matrix w mit diesen vier Elementen 3, 4, 5, 6 erstellen. Wenn Sie Z als Transponierte mal w berechnen möchten.




Mal sehen, wie Sie dabei vorgehen. Es stellt sich heraus, dass Z eine Eins-mal-Zwei-Matrix sein wird, und um den ersten Wert von Z zu berechnen, nehmen wir eine Transponierte, hier 1, 2, und multiplizieren diese mit der ersten Spalte von w, das heißt 3, 4. Um das erste Element von Z zu berechnen, erhalten Sie am Ende 1 mal 3 plus 2 mal 4, was, wie wir zuvor gesehen haben, gleich 11 ist, und daher ist das erste Element von Z 11. Lassen Sie uns herausfinden, was das zweite Element ist von Z. Es stellt sich heraus, dass Sie diesen Vorgang einfach wiederholen, aber jetzt eine

Transponierte mit der zweiten Spalte von w multiplizieren. Um diese Berechnung durchzuführen, haben Sie 1 mal 5 plus 2 mal 6, was gleich 5 plus 12 ist, was 17 ist. Das ist gleich 17. Z ist gleich dieser Eins-mal-Zwei-Matrix, 11 und 17. Nun, Nur noch eine letzte Sache, und dann sind wir am Ende dieses Videos angelangt, in dem es darum geht, wie man die Vektormatrixmultiplikation auf die Matrixmatrixmultiplikation verallgemeinert. Ich habe eine Matrix A mit diesen vier Elementen, die erste Spalte ist 1, 2 und die zweite Spalte ist negativ 1, negativ 2 und ich möchte wissen, wie man eine Transponierte mal w berechnet. Im Gegensatz zur vorherigen Folie ist A jetzt eine Matrix und nicht nur der Vektor, oder die Matrix ist nur ein Satz verschiedener Vektoren, die in Spalten gestapelt sind. Lassen Sie uns zunächst herausfinden, was eine A- Transponierung ist. Um die A-Transponierung zu berechnen, nehmen wir die Spalten von A und legen, ähnlich wie bei der Transponierung eines Vektors, die Spalten auf die Seite, eine Spalte nach der anderen. Die erste Spalte 1, 2 wird zur ersten Zeile 1, 2, legen wir sie einfach auf die Seite, und diese zweite Spalte, negativ 1, negativ 2, wird wie folgt auf die Seite negativ 1, negativ 2 gelegt. Die Art und Weise, wie Sie eine Matrix transponieren, besteht darin, dass Sie die Spalten nehmen und sie einfach eine Spalte nach der anderen auf die Seite legen. Am Ende erhalten Sie eine A- Transponierung . Als nächstes haben wir diese Matrix W, die wir als 3,4 , 5,6 schreiben werden .

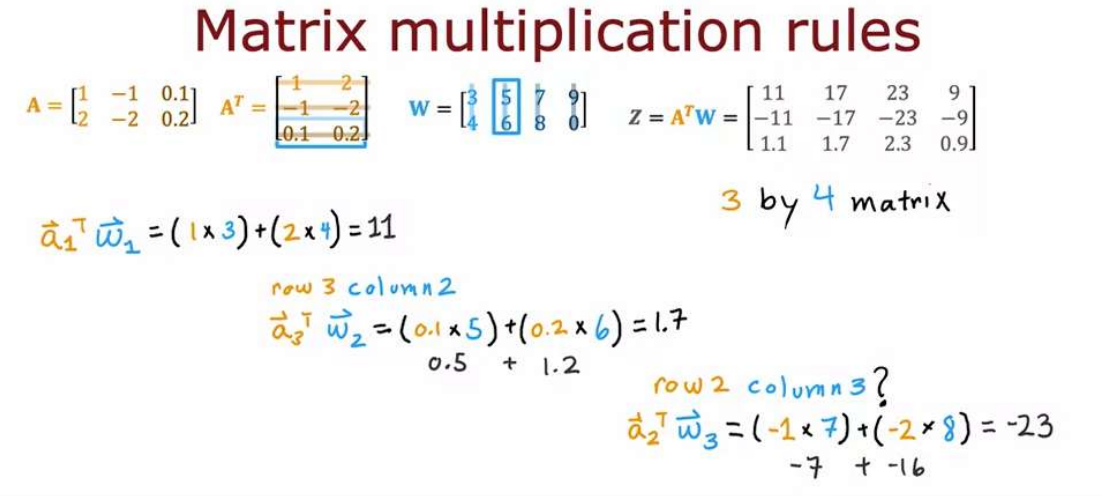



Es gibt eine Spalte 3, 4 und eine Spalte 5, 6. Ich ermutige Sie zum einen, über Matrizen nachzudenken. Zumindest ist es für neuronale Netzwerkimplementierungen nützlich, wenn Sie eine Matrix sehen, an die Spalten der Matrix denken und wenn Sie die Transponierte einer Matrix sehen, stellen Sie sich vor, dass die Zeilen dieser Matrix wie hier dargestellt gruppiert sind, mit A und A transpose sowie W. Jetzt zeige ich Ihnen, wie man A transpose und W multipliziert. Um diese Berechnung durchzuführen, nenne ich die Spalten von A, a_1 und a_2 und das bedeutet, dass a_1 transpose, dies die erste Zeile ist von A transpose, und a_2 transpose ist die zweite Zeile von A transpose. Lassen Sie mich dann wie zuvor die Spalten von W als w_1 und w_2 bezeichnen. Es stellt sich heraus, dass wir zur Berechnung der A-Transponierung W als Erstes die zweite Zeile von A einfach ignorieren, uns einfach auf die erste Zeile von A konzentrieren und diese Zeile 1, 2 nehmen müssen, die a_1 transponiert und ist Multiplizieren Sie das mit W. Wie das geht, wissen Sie bereits aus der vorherigen Folie. Das erste Element ist 1, 2, inneres Produkt oder Skalarprodukt wir haben 3, 4. Das ergibt 3 mal 1 plus 2 mal 4, also 11. Dann ist das zweite Element 1, 2 Ein transponiertes inneres Produkt wir Habe 5, 6. Es gibt 5 mal 1 plus 6 mal 2, also 5 plus 12, also 17. Das ergibt die erste Zeile von Z gleich A- Transponierung W. Alles, was wir getan haben, ist, a_1-Transponierung zu nehmen und das zu multiplizieren von W. Genau das haben wir auf der vorherigen Folie gemacht. Als nächstes vergessen wir zunächst a_1 und schauen uns einfach a_2 an, nehmen a_2 transponiert und

multiplizieren das mit W. Jetzt haben wir a_2 transponiert mal W. Um das zuerst zu berechnen, nehmen wir minus 1 und minus 2 und punktieren das mit 3 . 4. Das ist minus 1 mal 3 plus minus 2 mal 4 und das ergibt minus 11. Dann müssen wir a_2 transponieren mal die zweite Spalte berechnen und haben minus 1 mal 5 plus minus 2 mal 6, und das ergibt sei negativ 17. Am Ende erhalten Sie A transponiert mal W, was dieser Zwei-mal-Zwei-Matrix hier entspricht. Lassen Sie uns über die allgemeine Form der Matrix-Matrix-Multiplikation sprechen. Dies war ein Beispiel dafür, wie man einen Vektor mit einer Matrix multipliziert, oder eine Matrix mit einer Matrix besteht aus vielen Skalarprodukten zwischen Vektoren, die jedoch auf eine bestimmte Weise angeordnet sind, um die Elemente des oberen Z Element für Element zu konstruieren. Ich weiß, das war viel, aber im nächsten Video schauen wir uns die allgemeine Definition einer Matrix-Matrix-Multiplikation an und ich hoffe, dass das auch alles klar macht. Kommen wir zum nächsten Video.

## Regeln für die Matrixmultiplikation

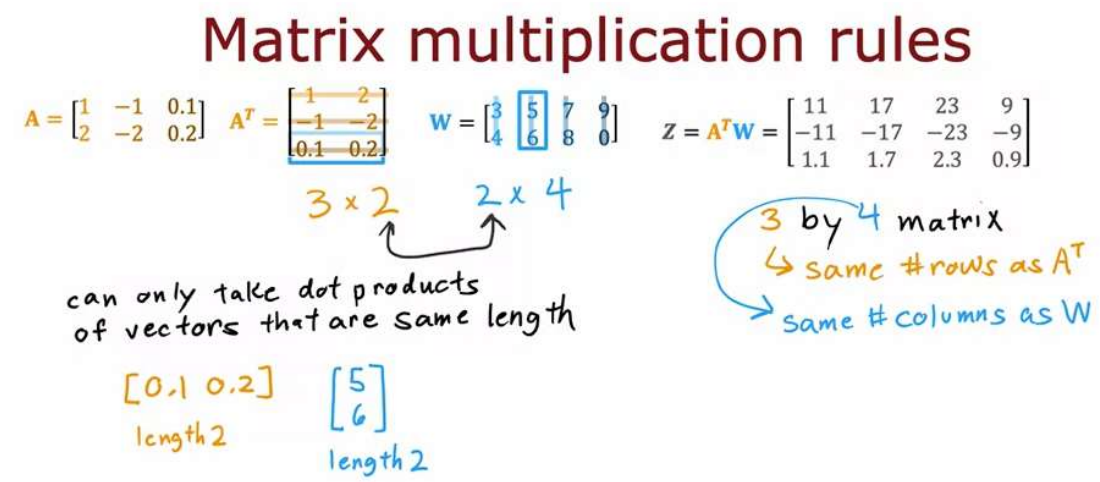
Werfen wir also einen Blick auf die allgemeine Form, wie man zwei Matrizen miteinander multipliziert. Und im letzten Video nach diesem werden wir dies auf die vektorisierte Implementierung eines neuronalen Netzwerks anwenden.  Hier ist die Matrix A, eine 2 x 3-Matrix, da sie zwei Zeilen und drei Spalten hat. Wie zuvor ermutige ich Sie, sich die Spalten dieser Matrix als drei Vektoren vorzustellen, die Vektoren a1, a2 und a3. Und was wir tun werden, ist, A- Transponierte zu nehmen und diese mit der Matrix W zu multiplizieren. Erstens: Was ist A- Transponierte? Nun, eine Transponierung erhält man, indem man die erste Spalte von A nimmt und sie wie folgt auf die Seite legt und dann die zweite Spalte von A nimmt und wie folgt auf die Seite legt. Und dann die dritte Spalte von A und so auf die Seite legen.



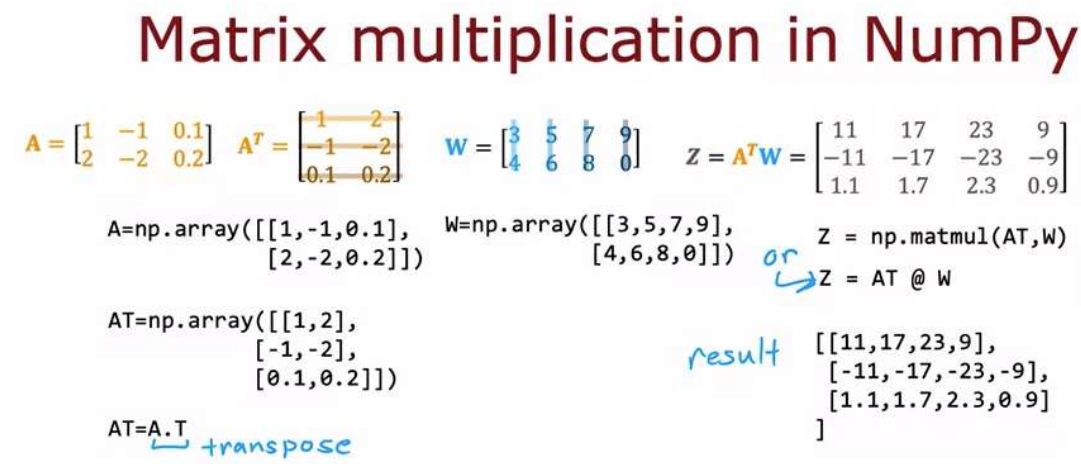

Und so sind diese Zeilen jetzt A1-Transponierung, A2-Transposition und A3-Transposition. Als nächstes ist hier die Matrix W. Ich empfehle Ihnen, sich W als gestapelte Faktoren w1, w2, w3 und w4 vorzustellen. Schauen wir uns also an, wie Sie dann die A- Transponierung mal W berechnen. Beachten Sie nun, dass ich auch leicht unterschiedliche Orangetöne verwendet habe, um die verschiedenen Spalten von A zu kennzeichnen, wobei der gleiche Farbton den Zahlen entspricht, die wir als gruppiert betrachten in einen Vektor. Und derselbe Farbton wird verwendet, um unterschiedliche Rollen der A- Transponierung anzuzeigen, da die verschiedenen Rollen der A-Transponierung A1-Transposition, A2-Transposition und A3-Transposition sind. Und auf ähnliche Weise habe ich unterschiedliche Farbtöne verwendet, um die verschiedenen Spalten von W zu

kennzeichnen. Da die Zahlen den gleichen Blauton haben, werden sie zu den Vektoren w1, w 2 oder w3 oder w4 gruppiert . Schauen wir uns nun an, wie Sie A- Transponierung mal W berechnen können. Ich werde vertikale Bögen zu den verschiedenen Blautönen und horizontale Balken mit den verschiedenen Orangetönen zeichnen, um anzuzeigen, welche Elemente von Z, also A-Transponierung W, beeinflusst werden oder von den verschiedenen Rollen der A-Transponierung beeinflusst werden und die von den verschiedenen Spalten von W beeinflusst oder beeinflusst werden. Schauen wir uns also zum Beispiel die erste Spalte von W an. Das ist also w1, wie hier durch den hellsten Blauton angezeigt. W1 wird also die hier gezeigte erste Spalte von Z durch diesen helleren Blauton beeinflussen oder ihr entsprechen. Und die Werte dieser zweiten Spalte von W, die w2 ist, wie durch diesen zweiten helleren Blauton angezeigt, wirken sich auf die in der zweiten Spalte von Z berechneten Werte aus und so weiter für die dritte und vierte Spalte. Schauen wir uns dementsprechend die A- Transponierung an. A1-Transposition ist die erste Reihe von A-Transposition, wie durch den hellsten Orangeton angezeigt, und A1-Transposition wirkt sich auf die Werte in der ersten Reihe von Z aus oder beeinflusst sie oder entspricht ihnen. Und A2-Transposition beeinflusst die zweite Reihe von Z und A3-Transporte Diese dritte Zeile von Z beeinflussen oder entsprechen ihr. Lassen Sie uns also herausfinden, wie die Matrix Z berechnet wird, die eine 3 x 4-Matrix sein wird. Also mit insgesamt 12 Zahlen. Beginnen wir damit, herauszufinden, wie die Zahl in der ersten Zeile, in der ersten Spalte von Z berechnet wird. Also dieses Element ganz oben links, da dies die erste Zeile und erste Spalte ist, die dem helleren Orangeton und dem helleren Farbton entsprechen von Blau. Die Art und Weise, wie Sie das berechnen, besteht darin, die erste Zeile einer Transponierten und die erste Spalte von W zu nehmen und deren inneres Produkt oder das Produkt zu nehmen. Diese Zahl ist also das Skalarprodukt von ( 1,2 ) mit (3,4), also (1 * 3) + (2 * 4) = 11. Schauen wir uns das zweite Beispiel an. Wie würden Sie dieses Element von Z berechnen? Das ist also in der dritten Zeile, Zeile 1, Zeile 2, Zeile 3. Das ist also in Zeile 3 und der zweiten Spalte, Spalte 1, Spalte 2. Also um die Zahl in Zeile zu berechnen 3, Spalte 2 von Z, würden Sie nun Zeile 3 von A transponieren und Spalte 2 von W nehmen und diese miteinander punktieren. Beachten Sie, dass dies dem dunkelsten Orangeton und dem zweithellsten Blauton entspricht. Und um dies zu berechnen, ist dies (0,1 * 5) + ( 0,2 * 6), also (0,5 + 1,2), was gleich 1,7 ist. Um also die Zahl in Zeile 3, Spalte 2 von Z zu berechnen, nehmen Sie die dritte Zeile, Zeile 3 einer Transponierten und Spalte 2 von W. Schauen wir uns noch ein weiteres Beispiel an und schauen wir, ob Sie dieses herausfinden können. Das ist Zeile 2, Spalte 3 der Matrix Z. Schauen Sie doch einmal nach, ob Sie herausfinden können, in welcher Zeile und in welcher Spalte das Skalarprodukt zusammengefasst werden soll und welche Zahl daher in dieses Element passt dieser Matrix. Hoffentlich hast du das verstanden. Sie sollten sich Zeile 2 von A-Transponierung und Spalte 3 von W schnappen. Und wenn Sie das Punktprodukt daraus bilden, erhalten Sie A2-Transponierung w3 ist (-1 * 7) + (-2 * 8 ) , was (-7 + -16 ist ), was gleich -23 ist. Und so berechnen Sie dieses Element der Matrix Z. Und wenn Sie dies für jedes Element der Matrix Z tun, können Sie alle Zahlen in dieser Matrix berechnen, was dann so aussieht. Wenn Sie möchten, können Sie das Video jederzeit anhalten, die Elemente auswählen und noch einmal überprüfen, ob die Formel, die wir durchgegangen sind, Ihnen den richtigen Wert für Z liefert. Ich möchte nur auf eine letzte interessante Voraussetzung für die Multiplikation von Matrizen hinweisen, nämlich die X-Transponierung ist hier eine 3 x 2-Matrix, weil sie 3 Zeilen und 2 Spalten hat, und W ist hier eine 2 x 4-Matrix, weil sie 2 Zeilen und 4 Spalten hat. Eine Voraussetzung für die Multiplikation zweier Matrizen ist, dass diese Zahl mit dieser Zahl übereinstimmt. Und das liegt daran, dass Skalarprodukte nur zwischen Vektoren gleicher Länge gebildet werden können. Sie können also das Skalarprodukt zwischen einem Vektor mit zwei Zahlen bilden. Und das liegt daran, dass Sie das Innenprodukt zwischen dem Vektor der Länge 2 nur mit einem anderen Vektor der Länge 2 bilden können . Sie können beispielsweise nicht das Innenprodukt zwischen dem Vektor der Länge 2 und einem Vektor der Länge 3 bilden. Und deshalb ist die Matrixmultiplikation nur dann gültig, wenn die Anzahl der Spalten der ersten Matrix, das heißt hier die A- Transponierte, gleich der

Anzahl der Rollen der zweiten Matrix ist, das ist hier die Anzahl der Rollen von W. Wenn Sie also in diesem Prozess Skalarprodukte bilden, nehmen Sie Skalarprodukte von Vektoren gleicher Größe. Und dann ist die andere Beobachtung, dass die Ausgabe Z einer Transponierten W entspricht. Die Dimensionen von Z betragen 3 mal 4. Die Ausgabe dieser Multiplikation hat also die gleiche Anzahl von Zeilen wie die X-Transponierte und die gleiche Anzahl von Spalten wie W Und das ist also auch eine weitere Eigenschaft der Matrixmultiplikation. Das ist also Matrixmultiplikation. Alle diese Videos sind optional. Vielen Dank, dass Sie mir dabei zur Seite standen. Und wenn Sie später in dieser Woche Interesse haben, gibt es auch einige rein optionale Tests, mit denen Sie noch mehr dieser Berechnungen selbst üben können. Nehmen wir einiges davon, was wir über die Matrixmultiplikation gelernt haben, und wenden wir es auf die vektorisierte Implementierung eines neuronalen Netzwerks an. Ich muss sagen, als ich die vektorisierte Implementierung zum ersten Mal verstanden habe, fand ich sie wirklich cool.




Ich selbst implementierte neuronale Netze schon seit einiger Zeit ohne die vektorisierte Implementierung. Und als ich die vektorisierte Implementierung endlich verstand und zum ersten Mal auf diese Weise implementierte, lief sie unglaublich viel schneller als alles, was ich jemals zuvor gemacht hatte.

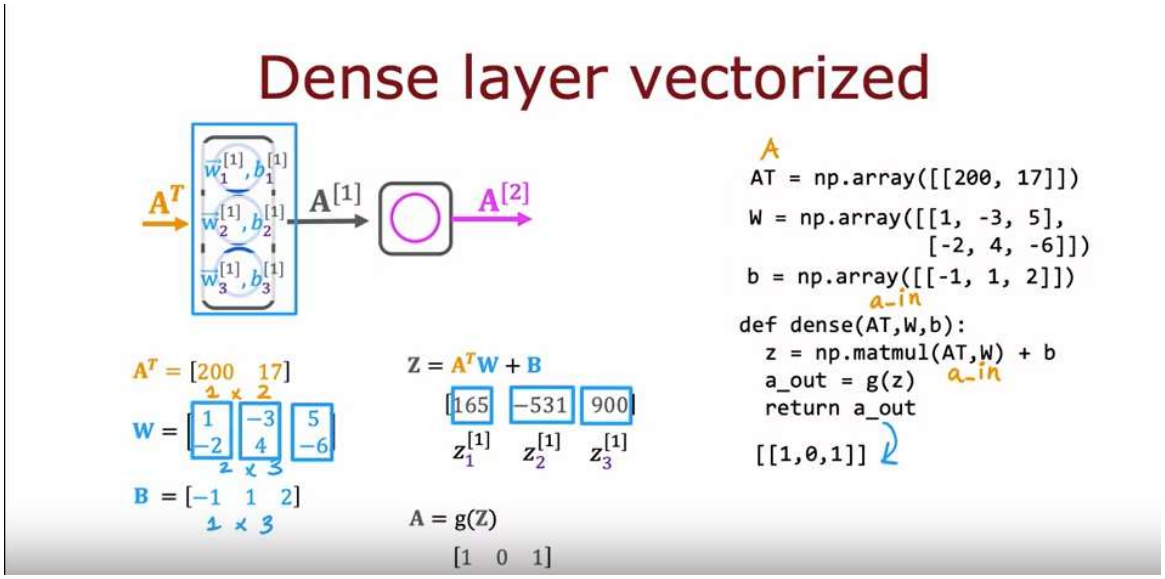


Und ich dachte: Wow, ich wünschte, ich hätte das früher herausgefunden. Die vektorisierte Implementierung ist etwas kompliziert, sorgt aber dafür, dass Ihre Netzwerke viel schneller laufen. Schauen wir uns das im nächsten Video an

## Matrixmultiplikationscode

Lassen Sie uns ohne weitere Umschweife mit der vektorisierten Implementierung eines neuronalen Netzwerks beginnen. Wir schauen uns den Code an, den Sie in einem früheren Video gesehen haben, und hoffen, dass Matmul, also die Matrixmultiplikationsberechnung, mehr Sinn ergibt. Lassen Sie uns einsteigen. Sie haben zuvor gesehen, wie Sie die Matrix A nehmen und die A- Transponierung mal W berechnen können, was hier zu dieser Matrix Z führt.

Wenn dies im Code die Matrix A ist, ist dies ein NumPy-Array mit den Elementen, die dem entsprechen, was ich geschrieben habe oben, dann wird die A- Transponierung, die ich als AT schreiben werde, diese Matrix hier sein, wobei die Spalten von A jetzt stattdessen wieder in Zeilen angeordnet sind. Übrigens, anstatt AT auf diese Weise einzurichten, eine andere Möglichkeit, AT in NumPy zu berechnen, schreiben wir AT gleich AT. Das ist die Transponierungsfunktion, die die Spalten einer Matrix nimmt und sie auf die Seite legt. So initialisieren Sie im Code die Matrix W als ein weiteres 2D-NumPy-Array. Um dann Z gleich A transponiert mal W zu berechnen, schreiben Sie Z gleich np.matmul, AT, W, und das wird diese Matrix Z hier berechnen, was Ihnen dieses Ergebnis hier unten liefert. Übrigens, wenn Sie den Code anderer lesen, sehen Sie manchmal, dass Z gleich AT und dann @ W ist. Dies ist eine alternative Möglichkeit, die Matmal-Funktion aufzurufen. Obwohl ich die Verwendung von np.matmul klarer finde. Bei dem Aufruf, den Sie in dieser Klasse sehen, verwenden wir einfach die Matmal-Funktion wie folgt und nicht dieses @. Schauen wir uns an, wie eine vektorisierte Implementierung von Forward Prop aussieht. Ich werde A- Transponierung so einstellen, dass sie den Eingabemerkmalswerten 217 entspricht. Dies sind nur die üblichen Eingabemerkmalswerte, 200 Grad Rösten von Kaffee für 17 Minuten. Dies ist eine Eins-zu-Zwei-Matrix. Ich nehme die Parameter w_1, w_2 und w_3 und stapele sie in Spalten wie diese, um diese Matrix W zu bilden. Die Werte b_1, b_2, b_3, ich werde sie in eine Eins-mal-Drei-Matrix einfügen, das ist diese Matrix B wie folgt. Dann stellt sich heraus, dass, wenn Sie Z gleich A berechnen und W plus B transponieren, dies zu diesen drei Zahlen führt. Dies wird berechnet, indem die Eingabemerkmalswerte mit der ersten Spalte multipliziert und dann B addiert werden, um 165 zu erhalten. Nehmen Punktproduktion dieser Merkmalswerte mit der zweiten Spalte, also einer Gewichtung w_2, und Addition von b_2, um minus 531 zu erhalten. Diese Merkmalswerte punktproduktiv mit den Gewichtungen w_3 plus b_3, um 900 zu erhalten. Wenn Sie möchten, können Sie das Video jederzeit anhalten Überprüfen Sie diese Berechnungen noch einmal. Aber das ergibt die Werte von z^1_1, Z^1_2 und Z^1_3. Wenn schließlich die Funktion g die Sigmoidfunktion elementweise auf diese drei Zahlen anwendet, d up ist 1 ,0,1 . Es ist 1,0,1, weil das Sigmoid von 165 so nahe bei Eins liegt, dass die numerische Rundung auf Eins basiert und dies die Basen 0 und 1 sind. Sehen wir uns an, wie Sie dies im Code implementieren. Eine Transponierte ist gleich diesem, ist diese Eins-mal-Zwei-Matrix von 217.




Die Matrix W ist diese Zwei-mal-Drei-Matrix und B, dies ist eine Eins-mal-Drei-Matrix. Die Art und Weise, wie Sie Forward Prop in einer Ebene implementieren können, ist eine dichte Eingabe. Eine Transponierung W b ist gleich z gleich matmul A Transponierung mal W plus b. Das implementiert lediglich diese Codezeile. Dann ist a_out, das die Ausgabe dieser Ebene ist, gleich g, der Aktivierungsfunktion, die elementweise auf diese Matrix Z angewendet wird. Sie geben a_out zurück, und das gibt Ihnen diesen Wert. Falls Sie diese Folie mit der Folie vor ein paar Videos vergleichen, gab es nur einen kleinen Unterschied, der konventionsgemäß in der Art und Weise lag, wie dies in TensorFlow implementiert ist. Anstatt diese Variable A,T zu nennen, nannten wir sie A_in , weshalb auch dies die korrekte Implementierung des Codes ist. In TensorFlow gibt es eine Konvention, dass einzelne Beispiele tatsächlich in Zeilen in der Matrix X und nicht in der Matrix X-Transponierung angeordnet sind, weshalb die Code-Implementierung in TensorFlow tatsächlich so aussieht. Dies erklärt jedoch, warum Sie mit nur wenigen Codezeilen Forward Prop im neuronalen Netzwerk implementieren können und darüber hinaus einen enormen Bonus erhalten, da moderne Computer sehr gut darin sind, Matrixmultiplikationen wie Matmul effizient umzusetzen. Das ist das letzte Video dieser Woche. Vielen Dank, dass Sie bis zum Ende dieser optionalen Videos bei mir geblieben sind. Ich hoffe, dass Sie für den Rest dieser Woche auch einen Blick auf die Quizze und die Übungslabore sowie die optionalen Labore werfen, um diesen Stoff noch tiefer zu üben. Sie wissen jetzt, wie man Inferenz und Weiterleitung in einem neuronalen Netzwerk durchführt, was ich wirklich cool finde, also herzlichen Glückwunsch. Nachdem Sie die Tests und Laborübungen durchlaufen haben, kommen Sie bitte auch noch einmal vorbei und in der nächsten Woche schauen wir uns an, wie man ein neuronales Netzwerk tatsächlich trainiert. Ich freue mich darauf, Sie nächste Woche zu sehen.

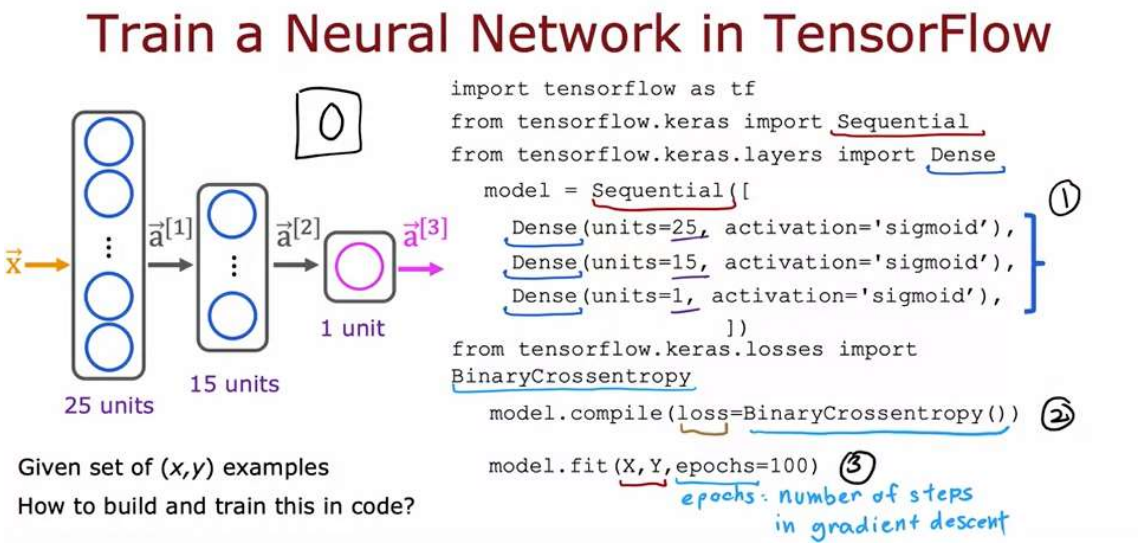
## TensorFlow-Implementierung

This week, you'll learn how to train your model in TensorFlow, and also learn about other important activation functions (besides the sigmoid function), and where to use each type in a neural network. You'll also learn how to go beyond binary classification to multiclass classification (3 or more categories). Multiclass classification will introduce you to a new activation function and a new loss function. Optionally, you can also learn about the difference between multiclass classification and multi-label classification. You'll learn about the Adam optimizer, and why it's an improvement upon regular gradient descent for neural network training. Finally, you will get a brief introduction to other layer types besides the one you've seen thus far.

## Learning Objectives

- Train a neural network on data using TensorFlow
- Understand the difference between various activation functions (sigmoid, ReLU, and linear)
- Understand which activation functions to use for which type of layer
- Understand why we need non-linear activation functions
- Understand multiclass classification
- Calculate the softmax activation for implementing multiclass classification
- Use the categorical cross entropy loss function for multiclass classification
- Use the recommended method for implementing multiclass classification in code
- (Optional): Explain the difference between multi-label and multiclass classification

Willkommen zurück zur zweiten Woche dieses Kurses über fortgeschrittene Lernalgorithmen. Letzte Woche haben Sie gelernt, wie man im neuronalen Netzwerk Inferenzen durchführt. Diese Woche werden wir uns mit dem Training eines neuronalen Netzwerks befassen. Ich denke, dass es wirklich Spaß macht, die eigenen Daten zu nutzen und darauf ein eigenes neuronales Netzwerk zu trainieren. Diese Woche schauen wir uns an, wie Sie das machen können. Tauchen wir ein.

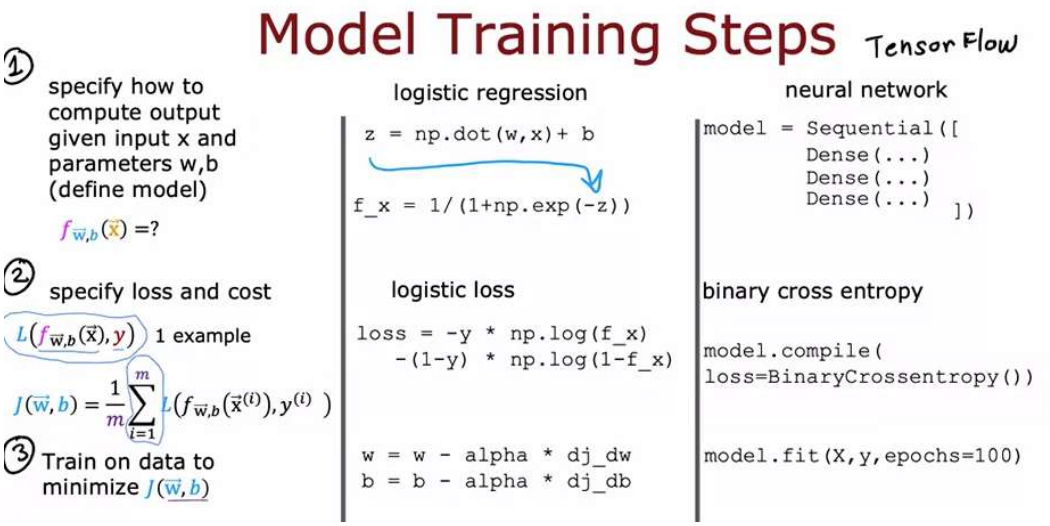



Fahren wir mit unserem laufenden Beispiel der handschriftlichen Ziffernerkennung fort, bei der dieses Bild als Null oder Eins erkannt wird. Hier verwenden wir die neuronale Netzwerkarchitektur, die Sie letzte Woche gesehen haben, bei der Sie ein Eingabe-X haben, also das Bild, und dann war die erste verborgene Schicht 25 Einheiten, die zweite verborgene Schicht 15 Einheiten und dann eine Ausgabeeinheit. Wenn Sie einen Trainingssatz mit Beispielen erhalten würden, der aus Bildern Lassen Sie mich fortfahren und Ihnen den Code zeigen, den Sie in TensorFlow verwenden können, um dieses Netzwerk zu trainieren. In den darauffolgenden Videos gehen wir dann näher auf die Details ein, um zu erklären, was der Code tatsächlich tut. Dies ist ein Code, den Sie schreiben. Dieser erste Teil kommt Ihnen vielleicht aus der vergangenen Woche bekannt vor, als Sie TensorFlow aufforderten, diese drei Schichten eines neuronalen Netzwerks sequentiell aneinanderzureihen. Die erste verborgene Schicht mit 25 Einheiten und Sigmoid-Aktivierung, die zweite verborgene Schicht und schließlich die Ausgabeschicht. Hier gibt es nichts Neues im Vergleich zu dem, was Sie letzte Woche gesehen haben. Im zweiten Schritt bitten Sie TensorFlow, das Modell zu kompilieren. Der wichtigste Schritt bei der Aufforderung an TensorFlow, das Modell zu kompilieren, besteht darin, anzugeben,

welche Verlustfunktion Sie verwenden möchten . In diesem Fall verwenden wir etwas, das der binären Kreuzentropieverlustfunktion ähnelt. Was das wirklich ist, erfahren Sie im nächsten Video. Nachdem Sie dann die Verlustfunktion angegeben haben, besteht der dritte Schritt darin, die Anpassungsfunktion aufzurufen, die TensorFlow anweist, das in Schritt 1 angegebene Modell mithilfe des Verlusts der Kostenfunktion, die Sie in Schritt 2 angegeben haben, an den Datensatz X, Y anzupassen. Als wir im ersten Kurs über den Gradientenabstieg sprachen, mussten wir entscheiden, wie viele Schritte der Gradientenabstieg ausgeführt werden soll oder wie lange der Gradientenabstieg ausgeführt werden soll. Epochen ist also ein Fachbegriff für die Anzahl der Schritte eines Lernalgorithmus wie des Gradientenabstiegs möchte laufen. Das ist es. Schritt 1 besteht darin, das Modell anzugeben, das TensorFlow mitteilt, wie die Inferenz berechnet werden soll. Schritt 2 kompiliert das Modell mithilfe einer bestimmten Verlustfunktion und Schritt 3 besteht darin, das Modell zu trainieren. So können Sie ein neuronales Netzwerk in TensorFlow trainieren. Wie immer hoffe ich, dass Sie diese Codezeilen nicht nur aufrufen, um das Modell zu trainieren, sondern dass Sie auch verstehen, was tatsächlich hinter diesen Codezeilen vor sich geht, sodass Sie sie nicht einfach aufrufen, ohne wirklich zu verstehen, was vor sich geht . Ich denke, das ist wichtig, denn wenn Sie einen Lernalgorithmus ausführen und dieser anfangs nicht funktioniert, hilft Ihnen der konzeptionelle mentale Rahmen dessen, was wirklich vor sich geht, beim Debuggen, wenn die Dinge nicht so funktionieren, wie Sie es erwarten. Fahren wir damit mit dem nächsten Video fort, in dem wir tiefer in die tatsächliche Wirkung dieser Schritte in der TensorFlow-Implementierung eintauchen. Wir sehen uns im nächsten Video.

## Training des Netzwerkes

Werfen wir einen Blick auf die Details dessen, was der TensorFlow-Code zum Training eines neuronalen Netzwerks tatsächlich tut. Bevor wir uns die Details des Trainings im neuronalen Netzwerk ansehen, erinnern wir uns daran, wie Sie im vorherigen Kurs ein logistisches Regressionsmodell trainiert haben.

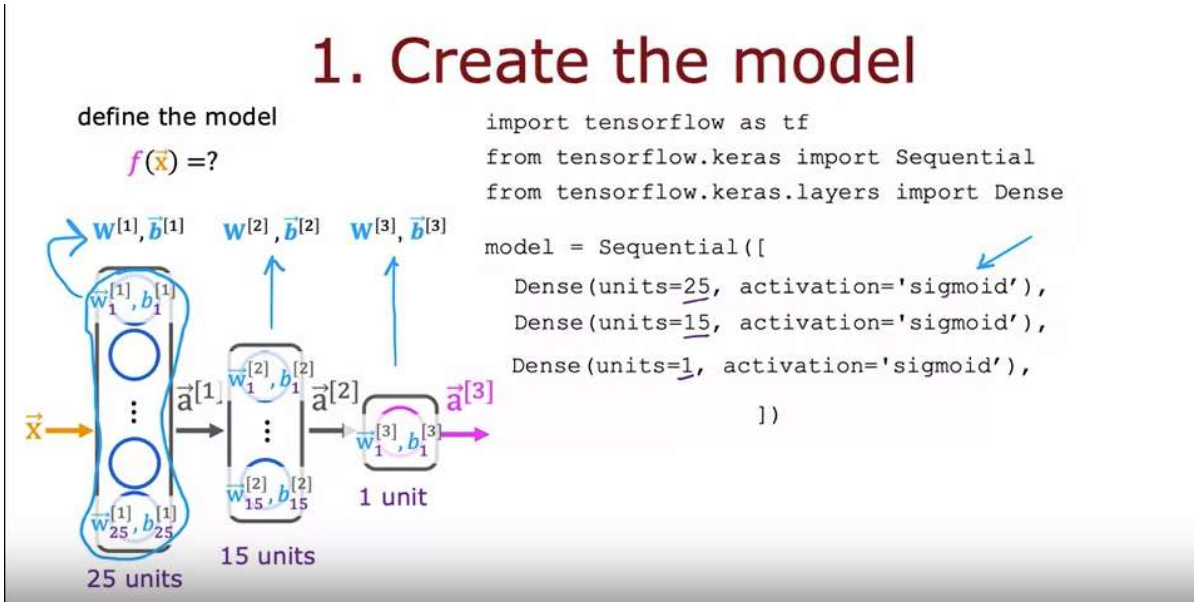



Schritt 1 beim Erstellen eines logistischen Regressionsmodells bestand darin, anzugeben, wie die Ausgabe anhand des Eingabemerkmals x und der Parameter w und b berechnet werden soll. Im ersten Kurs sagten wir, dass die logistische Regressionsfunktion vorhersagt, dass f von x gleich G ist. Die auf W angewendete Sigmoidfunktion X plus B, dann ist F von Modell. Der zweite Schritt, den wir zum Trainieren des Alphabetisierungs-Regressionsmodells durchführen mussten, bestand darin, die Verlustfunktion und auch die Kostenfunktion zu spezifizieren. Sie erinnern sich also vielleicht daran, dass die Verlustfunktion besagte, dass die religiöse Regression opus f von x und der Grundwahrheitsbezeichnung tatsächlich ist Label und ein Trainingssatz y war, dann war der Verlust

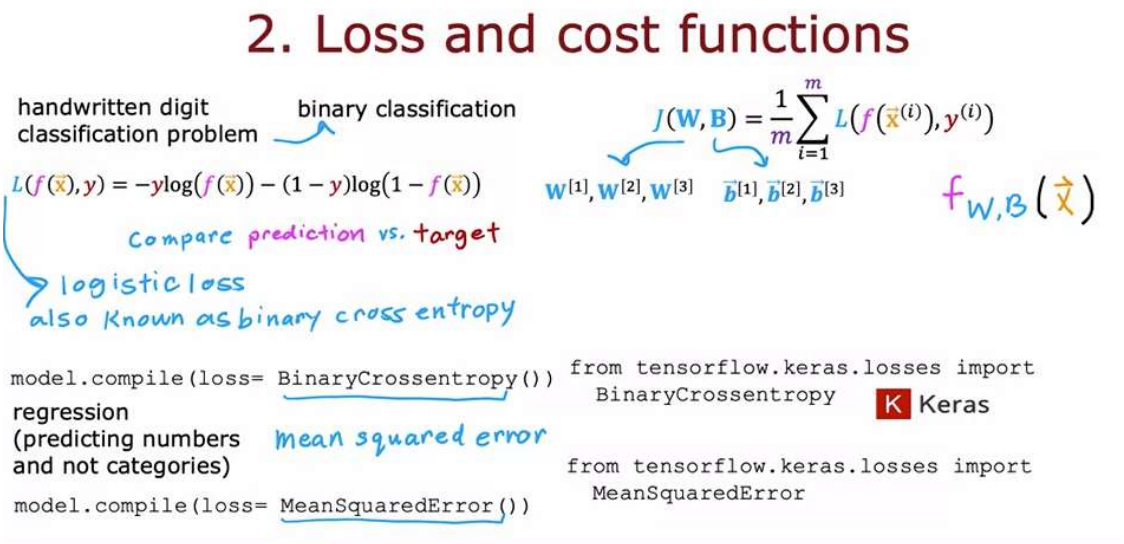
bei diesem einzelnen Trainingsbeispiel negativ y log f von x minus eins minus y mal log von eins minus f von x. Dies war ein Maß dafür, wie gut die logistische Regression bei einem einzelnen Trainingsbeispiel x Komma y funktioniert. Angesichts dieser Definition einer Verlustfunktion definieren wir dann die Kostenfunktion, und die Kostenfunktion war eine Funktion der Parameter W und B, und das war nur der Durchschnitt, der sich aus einem Durchschnitt von insgesamt M Trainingsbeispielen der berechneten Verlustfunktion ergibt die M-Trainingsbeispiele, X1, Y1 bis J ist ein Durchschnitt der über Ihren gesamten Trainingssatz berechneten Verlustfunktion.

Das war der zweite Schritt unserer Vorgehensweise beim Aufbau der logistischen Regression. Dann bestand der dritte und letzte Schritt zum Trainieren eines logistischen Regressionsmodells darin, einen Algorithmus speziell für den Gradientenabstieg zu verwenden, um die Kostenfunktion J von WB als Funktion der Parameter W und B zu minimieren. Wir minimieren die Kosten J als Funktion von Die Parameter verwenden den Gradientenabstieg, wobei W als W minus der Lernrate Alpha mal der Ableitung von J in Bezug auf W aktualisiert wird. Und B wird in ähnlicher Weise als B minus der Lernrate Alpha mal der Ableitung von J in Bezug auf B aktualisiert. Wenn diese drei Schritte. Schritt eins: Spezifizieren, wie die Ausgaben anhand der Eingabe X und der Parameter berechnet werden sollen, Schritt 2: Spezifizieren von Verlusten und Kosten, und Schritt drei: Minimieren der Kostenfunktion, die wir mit der logistischen Regression trainiert haben. Mit denselben drei Schritten können wir ein neuronales Netzwerk in TensorFlow trainieren. Schauen wir uns nun an, wie sich diese drei Schritte auf das Training eines neuronalen Netzwerks auswirken. Wir werden auf den nächsten drei Folien ausführlicher darauf eingehen, aber nur ganz kurz. Schritt eins besteht darin, anzugeben, wie die Ausgabe angesichts der Eingabe x und der Parameter W und B berechnet werden soll. Dies geschieht mit diesem Codeausschnitt, der aus der Spezifikation des neuronalen Netzwerks in der letzten Woche bekannt sein sollte und tatsächlich ausreichte, um die Berechnungen anzugeben, die bei der Vorwärtsausbreitung oder erforderlich sind zum Beispiel für den Inferenzalgorithmus. Der zweite Schritt besteht darin, das Modell zu kompilieren und ihm mitzuteilen, welchen Verlust Sie verwenden möchten. Hier ist der Code, mit dem Sie diese Verlustfunktion angeben, bei der es sich um die binäre Kreuzentropieverlustfunktion handelt. Sobald Sie diesen Verlust angegeben haben, wird ein Mittelwert ermittelt Der gesamte Trainingssatz liefert Ihnen auch die Kostenfunktion für das neuronale Netzwerk. Schritt drei besteht dann darin, die Funktion aufzurufen, um zu versuchen, die Kosten als Funktion der Parameter des neuronalen Netzwerks zu minimieren. Schauen wir uns diese drei Schritte im Zusammenhang mit dem Training eines neuronalen Netzwerks genauer an. Geben Sie im ersten Schritt an, wie die Ausgabe anhand der Eingabe x und der Parameter w und b berechnet werden soll. Dieser Codeausschnitt spezifiziert die

gesamte Architektur des neuronalen Netzwerks.

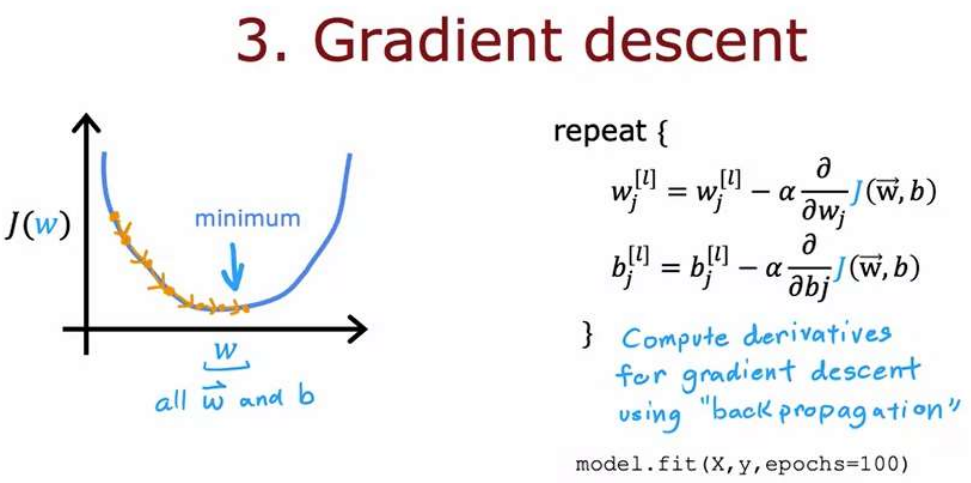



Es sagt Ihnen, dass es 25 versteckte Einheiten in der ersten verborgenen Ebene gibt, dann die 15 in der nächsten und dann eine Ausgabeeinheit und dass wir den Sigmoid-Aktivierungswert verwenden. Basierend auf diesem Codeausschnitt wissen wir auch, was die Parameter w1, v1 sind, obwohl die Parameter der ersten Schicht der zweiten Schicht und die Parameter der dritten Schicht sind. Dieses Codefragment spezifiziert die gesamte Architektur des neuronalen Netzwerks und teilt TensorFlow daher alles mit, was es benötigt. Um die Ausgabe a 3 oder f von x als Funktion der Eingabe x und der Parameter zu berechnen, haben wir hier wl und b l geschrieben. Fahren wir mit Schritt 2 fort. Im zweiten Schritt müssen Sie die Verlustfunktion angeben. Dadurch wird auch die Kostenfunktion definiert, die wir zum Trainieren des neuronalen Netzwerks verwenden. Für das handschriftliche Ziffernklassifizierungsproblem, bei dem Bilder entweder eine Null oder eine Eins haben, was bei weitem am häufigsten vorkommt, ist die zu verwendende Verlustfunktion tatsächlich dieselbe Verlustfunktion wie bei der logistischen Regression: negatives y log f von x minus 1 minus y mal log 1 minus f von x, wobei y die Grundwahrheitsbezeichnung ist, manchmal auch Zielbezeichnung y genannt, und f von x nun die Ausgabe des neuronalen Netzwerks ist. In TensorFlow wird dies als binäre Kreuzentropieverlustfunktion bezeichnet. Woher kommt dieser Name? Nun, in der Statistik stellt sich heraus, dass diese Funktion oben Kreuzentropieverlustfunktion genannt wird. Das ist es also, was Kreuzentropie bedeutet, und das Wort „binär“ unterstreicht nur noch einmal oder weist darauf hin, dass es sich um ein binäres Klassifizierungsproblem handelt, da jedes Bild entweder ein ist Null oder eine Eins. Die Syntax besteht darin, TensorFlow aufzufordern, das neuronale Netzwerk mithilfe dieser Verlustfunktion zu kompilieren. Eine weitere historische Anmerkung: Carers war ursprünglich eine Bibliothek, die unabhängig von TensorFlow entwickelt wurde, und ist eigentlich ein völlig separates Projekt von TensorFlow. Aber irgendwann wurde es in TensorFlow verschmolzen, weshalb wir tf.Keras-Bibliothek.losses im Namen dieser Verlustfunktion haben. Übrigens erinnere ich mich nicht immer an die Namen aller Verlustfunktionen und von TensorFlow, aber ich mache einfach selbst eine kurze Websuche, um den richtigen Namen zu finden, und füge ihn dann in meinen Code ein. Nachdem TensorFlow den Verlust in Bezug auf ein einzelnes Trainingsbeispiel angegeben hat, weiß er, dass es sich bei den Kosten, die Sie minimieren möchten, um den Durchschnitt handelt, indem der Durchschnitt aller m Trainingsbeispiele des Verlusts für alle Trainingsbeispiele herangezogen wird. Die Optimierung dieser Kostenfunktion führt dazu, dass das neuronale Netzwerk an Ihre binären Klassifizierungsdaten angepasst wird.

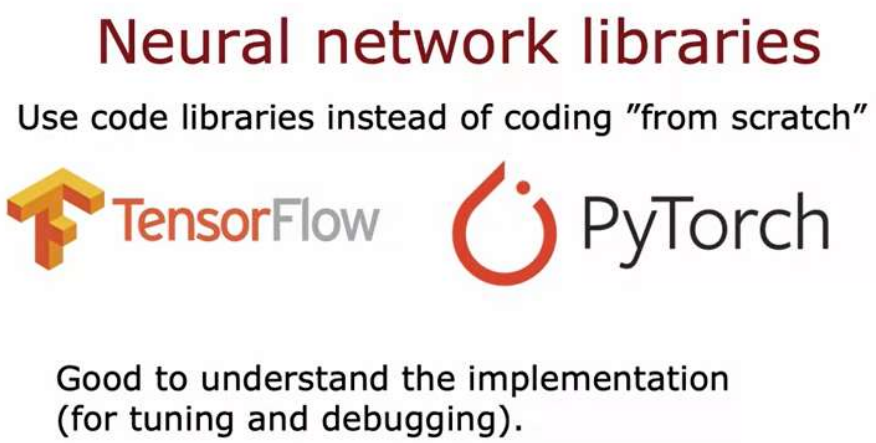



Falls Sie eher ein Regressionsproblem als ein Klassifizierungsproblem lösen möchten. Sie können TensorFlow auch anweisen, Ihr Modell mit einer anderen Verlustfunktion zu kompilieren. Wenn Sie beispielsweise ein Regressionsproblem haben und den quadratischen Fehlerverlust minimieren möchten . Hier ist der quadratische Fehlerverlust. Der Verlust, wenn Ihr Lernalgorithmus f von x mit einer Ziel- oder Grundwahrheitsbezeichnung von y ausgibt, beträgt die Hälfte des quadrierten Fehlers. Dann können Sie diese Verlustfunktion in TensorFlow verwenden, d. h. die vielleicht intuitiver benannte Verlustfunktion für den mittleren quadratischen Fehler. Dann versucht TensorFlow, den mittleren quadratischen Fehler zu minimieren. In diesem Ausdruck verwende ich j mit Großbuchstaben w, Komma und Großbuchstabe b, um die Kostenfunktion zu bezeichnen. Die Kostenfunktion ist eine Funktion aller Parameter im neuronalen Netzwerk. Sie können sich das Großbuchstaben W so vorstellen, dass es W1, W2, W3 umfasst . Alle W-Parameter und das gesamte neue Netzwerk umfassen b1, b2 und b3. Wenn Sie die Kostenfunktion in Bezug auf w und b optimieren, wenn wir versuchen würden, sie in Bezug auf alle Parameter im neuronalen Netzwerk zu optimieren. Oben hatte ich auch f von x als Ausgabe des neuronalen Netzwerks geschrieben, aber wir können auch f von wb schreiben, wenn wir betonen wollen, dass die Ausgabe des neuronalen Netzwerks als Funktion von x von allen Parametern abhängt in allen Schichten des neuronalen Netzwerks. Das ist die Verlustfunktion und die Kostenfunktion. Abschließend bitten Sie TensorFlow, die Kreuzfunktion zu minimieren. Sie erinnern sich vielleicht an den Gradientenabstiegsalgorithmus aus dem ersten Kurs. Wenn Sie den Gradientenabstieg verwenden, um die Parameter eines neuronalen Netzwerks zu trainieren, aktualisieren Sie wlj für jede Schicht l und für jede Einheit j wiederholt entsprechend wlj minus der Lernrate Alpha mal der partiellen Ableitung in Bezug auf diesen Parameter der Kostenfunktion j von wb und analog auch für die Parameter b. Nachdem Sie beispielsweise 100 Iterationen des Gradientenabstiegs durchgeführt haben, erhalten Sie hoffentlich einen guten Wert für die Parameter. Um den Gradientenabstieg nutzen zu können, müssen Sie vor allem diese partiellen Ableitungsterme berechnen. TensorFlow verwendet einen Algorithmus namens Backpropagation, um diese partiellen Ableitungsterme zu berechnen, und das ist in der Tat Standard beim Training neuronaler Netze. TensorFlow kann all diese Dinge für Sie erledigen. Es implementiert die gesamte Rückausbreitung innerhalb dieser Funktion namens fit. Alles, was Sie tun müssen, ist model.fit, x, y als Ihren Trainingssatz aufzurufen und ihm mitzuteilen, dass er dies für 100 Iterationen oder 100 Epochen tun soll. Was Sie später tatsächlich sehen werden, ist, dass TensorFlow einen Algorithmus verwenden kann, der sogar ein wenig schneller ist als der Gradientenabstieg, und Sie werden später in dieser Woche auch mehr darüber erfahren. Jetzt weiß ich, dass wir uns stark auf die TensorFlow-Bibliothek verlassen, um ein neuronales Netzwerk zu implementieren. Ein Muster, das ich bei mehreren Ideen gesehen habe, ist, dass Bibliotheken mit der Weiterentwicklung der

Technologie ausgereifter werden und die meisten Ingenieure Bibliotheken verwenden, anstatt Code von Grund auf zu implementieren. In der Geschichte der Informatik gab es dafür viele weitere Beispiele.



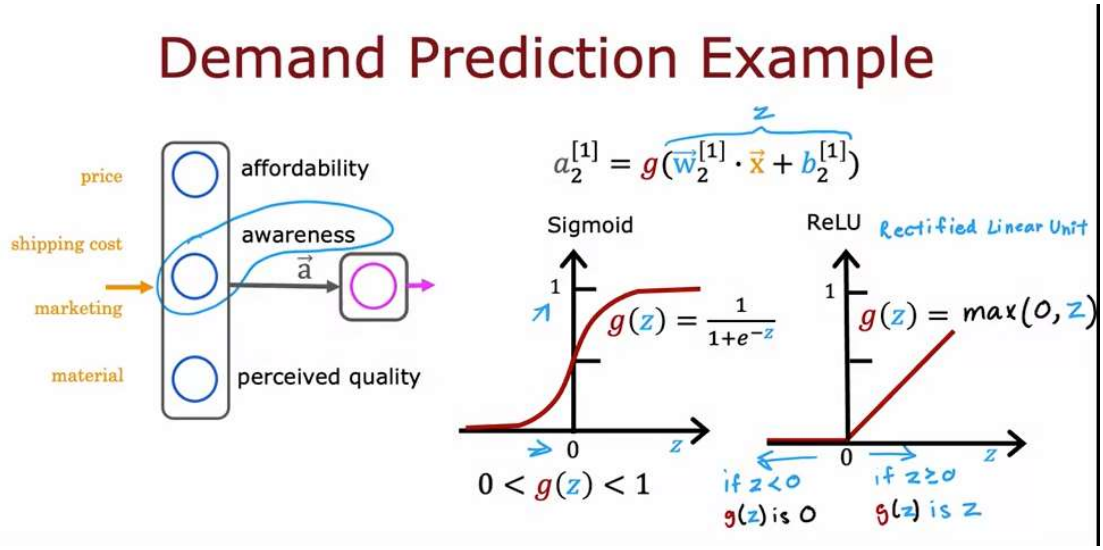

Früher, vor vielen Jahrzehnten, mussten Programmierer ihre eigene Sortierfunktion von Grund auf implementieren, aber jetzt sind Sortierbibliotheken so ausgereift, dass Sie wahrscheinlich die Sortierfunktion von jemand anderem aufrufen, anstatt sie selbst zu implementieren, es sei denn, Sie nehmen an einem Computerkurs teil und ich frage Sie es als Übung zu machen. Wenn Sie heute die Quadratwurzel einer Zahl berechnen möchten, beispielsweise die Quadratwurzel aus sieben, mussten Programmierer früher ihren eigenen Code schreiben, um dies zu berechnen, aber jetzt ruft so ziemlich jeder einfach eine Bibliothek auf, um Quadratwurzeln zu ziehen , oder Matrixoperationen, wie z. B. das Multiplizieren zweier Matrizen miteinander. Als Deep Learning jünger und weniger ausgereift war, implementierten viele Entwickler, darunter auch ich, Dinge von Grund auf mit Python, C++ oder einer anderen Bibliothek. Aber heute sind Deep-Learning-Bibliotheken so weit ausgereift, dass die meisten Entwickler diese Bibliotheken verwenden werden, und tatsächlich verwenden die meisten kommerziellen Implementierungen neuronaler Netze heute eine Bibliothek wie TensorFlow oder PyTorch.



Aber wie ich bereits erwähnt habe, ist es immer noch nützlich zu verstehen, wie sie unter der Haube funktionieren, damit Sie eine bessere Chance haben, zu wissen, wie Sie das Problem beheben können, wenn etwas Unerwartetes passiert, was bei heutigen Bibliotheken immer noch der Fall ist. Nachdem Sie nun wissen, wie Sie ein grundlegendes neuronales Netzwerk, auch mehrschichtiges Perzeptron genannt, trainieren, können Sie einige Dinge am neuronalen Netzwerk ändern, um es noch leistungsfähiger zu machen. Schauen wir uns im nächsten Video an, wie Sie als Alternative zur von uns verwendeten Sigmoid-Aktivierungsfunktion verschiedene Aktivierungsfunktionen einbauen können. Dadurch funktionieren Ihre neuronalen Netze noch viel besser. Schauen wir uns das im nächsten Video an.

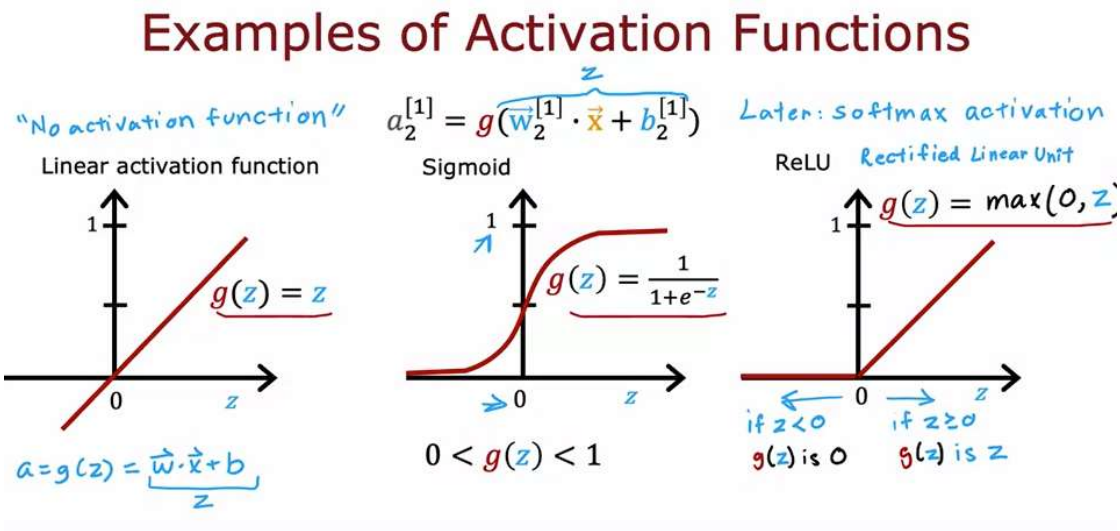
## Alternativen zur Sigmoidaktivierung

Bisher haben wir die Sigmoid-Aktivierungsfunktion in allen Knoten in den verborgenen Schichten und in der Ausgabeschicht verwendet. Und wir haben damit begonnen, weil wir neuronale Netze aufgebaut haben, indem wir die logistische Regression genommen und viele logistische Regressionseinheiten erstellt und diese aneinandergereiht haben.




Wenn Sie jedoch andere Aktivierungsfunktionen verwenden, kann Ihr neuronales Netzwerk viel leistungsfähiger werden. Werfen wir einen Blick darauf, wie das geht. Erinnern Sie sich an das Beispiel einer Nachfragevorhersage von letzter Woche, bei dem Sie anhand von Preis, Versandkosten, Marketing und Material versuchen würden, vorherzusagen, ob etwas sehr erschwinglich ist. Wenn ein guter Bekanntheitsgrad und eine hohe wahrgenommene Qualität vorliegen, versuchen Sie auf dieser Grundlage vorherzusagen, dass es sich um einen Verkaufsschlager handelt. Dies geht jedoch davon aus, dass das Bewusstsein möglicherweise binär ist, d. h. entweder die Menschen sind sich dessen bewusst oder nicht. Es scheint jedoch, dass der Grad, in dem potenzielle Käufer über das von Ihnen verkaufte T-Shirt Bescheid wissen, nicht binär ist. Sie können sich nur ein wenig, einigermaßen bewusst, äußerst bewusst sein oder es könnte völlig viral gegangen sein. Anstatt das Bewusstsein also als binäre Zahl 0, 1 zu modellieren, versuchen Sie, die Wahrscheinlichkeit des Bewusstseins abzuschätzen, oder anstatt das Bewusstsein einfach als eine Zahl zwischen 0 und 1 zu modellieren. Vielleicht sollte das Bewusstsein eine beliebige nicht-negative Zahl sein, weil es jede andere Zahl geben kann negativer Bewusstseinswert, der von 0 bis zu sehr, sehr großen Zahlen reicht. Während wir also zuvor diese Gleichung verwendet hatten, um die Aktivierung dieser zweiten verborgenen Einheit zur Schätzung des Bewusstseins zu berechnen, wobei g die Sigmoidfunktion war und nur zwischen 0 und 1 liegt. Wenn Sie zulassen möchten, dass a , 1 , 2

möglicherweise ein viel größeres positives Ergebnis annehmen Werte können wir stattdessen eine andere Aktivierungsfunktion austauschen. Es stellt sich heraus, dass diese Funktion eine sehr häufige Wahl der Aktivierungsfunktion in neuronalen Netzen ist. Es sieht aus wie das. Es gilt, wenn z dies ist, dann ist g( z) links 0 und dann gibt es diese gerade Linie 45° rechts von 0. Wenn also z größer oder gleich 0 ist, ist g( z) gerade gleich zu z. Das ist die rechte Hälfte dieses Diagramms. Und die mathematische Gleichung dafür lautet: g( z) gleich max(0, z). Fühlen Sie sich frei, selbst zu überprüfen, ob max( 0, z) zu dieser Kurve führt, die ich hier gezeichnet habe. Und wenn a 1, 2 g( z) für diesen Wert von z ist, dann kann der Deaktivierungswert a nicht 0 oder einen anderen nicht negativen Wert annehmen. Diese Aktivierungsfunktion hat einen Namen. Es trägt den Namen ReLU mit dieser komischen Großschreibung und ReLU steht wiederum für einen etwas geheimnisvollen Begriff, aber es steht für gleichgerichtete lineare Einheit.

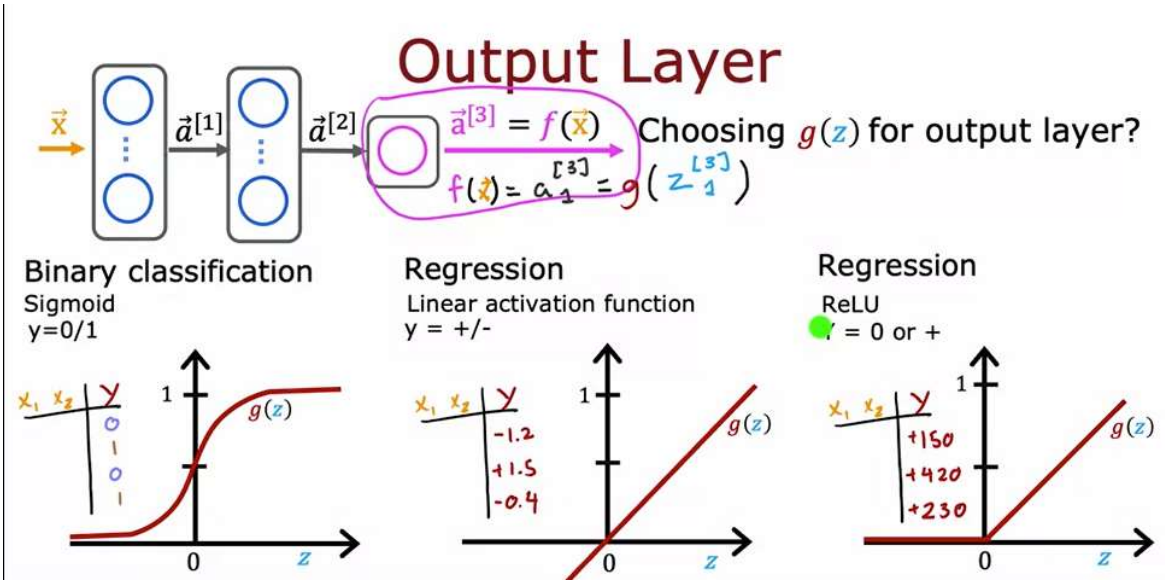



Machen Sie sich keine allzu großen Gedanken darüber, was gleichgerichtet bedeutet und was lineare Einheit bedeutet. Dies war nur der Name, den die Autoren dieser speziellen Aktivierungsfunktion gegeben hatten, als sie sie entwickelten. Aber die meisten Leute im Deep Learning sagen einfach ReLU, um sich auf dieses g( z) zu beziehen. Im Allgemeinen haben Sie die Wahl, was Sie für g( z) verwenden möchten, und manchmal verwenden wir eine andere Wahl als die Sigmoid-Aktivierungsfunktion. Hier sind die am häufigsten verwendeten Aktivierungsfunktionen. Sie haben die Sigmoid-Aktivierungsfunktion gesehen, g( z) entspricht dieser Sigmoid-Funktion. Auf der letzten Folie haben wir uns gerade die ReLU oder die gleichgerichtete lineare Einheit g( z) gleich max(0, z) angesehen. Es gibt noch eine weitere Aktivierungsfunktion, die erwähnenswert ist, die sogenannte lineare Aktivierungsfunktion, bei der einfach g( z) gleich z ist. Wenn Sie die lineare Aktivierungsfunktion verwenden, werden die Leute manchmal sagen, dass wir keine Aktivierungsfunktion verwenden, denn wenn a gleich g( z) ist, wobei g(z) gleich z ist, dann ist a genau gleich diesem wx plus b z. Und so ist es, als wäre überhaupt kein g drin. Wenn Sie also diese lineare Aktivierungsfunktion g( z) verwenden, sagen die Leute manchmal: „Nun, wir verwenden keine Aktivierungsfunktion.“ Obwohl ich mich in diesem Kurs auf die Verwendung der linearen Aktivierungsfunktion und nicht auf die Verwendung keiner Aktivierungsfunktion beziehe. Aber wenn Sie hören, dass jemand anderes diese Terminologie verwendet, ist es das, was er meint. Es bezieht sich lediglich auf die lineare Aktivierungsfunktion. Und diese drei sind wahrscheinlich die mit Abstand am häufigsten verwendeten Aktivierungsfunktionen in neuronalen Netzen. Später in dieser Woche werden wir auf die vierte Funktion namens Softmax-Aktivierungsfunktion eingehen. Aber mit diesen Aktivierungsfunktionen können Sie eine Vielzahl leistungsstarker neuronaler Netze aufbauen. Möchten Sie beim Aufbau eines neuronalen Netzwerks für jedes Neuron die Sigmoid-

Aktivierungsfunktion oder die ReLU-Aktivierungsfunktion verwenden? Oder eine lineare Aktivierungsfunktion? Wie wählen Sie zwischen diesen verschiedenen Aktivierungsfunktionen? Schauen wir uns das im nächsten Video an.

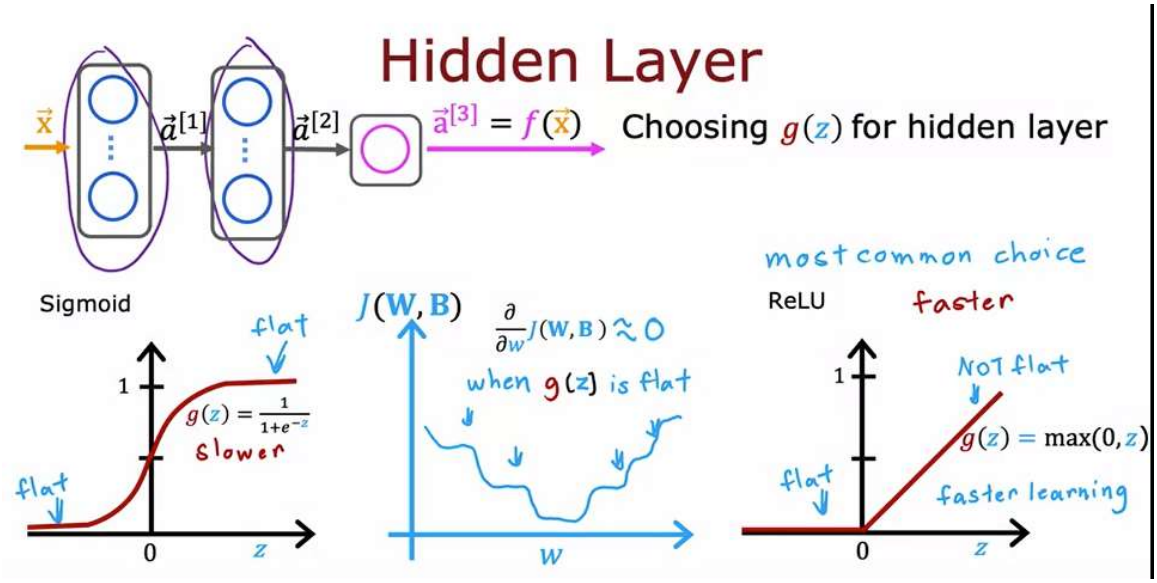
## Aktivierungsfunktionen auswählen

Schauen wir uns an, wie Sie die Aktivierungsfunktion für verschiedene Neuronen in Ihrem neuronalen Netzwerk auswählen können . Wir beginnen mit einigen Anleitungen zur Auswahl für die Ausgabeebene. Es stellt sich heraus, dass es abhängig von der Zielbezeichnung oder der Grundwahrheitsbezeichnung y eine ziemlich natürliche Wahl für die Aktivierungsfunktion für die Ausgabeebene gibt, und wir werden uns dann auch die Wahl der Aktivierungsfunktion ansehen für die verborgenen Schichten Ihres neuronalen Netzwerks. Lass uns einen Blick darauf werfen. Sie können unterschiedliche Aktivierungsfunktionen für verschiedene Neuronen in Ihrem neuronalen Netzwerk auswählen.



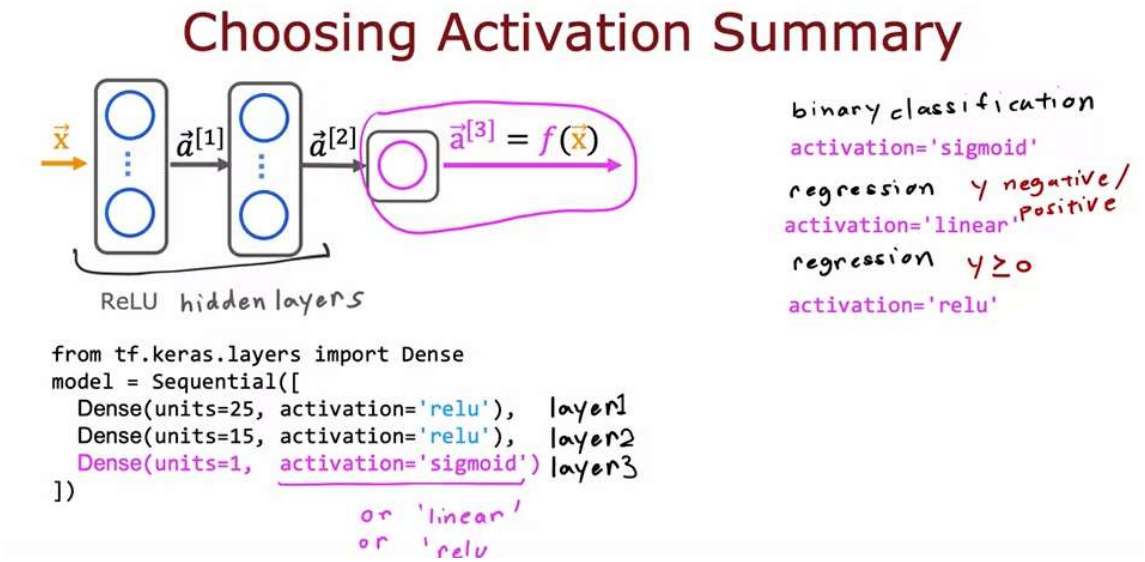

Wenn Sie die Aktivierungsfunktion für die Ausgabeschicht berücksichtigen, stellt sich heraus, dass es oft eine ziemlich natürliche Wahl gibt, je nachdem, was das Ziel oder die Grundwahrheitsbezeichnung ist j. Insbesondere wenn Sie an einem Klassifizierungsproblem arbeiten, bei dem y entweder Null oder Eins ist, also ein binäres Klassifizierungsproblem, dann ist die Sigmoid-Aktivierungsfunktion fast immer die natürlichste Wahl, da das neuronale Netzwerk dann lernt, die Wahrscheinlichkeit vorherzusagen, dass y ist gleich eins, genau wie wir es für die logistische Regression hatten. Meine Empfehlung lautet: Wenn Sie an einem binären Klassifizierungsproblem arbeiten, verwenden Sie Sigmoid auf der Ausgabeebene. Wenn Sie ein Regressionsproblem lösen, können Sie alternativ eine andere Aktivierungsfunktion wählen. Wenn Sie beispielsweise versuchen, vorherzusagen, wie sich der Aktienkurs von morgen im Vergleich zum Aktienkurs von heute verändern wird. Nun, es kann steigen oder fallen, und in diesem Fall wäre y eine Zahl, die entweder positiv oder negativ sein kann, und in diesem Fall würde ich Ihnen empfehlen, die lineare Aktivierungsfunktion zu verwenden. Warum das? Das liegt daran, dass dann die Ausgaben Ihres neuronalen Netzwerks, f von x, das im obigen Beispiel gleich a^3 ist, g wären, angewendet auf z^3, und mit der linearen Aktivierungsfunktion kann g von z beides annehmen positive oder negative Werte. Damit y positiv oder negativ sein kann, verwenden Sie eine lineare Aktivierungsfunktion. Wenn schließlich y nur nicht-negative Werte annehmen kann, z. B. wenn Sie den Preis eines Hauses vorhersagen, dieser niemals negativ sein kann, dann ist die ReLU-Aktivierungsfunktion die

natürlichste Wahl, denn wie Sie hier sehen, ist dies die natürlichste Wahl Die Aktivierungsfunktion nimmt nur nicht negative Werte an, entweder Null oder positive Werte. Bei der Auswahl der Aktivierungsfunktion, die Sie für Ihre Ausgabeebene verwenden möchten, gibt es normalerweise eine ziemlich natürliche Wahl, je nachdem, welches Label Sie vorhersagen möchten. Tatsächlich geht es bei der Anleitung auf dieser Folie darum, wie ich meine Aktivierungsfunktion fast immer auch für die Ausgabeschicht eines neuronalen Netzwerks auswähle. Wie wäre es mit den verborgenen Schichten eines neuronalen Netzwerks?




Es stellt sich heraus, dass die ReLU-Aktivierungsfunktion heute von vielen Praktikern bei weitem die häufigste Wahl für das Training neuronaler Netze ist. Obwohl wir neuronale Netze zunächst mithilfe der Sigmoid-Aktivierungsfunktion beschrieben hatten und in der frühen Geschichte der Entwicklung neuronaler Netze tatsächlich an vielen Stellen Sigmoid-Aktivierungsfunktionen verwendet wurden, hat sich das Gebiet dahingehend weiterentwickelt, dass ReLU viel häufiger und Sigmoids verwendet werden fast nie. Nun, die einzige Ausnahme besteht darin, dass Sie in der Ausgabeebene eine Sigmoid-Aktivierungsfunktion verwenden, wenn Sie ein Problem mit der Binärklassifizierung haben. Warum ist das so? Nun, es gibt ein paar Gründe. Wenn Sie zunächst die ReLU- und Sigmoid-Aktivierungsfunktionen vergleichen, ist die ReLU etwas schneller zu berechnen, da sie lediglich die Berechnung des Maximums von 0, z erfordert, während für das Sigmoid eine Potenzierung und dann eine Umkehrung usw. erforderlich sind, und so ist es etwas weniger effizient. Aber der zweite Grund, der sich als noch wichtiger erweist, ist, dass die ReLU-Funktion nur in einem Teil des Diagramms flach verläuft; Hier auf der linken Seite ist es völlig flach, wohingegen die Sigmoid-Aktivierungsfunktion an zwei Stellen flach ist. Auf der linken Seite des Diagramms wird es flach und auf der rechten Seite des Diagramms wird es flach. Wenn Sie den Gradientenabstieg verwenden, um ein neuronales Netzwerk zu trainieren, wären Gradientenabstiege sehr langsam, wenn Sie eine Funktion haben, die an vielen Stellen fett ist. Ich weiß, dass der Gradientenabstieg die Kostenfunktion J von W, B optimiert und nicht die Aktivierungsfunktion, aber die Aktivierungsfunktion ist ein Teil dessen, was in die Berechnung einfließt, und das führt zu mehr Stellen in der Kostenfunktion J von W, B Auch sie sind flach und weisen ein geringes Gefälle auf, was das Lernen verlangsamt. Ich weiß, dass das nur eine intuitive Erklärung war, aber Forscher haben herausgefunden, dass die Verwendung der ReLU-Aktivierungsfunktion auch dazu führen kann, dass Ihr neuronales Netzwerk etwas schneller lernt, weshalb dies für die meisten Praktiker der Fall ist, wenn Sie versuchen zu entscheiden, welche Aktivierungsfunktion sie haben soll Bei Verwendung mit versteckter Ebene ist die ReLU-Aktivierungsfunktion mittlerweile die mit Abstand häufigste Wahl. Da ich tatsächlich ein neuronales Netzwerk aufbaue, wähle ich auf diese Weise auch Aktivierungsfunktionen für die verborgenen

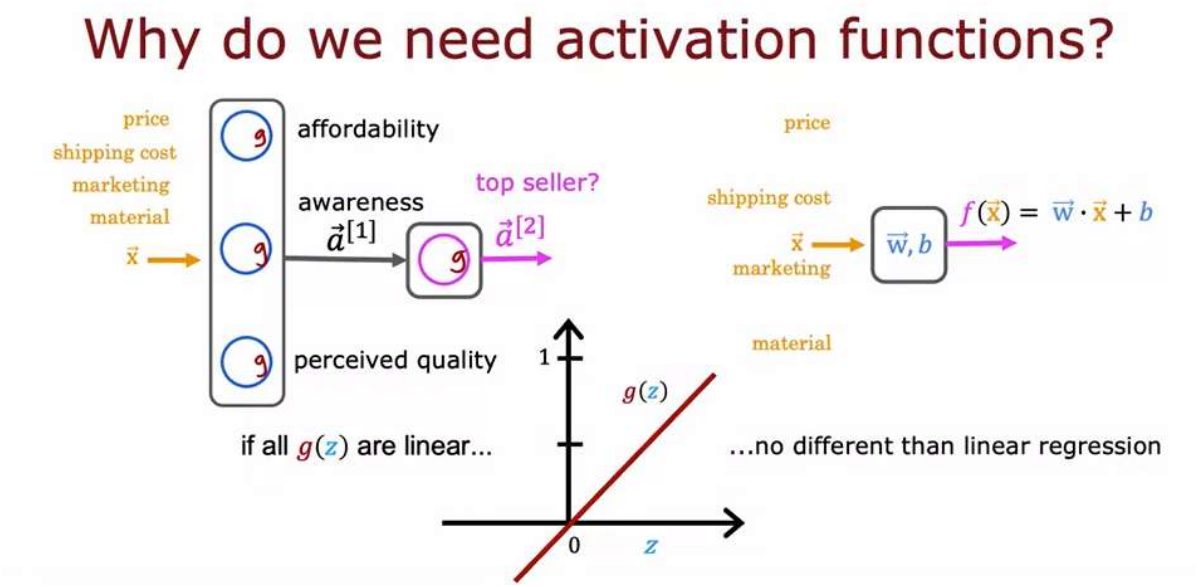
Schichten aus. Zusammenfassend empfehle ich Folgendes, wie Sie die Aktivierungsfunktionen für Ihr neuronales Netzwerk auswählen.



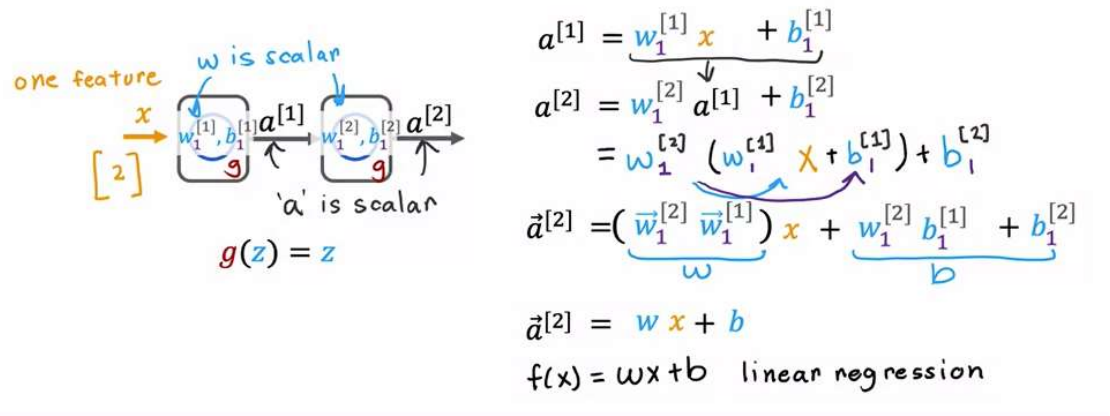

Verwenden Sie für die Ausgabeebene ein Sigmoid, wenn Sie ein binäres Klassifizierungsproblem haben. linear, wenn y eine Zahl ist, die positive oder negative Werte annehmen kann, oder verwenden Sie ReLU, wenn y nur positive Werte oder null positive Werte oder nicht negative Werte annehmen kann. Dann würde ich für die versteckten Ebenen empfehlen, einfach ReLU als Standardaktivierungsfunktion zu verwenden, und in TensorFlow würden Sie es so implementieren. Anstatt zu sagen, dass die Aktivierung gleich Sigmoid ist, wie wir es zuvor getan haben, habe ich für die verborgenen Schichten die erste verborgene Schicht, die zweite verborgene Schicht als TensorFlow zur Verwendung der ReLU-Aktivierungsfunktion und dann für die Ausgabeschicht in diesem Beispiel gefragt um die Sigmoid-Aktivierungsfunktion zu verwenden, aber wenn Sie die lineare Aktivierungsfunktion verwenden möchten, ist das die Syntax dafür, oder wenn Sie die ReLU-Aktivierungsfunktion verwenden möchten, die die Syntax dafür zeigt. Mit diesem umfangreicheren Satz an Aktivierungsfunktionen sind Sie gut aufgestellt, um viel leistungsfähigere neuronale Netze aufzubauen, als wenn Sie nur einmal nur die Sigmoid-Aktivierungsfunktion verwenden. Übrigens, wenn man sich die Forschungsliteratur anschaut, hört man manchmal von Autoren, die sogar andere Aktivierungsfunktionen verwenden, wie zum Beispiel die tan h-Aktivierungsfunktion oder die LeakyReLU-Aktivierungsfunktion oder die swish-Aktivierungsfunktion. Alle paar Jahre erfinden Forscher manchmal eine weitere interessante Aktivierungsfunktion, manchmal funktioniert sie sogar etwas besser. Ich habe zum Beispiel die LeakyReLU-Aktivierungsfunktion ein paar Mal in meiner Arbeit verwendet, und manchmal funktioniert sie etwas besser als die ReLU-Aktivierungsfunktion, die Sie in diesem Video kennengelernt haben. Aber ich denke, dass das, was Sie in diesem Video gelernt haben, für den größten Teil und für die überwiegende Mehrheit der Anwendungen gut genug wäre. Wenn Sie mehr über andere Aktivierungsfunktionen erfahren möchten, schauen Sie sich natürlich gerne im Internet um, und es gibt nur eine Handvoll Fälle, in denen diese anderen Aktivierungsfunktionen ebenfalls noch leistungsfähiger sein könnten. In diesem Sinne wünsche ich Ihnen auch viel Spaß beim Üben dieser Ideen, dieser Aktivierungsfunktionen in den Wahllaboren und in den Übungslaboren. Aber das wirft noch eine weitere Frage auf. Warum brauchen wir überhaupt Aktivierungsfunktionen? Warum verwenden wir nicht einfach die lineare Aktivierungsfunktion oder verwenden nirgendwo eine Aktivierungsfunktion? Es stellt sich heraus, dass dies überhaupt nicht funktioniert. Schauen wir uns im nächsten Video an, warum das so ist und warum Aktivierungsfunktionen so wichtig sind, damit Ihre neuronalen Netze funktionieren.

## Warum brauchen wir Aktivierungsfunktionen?

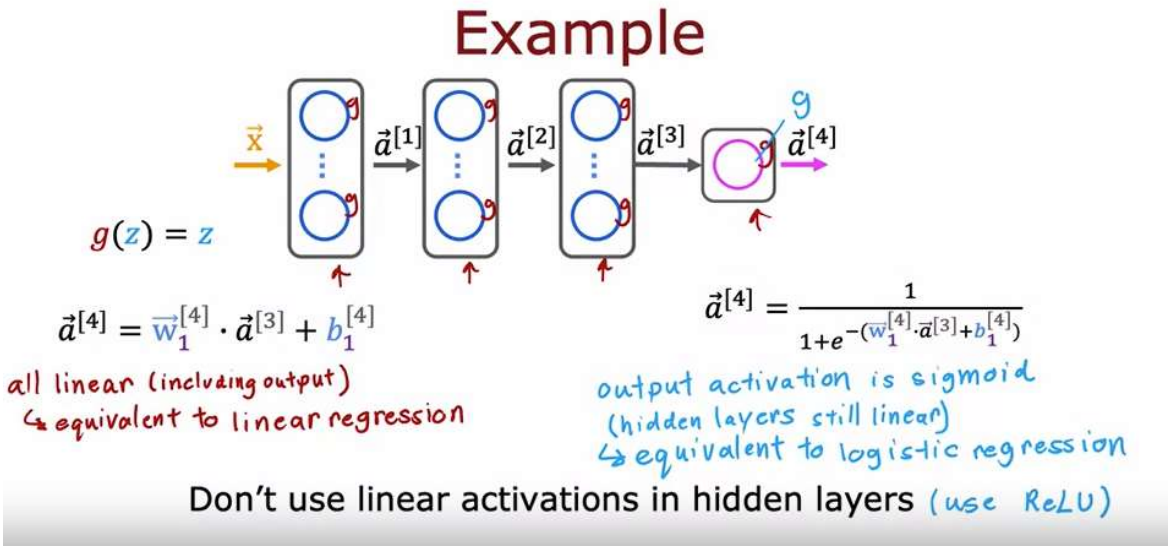
Werfen wir einen Blick darauf, warum neuronale Netze Aktivierungsfunktionen benötigen und warum sie einfach nicht funktionieren, wenn wir die lineare Aktivierungsfunktion in jedem Neuron im neuronalen Netz verwenden würden. Erinnern Sie sich an dieses Beispiel einer Nachfragevorhersage. Was würde passieren, wenn wir für alle Knoten in diesem neuronalen Netzwerk eine lineare Aktivierungsfunktion verwenden würden?




Es stellt sich heraus, dass dieses große neuronale Netzwerk nichts anderes als nur eine lineare Regression sein wird. Dies würde also den gesamten Zweck der Verwendung eines neuronalen Netzwerks zunichte machen, da es dann einfach nicht in der Lage wäre, etwas Komplexeres als das lineare Regressionsmodell, das wir im ersten Kurs kennengelernt haben, anzupassen. Lassen Sie uns dies anhand eines einfacheren Beispiels veranschaulichen. Schauen wir uns das Beispiel eines neuronalen Netzwerks an, in dem die Eingabe x nur eine Zahl ist und wir eine versteckte Einheit mit den Parametern w1 und b1 haben, die a1 ausgibt, was hier nur eine Zahl ist, und dann ist die zweite Schicht die Ausgabeschicht und Es hat auch nur eine Ausgabeeinheit mit den Parametern w2 und b2 und dann die Ausgabe a2, die ebenfalls nur eine Zahl, nur ein Skalar ist, die Ausgabe des neuronalen Netzwerks f von x. Sehen wir uns an, was dieses neuronale Netzwerk tun würde, wenn wir überall die lineare Aktivierungsfunktion g von z gleich z verwenden würden. Um also a1 als Funktion von x zu berechnen, verwendet das neuronale Netzwerk a1 gleich g von w1 mal x plus b1. Aber g von z ist gleich z. Das ist also nur w1 mal x plus b1. Dann ist a2 gleich w2 mal a1 plus b2, weil g von z gleich z ist. Lassen Sie mich diesen Ausdruck für a1 nehmen und ihn dort einsetzen. Das ergibt also w2 mal w1 x plus b1 plus b2. Wenn wir vereinfachen, wird dies zu w2, w1 mal x plus w2, b1 plus b2. Es stellt sich heraus, dass, wenn ich w gleich w2 mal w1 setze und b gleich dieser Größe hier setze, wir gerade gezeigt haben, dass a2 gleich wx plus b ist.




Also ist a2 nur eine lineare Funktion der Eingabe x. Anstatt ein neuronales Netzwerk mit einer verborgenen Schicht und einer Ausgabeschicht zu verwenden, hätten wir genauso gut einfach ein lineares Regressionsmodell verwenden können. Wenn Sie mit linearer Algebra vertraut sind, ergibt sich dieses Ergebnis aus der Tatsache, dass eine lineare Funktion einer linearen Funktion selbst eine lineare Funktion ist. Aus diesem Grund kann ein neuronales Netzwerk aufgrund mehrerer Schichten keine komplexeren Merkmale berechnen oder etwas Komplexeres als nur eine lineare Funktion lernen. Wenn Sie also im Allgemeinen ein neuronales Netzwerk mit mehreren Schichten wie diesem hätten und sagen würden, Sie würden eine lineare Aktivierungsfunktion für alle verborgenen Schichten und auch eine lineare Aktivierungsfunktion für die Ausgabeschicht verwenden, dann ergibt sich Folgendes Das Modell berechnet eine Ausgabe, die vollständig der linearen Regression entspricht. Die Ausgabe a4 kann als lineare Funktion der Eingabemerkmale x plus b ausgedrückt werden. Oder alternativ: Wenn wir immer noch eine lineare Aktivierungsfunktion für alle verborgenen Schichten verwenden würden, für diese drei verborgenen Schichten hier, aber wir würden eine logistische Aktivierungsfunktion für die Ausgabeschicht verwenden, dann stellt sich heraus, dass Sie dieses Modell zeigen können entspricht der logistischen Regression, und a4 kann in diesem Fall als 1 über 1 plus e zum negativen wx plus b für einige Werte von w und b ausgedrückt werden. Dieses große neuronale Netzwerk macht also nichts, was man nicht auch mit logistischer Regression machen kann.

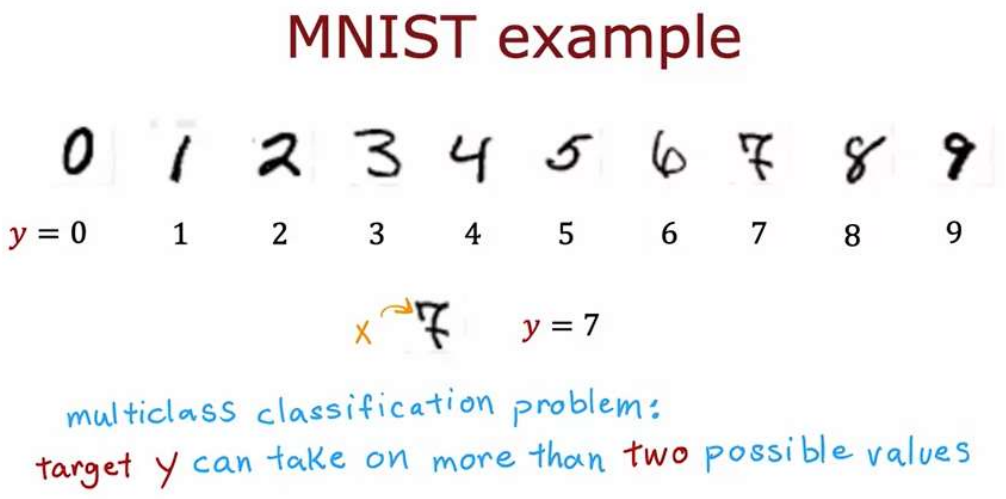


Aus diesem Grund lautet eine allgemeine Faustregel, die lineare Aktivierungsfunktion nicht in den verborgenen Schichten des neuronalen Netzwerks zu verwenden. Tatsächlich empfehle ich, dass die Verwendung der ReLU-Aktivierungsfunktion normalerweise völlig ausreichen sollte. Aus diesem Grund benötigt ein neuronales Netzwerk überall andere Aktivierungsfunktionen als nur die lineare Aktivierungsfunktion. Bisher haben Sie gelernt, neuronale Netze für binäre Klassifizierungsprobleme aufzubauen, bei denen y entweder Null oder Eins ist. Sowie für Regressionsprobleme, bei denen y negative oder positive Werte annehmen kann, oder vielleicht auch nur positive und nicht negative Werte. Im nächsten Video möchte ich mit Ihnen eine Verallgemeinerung dessen teilen, was Sie bisher zur Klassifizierung gesehen haben. Insbesondere dann, wenn y nicht nur zwei Werte annimmt, sondern drei oder vier oder zehn oder sogar mehr kategoriale Werte annehmen kann.

Schauen wir uns an, wie Sie ein neuronales Netzwerk für diese Art von Klassifizierungsproblem aufbauen können.

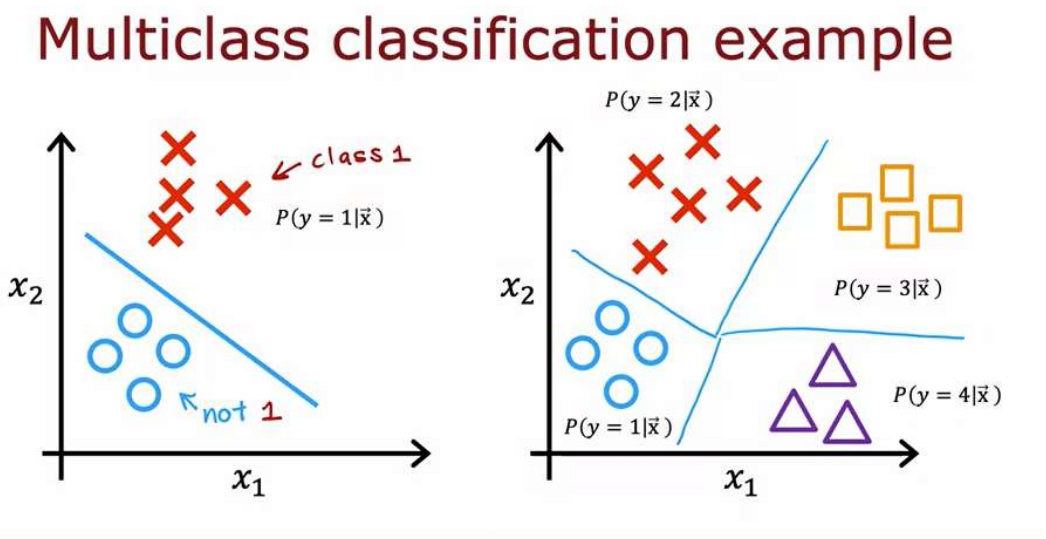
## Mehrklassig

zu Klassifizierungsproblemen, bei denen es mehr als nur zwei mögliche Ausgabebezeichnungen geben kann, also nicht nur Null oder 1. Schauen wir uns an, was das bedeutet. Bei den handschriftlichen Ziffernklassifizierungsproblemen , die wir uns bisher angesehen haben, haben wir lediglich versucht, zwischen den handschriftlichen Ziffern 0 und 1 zu unterscheiden. Aber wenn Sie versuchen, Protokolle oder Postleitzahlen in einem Umschlag zu lesen, sind es tatsächlich 10 mögliche Ziffern, die Sie vielleicht erkennen möchten.




Oder alternativ haben Sie im ersten Kurs das Beispiel gesehen, wenn Sie versuchen zu klassifizieren, ob Patienten möglicherweise eine von drei oder fünf verschiedenen möglichen Krankheiten haben. Auch das wäre ein Multiklassen-Klassifizierungsproblem , oder eine Sache, an der ich viel gearbeitet habe, ist die visuelle Fehlerinspektion des Teileherstellers im Werk. Wenn Sie sich das Bild einer Pille ansehen, die ein Pharmaunternehmen hergestellt hat, und versuchen herauszufinden, ob sie einen Kratzeffekt, Verfärbungsfehler oder einen Chip-Defekt aufweist. Und das wären wieder mehrere Klassen verschiedener Arten von Mängeln, die man dieser Pille zuordnen könnte. Ein Klassifizierungsproblem mit mehreren Klassen ist also immer noch ein Klassifizierungsproblem, da y nur eine kleine Anzahl diskreter Kategorien annehmen kann, keine Zahl, aber jetzt kann y mehr als nur zwei mögliche Werte annehmen. Während Sie also zuvor die Klassifizierung gekauft haben , verfügten Sie möglicherweise über einen Datensatz wie diesen mit den Merkmalen x1 und x2. In

diesem Fall würde die logistische Regression das Modell anpassen, um die Wahrscheinlichkeit abzuschätzen, dass y angesichts der Merkmale x 1 ist. Da y bei Mehrklassenklassifizierungsproblemen entweder 01 war, hätten Sie stattdessen einen Datensatz, der möglicherweise so aussieht.

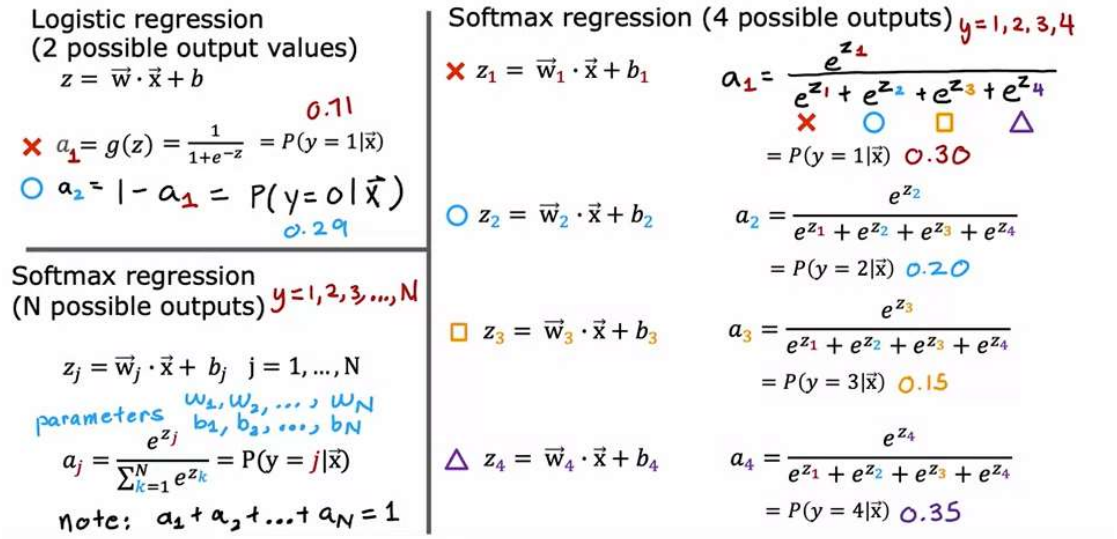



Wo wir vier Klassen haben, wobei das Os eine Klasse und das x eine andere Klasse darstellt. Die Dreiecke repräsentieren die dritte Klasse und die Quadrate repräsentieren die vierte Klasse. Und anstatt nur die Wahrscheinlichkeit abzuschätzen, dass y gleich 1 ist, wollen wir jetzt abschätzen, wie hoch die Wahrscheinlichkeit ist, dass y gleich 1 ist, oder wie hoch die Wahrscheinlichkeit ist, dass y gleich 2 ist? Oder wie groß ist die Chance, dass y gleich 3 ist, oder wie groß ist die Chance, dass y gleich 4 ist? Und es stellt sich heraus, dass der Algorithmus, den Sie im nächsten Video kennengelernt haben, eine Entscheidungsgrenze lernen kann, die vielleicht so aussieht und den explodierten Raum in vier Kategorien unterteilt, anstatt nur in zwei Kategorien. Das ist also die Definition des Mehrklassenklassifizierungsproblems. Im nächsten Video schauen wir uns das an

## Softmax

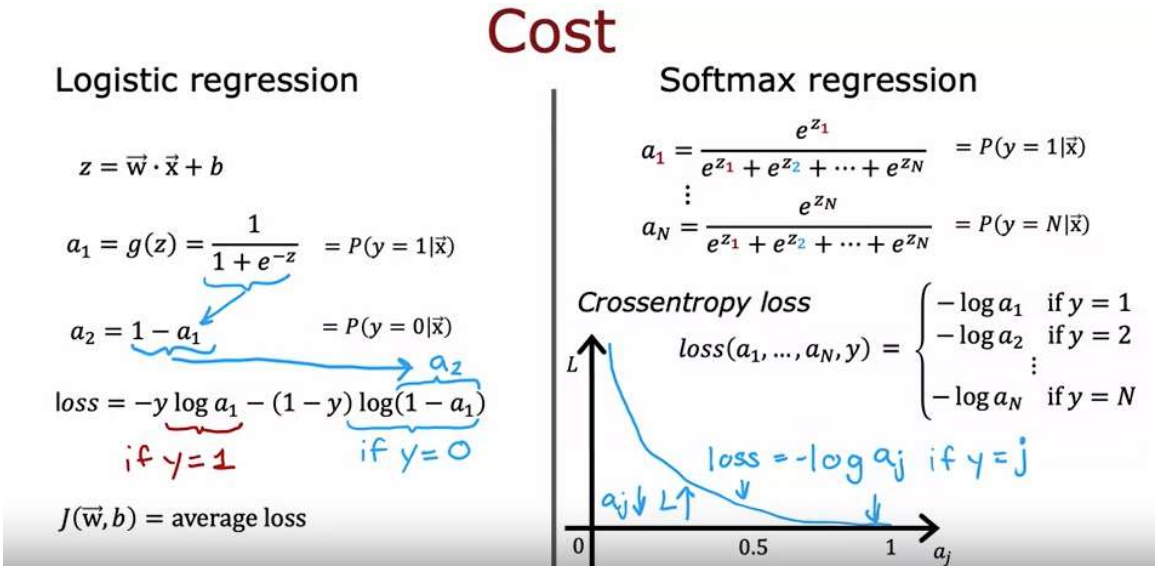
Der Softmax-Regressionsalgorithmus ist eine Verallgemeinerung des logistischen Regressionsalgorithmus und mit diesem können Sie Klassifizierungsprobleme mit mehreren Klassen lösen. Anschließend nehmen wir die Softmax- Regression und passen sie in ein neues neuronales Netzwerk ein, sodass Sie auch ein neuronales Netzwerk trainieren können, um Klassifizierungsprobleme mit mehreren Klassen durchzuführen. Kommen wir zum nächsten Video. Der Softmax-Regressionsalgorithmus ist eine Verallgemeinerung der logistischen Regression, einem binären Klassifizierungsalgorithmus für Mehrklassen-Klassifizierungskontexte. Werfen wir einen Blick darauf, wie es funktioniert. Denken Sie daran, dass die logistische Regression angewendet wird, wenn y zwei mögliche Ausgabewerte annehmen kann, entweder Null oder Eins, und die Art und Weise, wie diese Ausgabe berechnet wird, ist, dass Sie zuerst z gleich w.Produkt von x plus b berechnen und dann berechnen, was I ist Ich werde hier a gleich g von z nennen, was eine auf z angewendete Sigmoidfunktion ist. Wir haben dies als logistische Regressionsschätzungen der Wahrscheinlichkeit interpretiert, dass y angesichts dieser Eingabemerkmale x gleich 1 ist. Nun eine kurze Quizfrage; Wenn die Wahrscheinlichkeit, dass y gleich 1 ist, 0,71 beträgt, wie groß ist dann die Wahrscheinlichkeit, dass y gleich Null ist? Nun, die Wahrscheinlichkeit, dass y die Eins ist, und die Chancen, dass y die Null ist, müssen zusammen eins ergeben, oder? Die Wahrscheinlichkeit, dass es eins ist, liegt also bei 71 Prozent, die Wahrscheinlichkeit, dass es gleich Null ist, muss bei 29 Prozent oder 0,29 liegen. Um die logistische Regression ein wenig zu verschönern, um uns auf die Verallgemeinerung auf die Softmax-Regression vorzubereiten, stelle ich mir die logistische Regression als die tatsächliche Berechnung zweier Zahlen vor: Erstens a_1, das ist die Größe, die wir

zuvor für die Wahrscheinlichkeit von y hatten gleich 1 bei gegebenem x ist, und zweitens stelle ich mir die logistische Regression so vor, dass auch a_2 berechnet wird, was 1 minus dies ist, was nur die Chance ist, dass y bei gegebenen Eingabemerkmalen x gleich Null ist, und somit a_1 und a_2 muss sich natürlich zu 1 addieren.




Lassen Sie uns dies nun auf die Softmax-Regression verallgemeinern, und ich werde dies anhand eines konkreten Beispiels tun, wann y vier mögliche Ausgaben annehmen kann, sodass y die Werte 1 annehmen kann , 2, 3 oder 4. Folgendes bewirkt die Softmax-Regression: Sie berechnet z_1 als w_1.Produkt mit x plus b_1, und dann ist z_2 gleich w_2.Produkt von x plus b_2 und so weiter für z_3 und z_4. Hier sind w_1, w_2, w_3, w_4 sowie b_1, b_2, b_3, b_4 die Parameter der Softmax-Regression. Als nächstes ist hier die Formel für die Softmax-Regression. Wir berechnen a_1 gleich e^z_1 dividiert durch e^ z_1 plus e^ z_2 plus e^z_3 plus e^ z_4 und a_1 wird als Schätzung des Algorithmus interpretiert Wahrscheinlichkeit, dass y angesichts der Eingabemerkmale x gleich 1 ist. Dann berechnen wir mit der Formel für die Softmax-Regression a_2 gleich e^z_2 dividiert durch denselben Nenner, e^z_1 plus e^z_2, plus e^z_3, plus e^z4, und interpretieren a_2 als Schätzung des Algorithmus der Wahrscheinlichkeit, dass y angesichts der Eingabemerkmale x gleich 2 ist. Ähnlich verhält es sich für a_3, wo hier der Zähler jetzt e^z_3 dividiert durch denselben Nenner ist. Das ist die geschätzte Wahrscheinlichkeit, dass y a_3 ist, und ähnlich nimmt a_4 diesen Ausdruck an. Während wir links die Spezifikation für das logistische Regressionsmodell aufgeschrieben haben, sind diese Gleichungen rechts unsere Spezifikation für das Softmax-Regressionsmodell. Es hat die Parameter w_1 bis w_4 und b_1 bis b_4, und wenn Sie lernen können, für alle diese Parameter die richtige Auswahl zu treffen, können Sie auf diese Weise vorhersagen, wie hoch die Wahrscheinlichkeit ist, dass y bei gegebener Menge von 1, 2, 3 oder 4 ist Eingabemerkmale x. Kurzes Quiz, mal sehen, führen Sie eine Softmax-Regression für eine neue Eingabe x durch und Sie stellen fest, dass a_1 0,30, a_2 0,20 und a_3 0,15 ist. Was glauben Sie, was a_4 sein wird? Warum werfen Sie nicht einen Blick auf dieses Quiz und finden heraus, ob Sie die richtige Antwort finden? Sie haben vielleicht bemerkt, dass die Wahrscheinlichkeit, dass y die Werte 1, 2, 3 oder 4 annimmt, eins ergeben muss, a_4. Die Wahrscheinlichkeit, dass y bei einer Vier liegt, muss 0,35 betragen, was 1 minus 0,3 ist minus 0,2 minus 0,15. Hier habe ich die Formeln für die Softmax-Regression im Fall von vier möglichen Ausgaben aufgeschrieben, und jetzt schreiben wir die Formel für den allgemeinen Fall der Softmax-Regression auf. Im allgemeinen Fall kann y n mögliche Werte annehmen, also kann y 1, 2, 3 usw. bis n sein. In diesem Fall berechnet sich die Softmax-Regression so, dass z_ j gleich w_ j ist. Produkt mit x plus b_j, wobei die Parameter der Softmax-Regression nun w_1, w_2 bis w_n sowie b_1, b_2 bis b_n

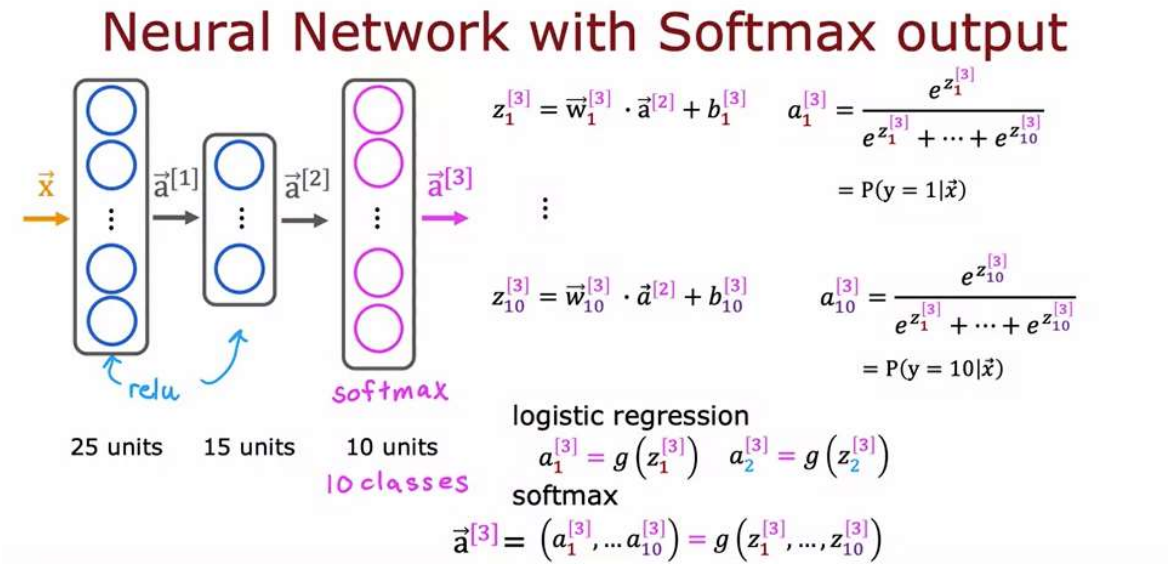
sind. Dann berechnen wir schließlich aj gleich e zu zj dividiert durch die Summe von k gleich 1 bis n von e zu z sub k. Während ich hier eine andere Variable k verwende, um die Summierung zu indizieren, weil sich hier j auf eine bestimmte feste Zahl bezieht, z. B. j gleich 1. A, j wird als die Schätzung des Modells interpretiert, dass y angesichts der Eingabemerkmale x gleich j ist. Beachten Sie, dass diese Zahlen konstruktionsbedingt immer 1 ergeben, wenn Sie a1, a2 bis zum Ende von a addieren. Wir haben angegeben, wie Sie das Softmax-Regressionsmodell berechnen würden. Ich werde es in diesem Video nicht beweisen, aber es stellt sich heraus, dass die Softmax-Regression im Grunde dasselbe berechnet wie die logistische Regression, wenn Sie die Softmax-Regression mit n gleich 2 anwenden, es also nur zwei mögliche Ausgabeklassen gibt. Die Parameter sind am Ende etwas anders, aber am Ende reduziert sich das Ergebnis auf ein logistisches Regressionsmodell. Aber deshalb ist das Softmax-Regressionsmodell die Verallgemeinerung der logistischen Regression. Nachdem wir definiert haben, wie die Softmax-Regression ihre Ausgaben berechnet, werfen wir nun einen Blick darauf, wie die Kostenfunktion für die Softmax-Regression angegeben wird. Erinnern Sie sich an die logistische Regression: Das hatten wir. Wir sagten, z sei gleich diesem. Dann habe ich früher geschrieben, dass a1 g von z ist, was als Wahrscheinlichkeit von y interpretiert wurde, die 1 ist. Wir haben auch geschrieben, dass a2 die Wahrscheinlichkeit ist, dass y gleich 0 ist. Zuvor hatten wir den Verlust der logistischen Regression als negatives y log a1 geschrieben minus 1 minus y log 1 minus a1. Aber 1 minus a1 ist auch gleich a2, weil a2 gemäß diesem Ausdruck hier eins minus a1 ist. Ich kann den Verlust für die logistische Regression ein wenig umschreiben oder vereinfachen, sodass er negativ y log a1 minus 1 minus y log von a2 ist. Mit anderen Worten: Wenn y gleich 1 ist, ist der Verlust negativ log a1. Wenn y gleich 0 ist, ist der Verlust negativ log a2, und wie zuvor ist die Kostenfunktion für alle Parameter im Modell der durchschnittliche Verlust, der Durchschnitt über den gesamten Trainingssatz. Das war eine Kostenfunktion für diese Regression. Schreiben wir die Kostenfunktion auf, die üblicherweise die Softmax-Regression verwendet. Denken Sie daran, dass dies die Gleichungen sind, die wir für die Softmax-Regression verwenden. Der Verlust, den wir für die Softmax-Regression verwenden werden, ist genau dieser. Der Verlust, wenn der Algorithmus a1 durch an führt. Die Grundwahrheitsbezeichnung besagt, dass der Verlust negativ log a1 ist, wenn y gleich 1 ist. Sagt den negativen Logarithmus der Wahrscheinlichkeit, dass y gleich 1 war, oder wenn y gleich 2 ist, dann definiere ich a2 als negativen Logarithmus. Y ist gleich 2. Der Verlust des Algorithmus in diesem Beispiel ist der negative Logarithmus der Wahrscheinlichkeit, mit der angenommen wird, dass y gleich 2 ist. Wenn y gleich n ist, ist der Verlust der negative Logarithmus von n . Um zu veranschaulichen, was dies bewirkt: Wenn y gleich j ist, dann ist der Verlust der negative Logarithmus von a j. So sieht diese Funktion aus. Der negative Logarithmus von aj ist eine Kurve, die so aussieht. Wenn aj sehr nahe bei 1 lag, liegt der Wert jenseits dieses Teils der Kurve und der Verlust wird sehr gering sein. Aber wenn es beispielsweise dachte, aj hätte nur eine 50-prozentige Chance, dann wäre der Verlust etwas größer. Je kleiner aj ist, desto größer ist der Verlust. Dies gibt dem Algorithmus einen Anreiz, aj so groß wie möglich zu machen, so nahe wie möglich an 1. Denn was auch immer der tatsächliche Wert y war, Sie möchten, dass der Algorithmus hoffentlich sagt, dass die Wahrscheinlichkeit, dass y dieser Wert ist, ziemlich groß war. Beachten Sie, dass y in dieser Verlustfunktion in jedem Trainingsbeispiel nur einen Wert annehmen kann. Am Ende berechnen Sie diesen negativen Logarithmus von aj nur für einen Wert von aj, nämlich für den tatsächlichen Wert von y gleich j in diesem speziellen Trainingsbeispiel. Wenn y beispielsweise gleich 2 wäre, berechnen Sie am Ende den negativen Logarithmus von a2, aber nicht den anderen negativen Logarithmus von a1 oder die anderen Terme hier.




Das ist die Form des Modells sowie die Kostenfunktion für die Softmax-Regression. Wenn Sie dieses Modell trainieren, können Sie mit der Erstellung von Klassifizierungsalgorithmen für mehrere Klassen beginnen. Als Nächstes möchten wir dieses Softmax-Regressionsmodell in ein neues Netzwerk integrieren, sodass Sie wirklich etwas noch Besseres tun können, nämlich ein neues Netzwerk für die Klassifizierung mehrerer Klassen zu trainieren. Lassen Sie uns das im nächsten Video durchgehen.

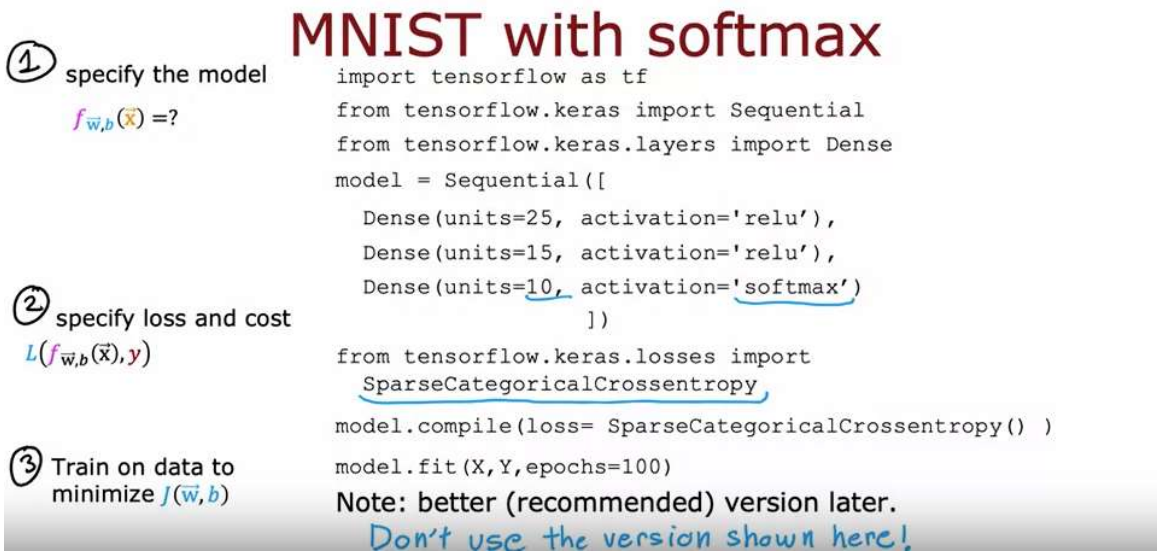
## Neuronales Netzwerk mit Softmax-Ausgabe

Um ein neuronales Netzwerk aufzubauen, das eine Klassifizierung mehrerer Klassen durchführen kann, nehmen wir das Softmax-Regressionsmodell und fügen es im Wesentlichen in die Ausgabeschicht eines neuronalen Netzwerks ein. Werfen wir einen Blick darauf, wie das geht. Zuvor haben wir die handschriftliche Ziffernerkennung mit nur zwei Klassen durchgeführt. Mit dieser Architektur verwenden wir ein neues neuronales Netzwerk. Wenn Sie nun eine handschriftliche Ziffernklassifizierung mit 10 Klassen durchführen möchten, alle Ziffern von Null bis Neun, dann werden wir dieses neuronale Netzwerk so ändern, dass es etwa 10 Ausgabeeinheiten hat. Und diese neue Ausgabeschicht wird eine Softmax-Ausgabeschicht sein.




Manchmal sagen wir also, dass dieses neuronale Netzwerk einen Softmax-Ausgang hat oder dass diese obere Schicht eine Softmax-Schicht ist. Und die Art und Weise, wie die Vorwärtsausbreitung in diesem neuronalen Netzwerk funktioniert, wird bei gegebener Eingabe X A1 genauso berechnet wie

zuvor. Und dann, A2, werden auch die Aktivierungen für die zweite verborgene Schicht genauso berechnet wie zuvor. Und wir müssen jetzt die Aktivierungen für diese Ausgabeschicht berechnen , also a3. So funktioniert es. Wenn Sie 10 Ausgabeklassen haben, berechnen wir Z1, Z2 bis Z10 anhand dieser Ausdrücke. Das ist also tatsächlich sehr ähnlich zu dem, was wir zuvor für die Formel hatten, die Sie zur Berechnung von Z verwenden. Z1 ist W1.Produkt mit a2, den Aktivierungen aus der vorherigen Ebene plus b1 und so weiter für Z1 bis Z10. Dann ist a1 gleich e zu Z1 geteilt durch e zu Z1 plus bis zu e zu Z10. Und das ist unsere Schätzung der Wahrscheinlichkeit, dass y gleich 1 ist. Und das Gleiche gilt für a2 und das Gleiche bis hin zu a10. Dies gibt Ihnen also Ihre Schätzung der Wahrscheinlichkeit, dass y gut zu eins, zwei usw. ist, bis hin zum zehnten möglichen Label für y. Und der Vollständigkeit halber: Wenn Sie angeben möchten, dass dies die Mengen sind, die der dritten Schicht zugeordnet sind, sollte ich technisch gesehen diese Super-Strip-Dreier hinzufügen. Dadurch wird die Notation etwas unübersichtlicher. Dies macht jedoch deutlich, dass es sich beispielsweise um den Wert Z( 3), 1 und die Parameter handelt, die der ersten Einheit der dritten Schicht dieses neuronalen Netzwerks zugeordnet sind. Und damit liefert Ihnen Ihre offene Softmax-Ebene nun Schätzungen der Wahrscheinlichkeit, dass y eines dieser 10 möglichen Ausgabeetiketten ist. Ich möchte erwähnen, dass die Softmax-Ebene manchmal auch als Softmax-Aktivierungsfunktion bezeichnet wird. In einer Hinsicht ist sie im Vergleich zu den anderen Aktivierungsfunktionen, die wir bisher gesehen haben, wie Sigma, Radial und Linear, etwas ungewöhnlich Wenn wir Sigmoid- oder Wert- oder lineare Aktivierungsfunktionen betrachten, war a1 eine Funktion von Z1 und a2 eine Funktion von Z2 und nur von Z2. Mit anderen Worten, um die Aktivierungswerte zu erhalten, könnten wir die Aktivierungsfunktion g, sei es sigmoid oder selten oder etwas anderes elementweise, auf Z1 und Z2 usw. anwenden, um a1 und a2 sowie a3 und a4 zu erhalten. Beachten Sie jedoch, dass a1 bei der Softmax-Aktivierungsfunktion eine Funktion von Z1 und Z2 und Z3 bis hin zu Z10 ist. Jeder dieser Aktivierungswerte hängt also von allen Werten von Z ab. Und dies ist eine Eigenschaft, die ein wenig einzigartig für die Softmax-Ausgabe oder die Softmax-Aktivierungsfunktion ist oder anders ausgedrückt, wenn Sie a1 bis a10 berechnen möchten eine Funktion von Z1 bis hin zu Z 10 gleichzeitig. Und das ist anders als die anderen Aktivierungsfunktionen, die wir bisher gesehen haben. Schauen wir uns abschließend an, wie Sie dies in Tensorflow implementieren würden. Wenn Sie das neuronale Netzwerk implementieren möchten, das ich hier auf dieser Folie gezeigt habe, ist dies der Code dafür. Ähnlich wie zuvor gibt es drei Schritte zum Spezifizieren und Trainieren des Modells. Der erste Schritt besteht darin, Tensorflow anzuweisen, drei Schichten nacheinander aneinanderzureihen. Die erste Schicht besteht aus 25 Einheiten mit Rail-You-Aktivierungsfunktion. Zweite Schicht, 15 Einheiten Rallye-Aktivierungsfunktion. Und dann die dritte Ebene: Da es jetzt 10 Ausgabeeinheiten gibt, möchten Sie a1 bis a10 ausgeben, also sind es jetzt 10 Ausgabeeinheiten. Und wir werden Tensorflow anweisen, die Softmax-Aktivierungsfunktion zu verwenden.




Und die Kostenfunktion, die Sie im letzten Video gesehen haben, Tensorflow ruft die SparseCategoricalCrossentropy-Funktion auf. Ich weiß also, dass dieser Name ein bisschen langatmig ist, während wir für die logistische Regression die BinaryCrossentropy-Funktion hatten, verwenden wir hier die SparseCategoricalCrossentropy-Funktion. Und was „sparse categorical" bedeutet, ist, dass Sie immer noch in Kategorien eingeteilt sind. Es ist also kategorisch. Dies nimmt Werte von 1 bis 10 an. Und sparse bezieht sich darauf, dass y nur einen dieser 10 Werte annehmen kann. Jedes Bild ist also entweder 0 oder 1 oder 2 oder so weiter bis 9. Sie werden kein Bild sehen, das gleichzeitig die Zahl zwei und die Zahl sieben ist, die so spärlich sind, dass jede Ziffer nur einer dieser Kategorien entspricht. Aber deshalb wird die Verlustfunktion, die Sie im letzten Video gesehen haben, trotz der spärlichen kategorialen Kreuzentropieverlustfunktion als intensiv bezeichnet. Und dann ist der Code zum Trainieren des Modells genau derselbe wie zuvor. Und wenn Sie diesen Code verwenden, können Sie ein neuronales Netzwerk auf ein Klassifizierungsproblem mit mehreren Klassen trainieren. Nur ein wichtiger Hinweis: Wenn Sie diesen Code genau so verwenden, wie ich ihn hier geschrieben habe, wird er funktionieren, aber verwenden Sie diesen Code eigentlich nicht, da sich herausstellt, dass es in Tensorflow eine bessere Version des Codes gibt, die dafür sorgt, dass Tensorflow besser funktioniert . Obwohl der in dieser Folie gezeigte Code funktioniert. Verwenden Sie diesen Code nicht so, wie ich ihn hier geschrieben habe, denn in einem späteren Video dieser Woche sehen Sie eine andere Version, die eigentlich die empfohlene Version für die Implementierung ist, die besser funktionieren wird. Aber wir werden uns das in einem späteren Video ansehen. Jetzt wissen Sie also, wie Sie mit einer Einschränkung ein neuronales Netzwerk mit einer Softmax-Ausgabeschicht trainieren. Es gibt eine andere Version des Codes, mit der Tensorflow diese Wahrscheinlichkeiten viel genauer berechnen kann. Schauen wir uns das im nächsten Video an. Wir sollten Ihnen auch den tatsächlichen Code zeigen, den ich Ihnen empfehle, wenn Sie ein neuronales Softmax-Netzwerk trainieren. Kommen wir zum nächsten Video.

## Verbesserte Implementierung von Softmax

Die Implementierung eines neuronalen Netzwerks mit einer Softmax-Schicht, die Sie im letzten Video gesehen haben, wird einwandfrei funktionieren. Aber es gibt eine noch bessere Möglichkeit, es umzusetzen. Werfen wir einen Blick darauf, was bei dieser Implementierung schief gehen kann und wie wir sie verbessern können. Lassen Sie mich Ihnen zwei verschiedene Möglichkeiten zeigen, dieselbe Menge in einem Computer zu berechnen.

# Numerical Roundoff Errors

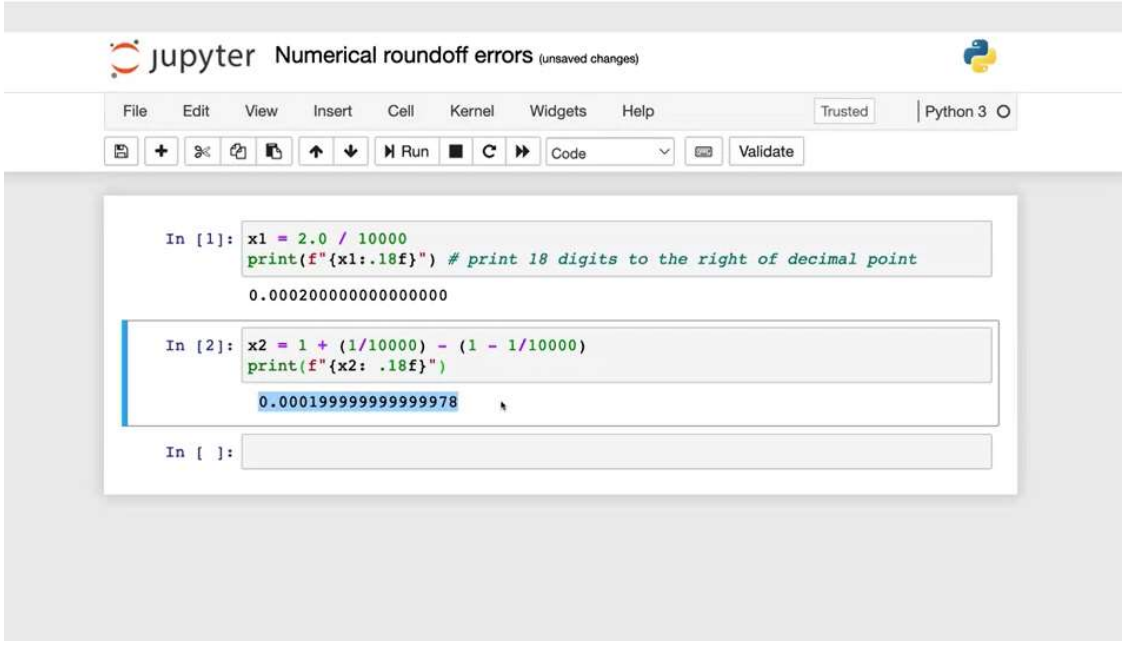
option 1

$$x = \frac{2}{10{,}000}$$

option 2

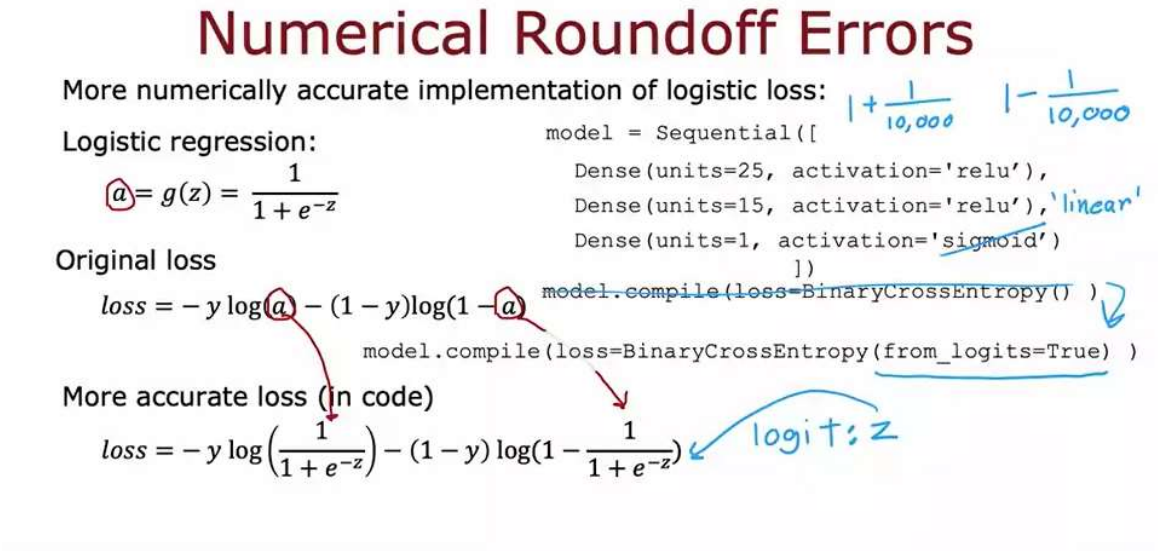
$$x = \left(1 + \frac{1}{10{,}000}\right) - \left(1 - \frac{1}{10{,}000}\right)$$

Option 1: Wir können x auf 2/10.000 setzen. Option 2: Wir können x gleich 1 plus 1/10.000 minus 1 minus 1/10.000 setzen, wobei Sie dies zuerst berechnen und dann dies berechnen und die Differenz bilden. Wenn Sie diesen Ausdruck vereinfachen, ergibt sich ein Wert von 2/10.000. Lassen Sie mich dies in diesem Notizbuch veranschaulichen. Zuerst setzen wir x gleich 2/10.000 und geben das Ergebnis mit einer Genauigkeit von vielen Dezimalstellen aus. Das sieht ziemlich gut aus. Zweitens setze ich x gleich. Ich werde darauf bestehen, 1/1 plus 10.000 zu berechnen und dann davon 1 minus 1/10.000 zu subtrahieren. Drucken wir das aus. Es sieht nur ein wenig anders aus, als ob ein Rundungsfehler vorliegt. Da der Computer nur über eine begrenzte Menge an Speicher verfügt, um jede Zahl zu speichern, die in diesem Fall als Gleitkommazahl bezeichnet wird, kann das Ergebnis abhängig davon, wie Sie den Wert 2/10.000 berechnen, mehr oder weniger numerische Rundungsfehler aufweisen.



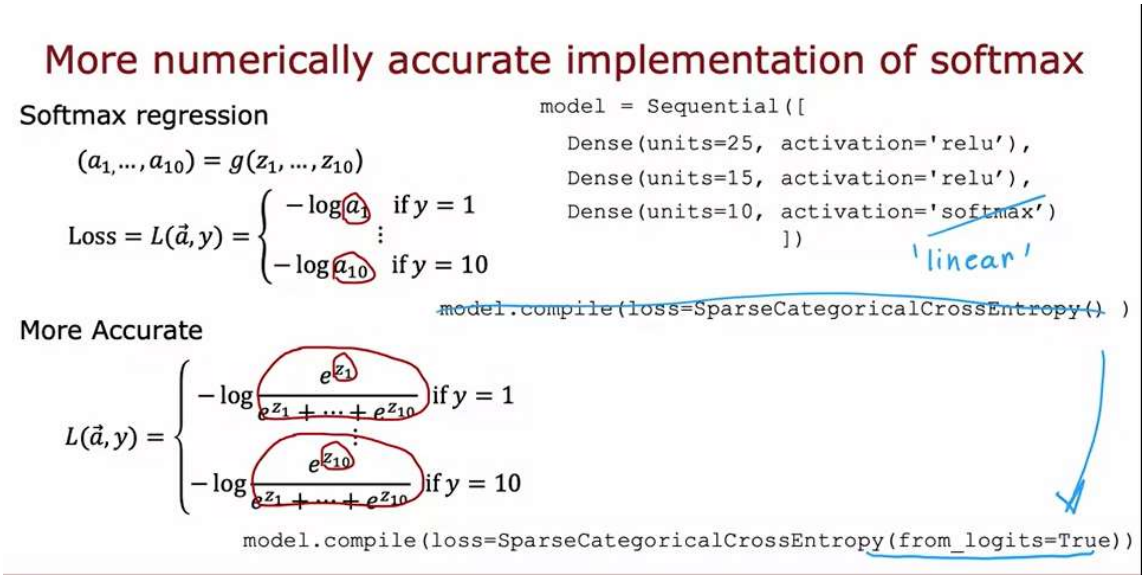

Es stellt sich heraus, dass die Art und Weise, wie wir die Kostenfunktion für Softmax berechnet haben, zwar korrekt ist, es jedoch eine andere Art der Formulierung gibt, die diese numerischen Rundungsfehler reduziert, was zu genaueren Berechnungen innerhalb von TensorFlow führt. Lassen Sie mich dies zunächst anhand der logistischen Regression etwas detaillierter erläutern. Anschließend zeigen wir, wie sich diese Ideen auf die Verbesserung unserer Softmax-Implementierung anwenden

lassen. Lassen Sie mich diese Ideen zunächst anhand der logistischen Regression veranschaulichen. Anschließend zeigen wir Ihnen, wie Sie auch Ihre Softmax-Implementierung verbessern können. Denken Sie daran, dass Sie für die logistische Regression, wenn Sie die Verlustfunktion für ein gegebenes Beispiel berechnen möchten, zuerst diese Ausgabeaktivierung a berechnen würden, die g von z oder 1/1 plus e zum negativen z ist. Dann berechnen Sie den Verlust mit diesem Ausdruck hier. Tatsächlich würden die Codes für eine logistische Ausgabeschicht mit diesem binären Kreuzentropieverlust so aussehen. Bei der logistischen Regression funktioniert das einwandfrei, und normalerweise sind die numerischen Rundungsfehler nicht so schlimm. Es stellt sich jedoch heraus, dass, wenn Sie TensorFlow zulassen, a nicht als Zwischenterm berechnet werden muss. Aber stattdessen sagen Sie TensorFlow, dass dieser Ausdruck hier unten verloren geht. Alles, was ich getan habe, ist, dass ich es hier unten zu diesem Ausdruck erweitert habe. Dann kann TensorFlow die Terme in diesem Ausdruck neu anordnen und eine numerisch genauere Methode zur Berechnung dieser Verlustfunktion finden. Während das ursprüngliche Verfahren darin bestand, darauf zu bestehen, als Zwischenwert 1 plus 1/10.000 und einen weiteren Zwischenwert, 1 minus 1/10.000, zu berechnen und diese beiden dann zu manipulieren, um 2/10.000 zu erhalten.

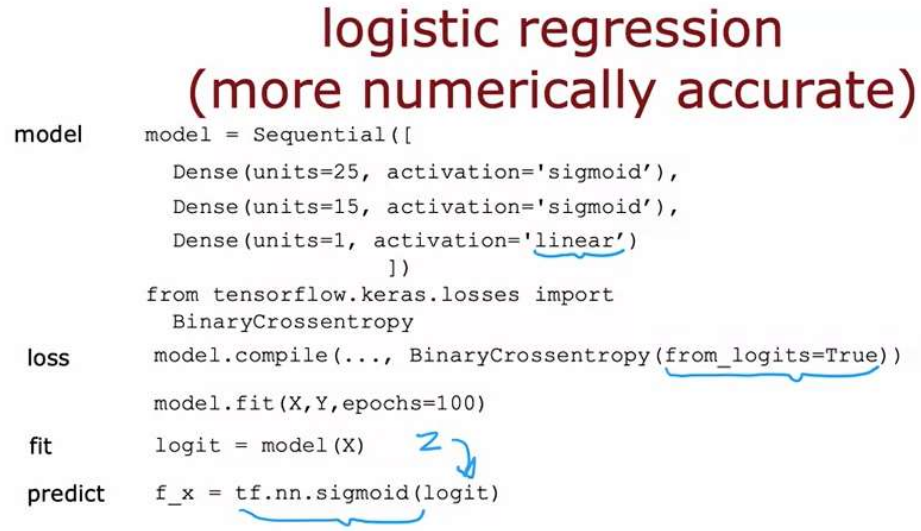



Diese Teilimplementierung bestand darauf, a explizit als Zwischengröße zu berechnen. Durch die direkte Angabe dieses Ausdrucks unten als Verlustfunktion erhält TensorFlow jedoch mehr Flexibilität hinsichtlich der Berechnung dieser Funktion und der Frage, ob a explizit berechnet werden soll oder nicht. Der Code, den Sie dazu verwenden können, wird hier gezeigt. Dadurch wird die Ausgabeschicht so eingestellt, dass sie nur eine lineare Aktivierungsfunktion verwendet, und sowohl die Aktivierungsfunktion, 1/1 plus zum negativen z, als auch dieses Kreuz werden gesetzt Entropieverlust in die Spezifikation der Verlustfunktion hier einbeziehen. Das ist es, was dieses from logits equal true-Argument dazu veranlasst, dass TensorFlow dies tut. Falls Sie sich fragen, was die Logits sind: Im Grunde ist es diese Zahl z. TensorFlow berechnet z als Zwischenwert, kann die Terme jedoch neu anordnen, um eine genauere Berechnung zu ermöglichen. Ein Nachteil dieses Codes ist, dass er etwas weniger lesbar wird. Dies führt jedoch dazu, dass TensorFlow einen etwas geringeren numerischen Rundungsfehler aufweist. Im Fall der logistischen Regression funktioniert jede dieser Implementierungen tatsächlich einwandfrei, aber die numerischen Rundungsfehler können bei Softmax schlimmer werden. Nehmen wir nun diese Idee und wenden sie auf die Softmax-Regression an. Erinnern Sie sich daran, was Sie im letzten Video gesehen haben, als Sie die Aktivierungen wie folgt berechnet haben. Die Aktivierungen sind g von z_1 bis z_10, wobei a_1 zum Beispiel e zu z_1 dividiert durch die Summe von e zu z_j ist, und dann war der Verlust davon abhängig, was der tatsächliche Wert von y ist, ein negativer Logarithmus von aj für einen der ajs und das war also der

Code, den wir für diese Berechnung in zwei separaten Schritten durchführen mussten. Aber noch einmal: Wenn Sie stattdessen angeben, dass der Verlust auftritt, wenn y gleich 1 ist, ist der Logarithmus dieser Formel negativ und so weiter. Wenn y gleich 10 ist, ist diese Formel, dann gibt dies TensorFlow die Möglichkeit, Terme neu anzuordnen und dieses Integral numerisch genau zu berechnen. Um Ihnen eine Vorstellung davon zu geben, warum TensorFlow dies tun möchte: Es stellt sich heraus, dass, wenn einer der Z-Werte, die wirklich kleiner als e bis zu einer negativen kleinen Zahl sind, sehr, sehr klein wird oder wenn einer der Z-Werte eine sehr große Zahl ist, dann z zum z kann eine sehr große Zahl werden und durch Neuanordnen der Begriffe kann TensorFlow einige dieser sehr kleinen oder sehr großen Zahlen vermeiden und daher mehr Rechenleistung für die Verlustfunktion liefern. Der Code dafür wird hier in der Ausgabeebene angezeigt.



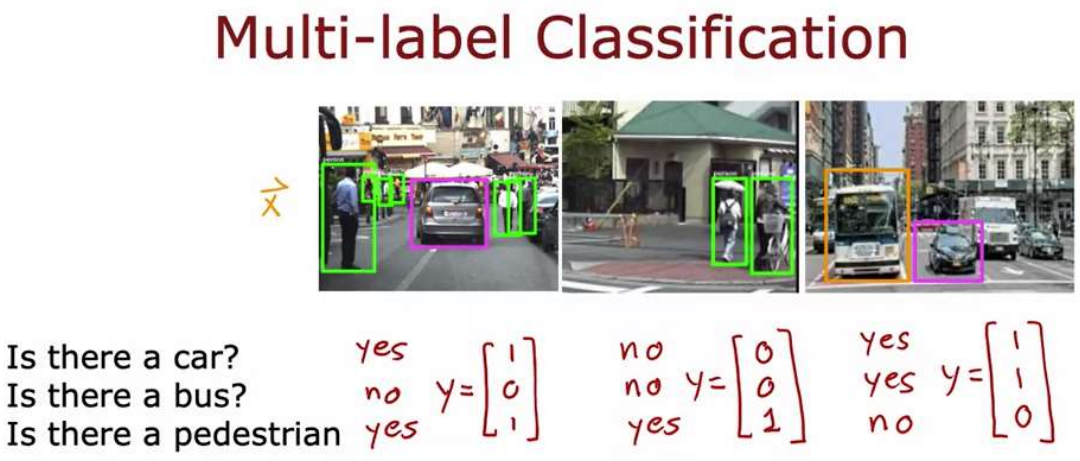

Wir verwenden jetzt nur eine lineare Aktivierungsfunktion, sodass die Ausgabeebene nur z_1 bis z_10 berechnet und diese gesamte Verlustberechnung dann in der Verlustfunktion hier erfasst wird wieder haben wir den Parameter from logists equal true. Auch hier bewirken diese beiden Codeteile im Wesentlichen das Gleiche, mit der Ausnahme, dass die empfohlene Version numerisch genauer ist, obwohl sie leider auch etwas schwieriger zu lesen ist. Wenn Sie den Code einer anderen Person lesen und dies sehen und sich fragen, was da vor sich geht, entspricht es tatsächlich der ursprünglichen Implementierung, zumindest im Konzept, außer dass es numerisch genauer ist. Die numerischen Rundungsfehler für die_logist-Regression sind nicht so schlimm, aber es wird empfohlen, stattdessen diese Implementierung bis zum Ende zu verwenden, und konzeptionell macht dieser Code dasselbe wie die erste Version, die Sie zuvor hatten, außer dass es sich um eine handelt etwas numerisch genauer. Obwohl die Kehrseite vielleicht auch etwas schwieriger zu interpretieren ist. Jetzt gibt es nur noch ein Detail: Wir haben das neuronale Netzwerk jetzt so geändert, dass es eine lineare Aktivierungsfunktion anstelle einer Softmax-Aktivierungsfunktion verwendet. Die letzte Schicht des neuronalen Netzwerks gibt diese Wahrscheinlichkeiten A_1 bis A_10 nicht mehr aus.



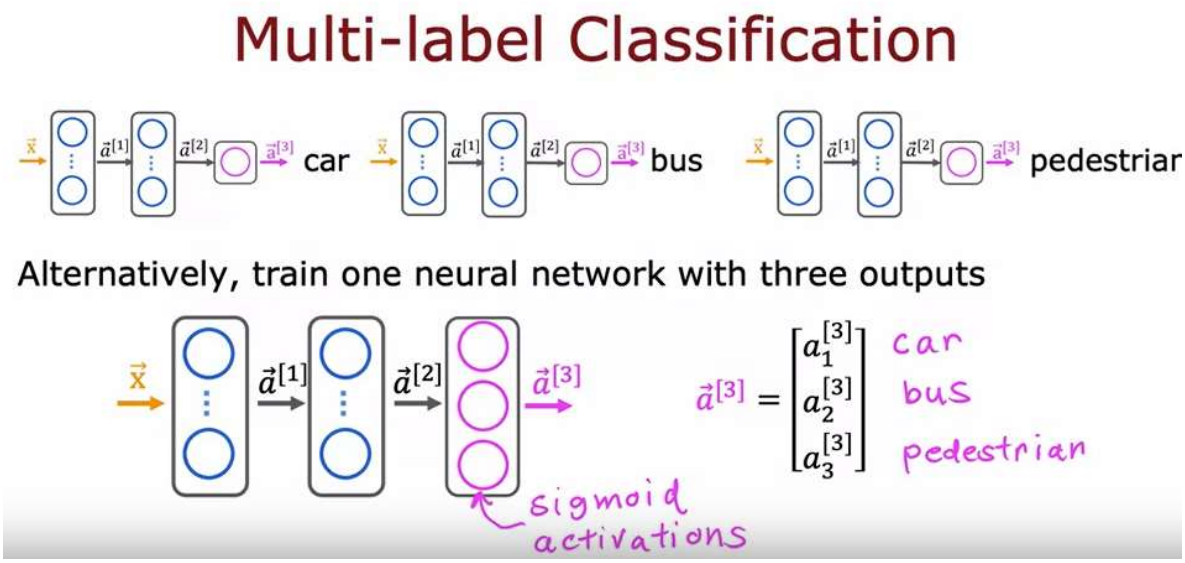

Dies geschieht, anstatt z_1 bis z_10 zu setzen. Im Fall der logistischen Regression habe ich nicht darüber gesprochen, aber wenn Sie die logistische Funktion der Ausgabe mit der Verlustfunktion kombinieren, müssen Sie für logistische Regressionen auch den Code auf diese Weise ändern, um den Ausgabewert zu nehmen und ihn abzubilden durch die Logistikfunktion, um tatsächlich die Wahrscheinlichkeit zu erhalten. Sie wissen jetzt, wie Sie eine Mehrklassenklassifizierung mit einer Softmax-Ausgabeschicht durchführen und wie Sie diese auf numerisch stabile Weise durchführen. Bevor ich die Klassifizierung mehrerer Klassen abschließe, möchte ich Ihnen eine andere Art von Klassifizierungsproblem vorstellen, das als Klassifizierungsproblem mit mehreren Etiketten bezeichnet wird. Lassen Sie uns im nächsten Video darüber sprechen.

## Klassifizierung mit mehreren Ausgaben (multi_Label)

Sie haben etwas über die Klassifizierung mehrerer Klassen gelernt, bei der die Ausgabebezeichnung Y eine von zwei oder möglicherweise viel mehr als zwei mögliche Kategorien sein kann. Es gibt eine andere Art von Klassifizierungsproblem, das sogenannte Multi-Label-Klassifizierungsproblem, bei dem es sich bei der Zuordnung jedes Bildes um mehrere Labels handeln kann.



Lassen Sie mich Ihnen zeigen, was ich damit meine. Wenn Sie ein selbstfahrendes Auto oder vielleicht ein Fahrerassistenzsystem bauen, möchten Sie vielleicht, wenn Sie ein Bild davon haben, was sich vor Ihrem Auto befindet, eine Frage stellen wie: Gibt es ein Auto oder mindestens ein Auto? Oder gibt es einen Bus, oder gibt es einen Fußgänger oder gibt es überhaupt Fußgänger? In diesem Fall gibt es ein Auto, keinen Bus und mindestens einen Fußgänger oder in diesem zweiten Bild keine Autos, keine Busse und ja zu Fußgängern und ja ein Auto, ja einen Bus und keine Fußgänger. Dies sind Beispiele für Klassifizierungsprobleme mit mehreren Etiketten, da mit einer einzelnen Eingabe, Bild In diesem Fall ist das Ziel des Y tatsächlich ein Vektor aus drei Zahlen, und dies unterscheidet sich von der Mehrklassenklassifizierung, bei der beispielsweise bei der handschriftlichen Ziffernklassifizierung Y nur eine einzelne Zahl war, selbst wenn diese Zahl annehmen könnte 10 verschiedene mögliche Werte. Wie baut man ein neuronales Netzwerk für die Multi-Label-Klassifizierung auf? Eine Möglichkeit besteht darin, dies einfach als drei völlig separate Probleme des maschinellen Lernens zu behandeln. Sie könnten ein neuronales Netzwerk aufbauen, um zu entscheiden: Gibt es Autos? Der zweite dient zur Erkennung von Bussen und der dritte zur Erkennung von Fußgängern. Das ist eigentlich kein unvernünftiger Ansatz. Hier ist das erste neuronale Netzwerk zur Erkennung von Autos, das zweite zur Erkennung von Bussen und das dritte zur Erkennung von Fußgängern. Aber es gibt noch eine andere Möglichkeit, dies zu tun, nämlich ein einzelnes neuronales Netzwerk so zu trainieren, dass es alle drei Autos, Busse und Fußgänger gleichzeitig erkennt. Das heißt, wenn Ihre neuronale Netzwerkarchitektur so aussieht, gibt es Eingabe X. Die erste verborgene Schicht bietet a^1, die zweite verborgene Ebene bietet a^2 und dann die letzte Ausgabeebene. In diesem Fall haben wir drei Ausgabeneuronen und geben a^3 aus, was ein Vektor aus drei Zahlen sein wird. Da wir drei binäre Klassifizierungsprobleme lösen, gibt es also ein Auto? Gibt es einen Bus? Gibt es einen Fußgänger? Sie können für jeden dieser drei Knoten in der Ausgabeschicht eine Sigmoid-Aktivierungsfunktion verwenden. Daher ist a^3 in diesem Fall a_1^3, a_2^3 und a_3^3, je nachdem, ob das Lernen [unverständlich ist ] als Auto und kein Bus und keine Fußgänger im Bild. Multi-Class-Klassifizierung und Multi-Label-Klassifizierung werden manchmal miteinander verwechselt. Aus diesem Grund möchte ich Ihnen in diesem Video auch nur eine Definition von Multi-Label-Klassifizierungsproblemen mitteilen, damit Sie je nach Anwendung eine Auswahl treffen können das Richtige für den Job, den Sie machen möchten.



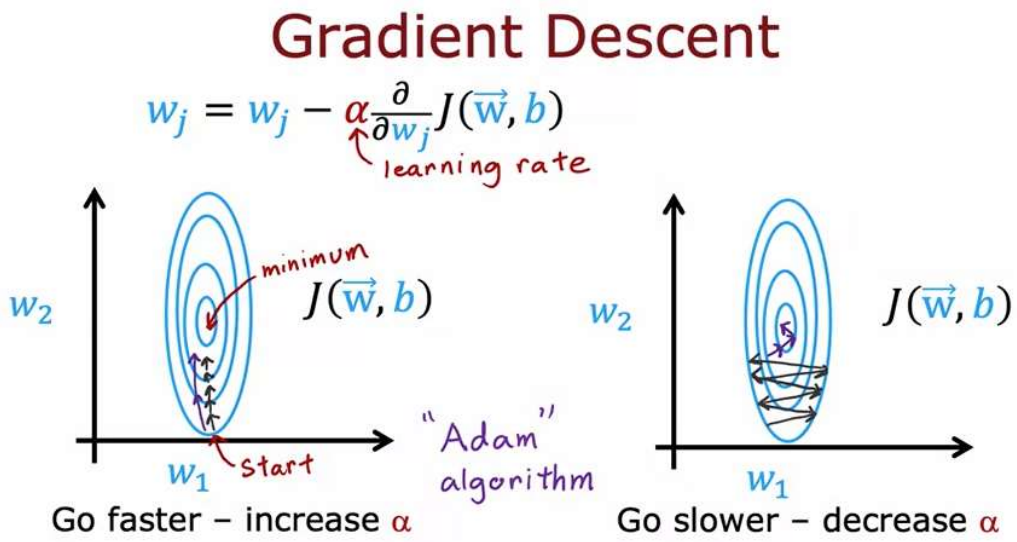

Das war's also mit der Multi-Label-Klassifizierung. Ich finde, dass die Klassifizierung mehrerer Klassen und die Klassifizierung mehrerer Labels manchmal miteinander verwechselt werden. Aus diesem Grund möchte ich Ihnen in diesem Video ausführlich erklären, was eine Klassifizierung mehrerer Labels ist, sodass Sie je nach Anwendung eine Auswahl treffen können Schreiben Sie an für den Job,

den Sie erledigen möchten. Damit ist der Abschnitt über die Klassifizierung mehrerer Klassen und Etiketten abgeschlossen. Im nächsten Video beginnen wir mit der Betrachtung einiger fortgeschrittener neuronaler Netzwerkkonzepte, einschließlich eines Optimierungsalgorithmus, der noch besser ist als der Gradientenabstieg. Schauen wir uns diesen Algorithmus im nächsten Video an, denn er wird Ihnen helfen, Ihre Lernalgorithmen viel schneller zum Lernen zu bringen. Kommen wir zum nächsten Video.

## Erweiterte Optimierung

Der Gradientenabstieg ist ein Optimierungsalgorithmus, der im maschinellen Lernen weit verbreitet ist und die Grundlage vieler Algorithmen wie der linearen Regression und der logistischen Regression sowie früher Implementierungen neuronaler Netze bildete. Es stellt sich jedoch heraus, dass es mittlerweile einige andere Optimierungsalgorithmen zur Minimierung der Kostenfunktion gibt, die sogar noch besser sind als der Gradientenabstieg. In diesem Video werfen wir einen Blick auf einen Algorithmus, der Ihnen dabei helfen kann, Ihr neuronales Netzwerk viel schneller als beim Gradientenabstieg zu trainieren. Denken Sie daran, dass dies der Ausdruck für einen Schritt des Gradientenabstiegs ist.

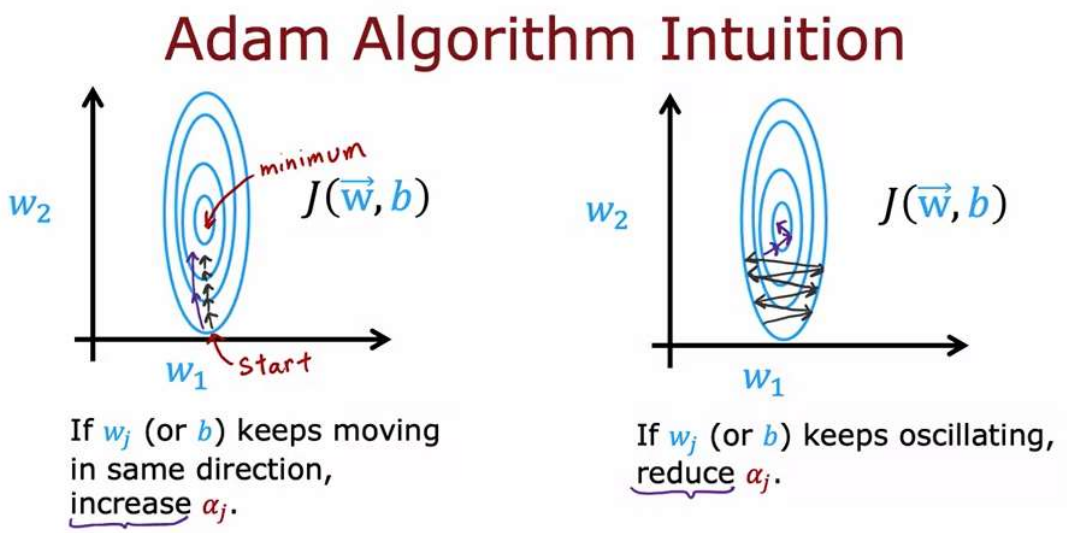
Ein Parameter w_j wird als w_j minus der Lernrate Alpha multipliziert mit diesem partiellen Ableitungsterm aktualisiert. Wie können wir diese Arbeit noch besser machen? In diesem Beispiel habe ich die Kostenfunktion J mithilfe eines Konturdiagramms dargestellt, das diese Ellipsen umfasst, und das Minimum dieser Kostenfunktion befindet sich hier unten in der Mitte dieser Ellipse. Wenn Sie nun hier unten mit dem Gefälleabstieg beginnen würden, könnte ein Schritt des Gefälles, wenn Alpha klein ist, Sie ein kleines Stück in diese Richtung führen. Dann noch ein Schritt, dann noch ein Schritt, dann noch ein Schritt, dann noch ein Schritt, und Sie bemerken, dass jeder einzelne Schritt des Gradientenabstiegs im Wesentlichen in die gleiche Richtung verläuft, und wenn Sie sehen, dass dies der Fall ist, fragen Sie sich vielleicht, nun ja , warum machen wir Alpha nicht größer? Können wir einen Algorithmus haben, um Alpha automatisch zu vergrößern?




Sie machen einfach größere Schritte und erreichen das Minimum schneller. Es gibt einen Algorithmus namens Adam-Algorithmus, der das kann. Wenn festgestellt wird, dass die Lernrate zu gering ist und wir immer wieder winzige Schritte in die gleiche Richtung unternehmen, sollten wir die Lernrate Alpha einfach erhöhen. Im Gegensatz dazu gilt auch hier wieder die gleiche Kostenfunktion. Wenn wir hier beginnen und eine relativ große Lernrate Alpha haben, führt uns vielleicht ein Schritt des

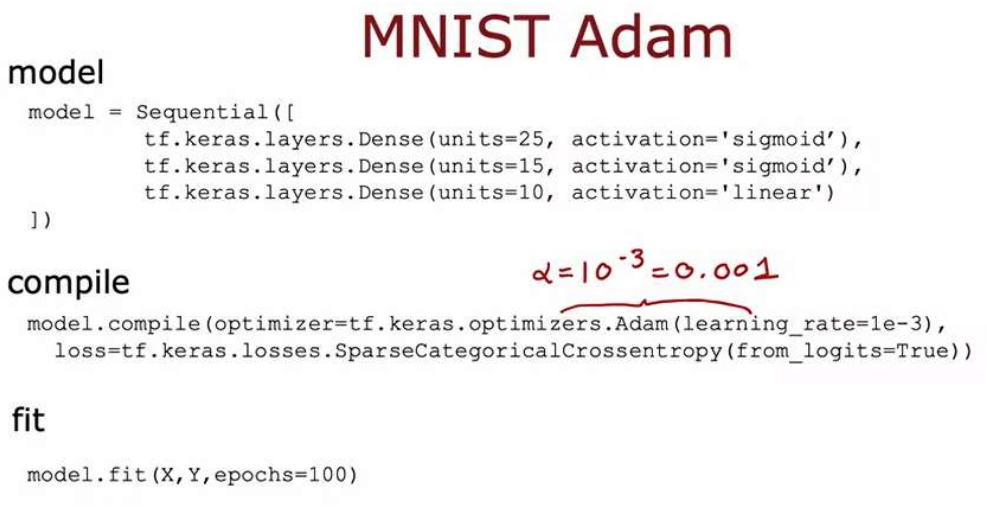
Gradientenabstiegs hierher, der zweite Schritt führt uns hierher, der dritte Schritt und der vierte Schritt und der fünfte Schritt und der sechste Schritt, und wenn Sie dabei einen Gradientenabstieg sehen, oszilliert er hin und her. Sie wären versucht zu sagen: Warum verringern wir nicht die Lernraten? Der Adam-Algorithmus kann dies auch automatisch tun, und mit einer geringeren Lernrate können Sie dann einen sanfteren Weg zum Minimum der Kostenfunktion einschlagen. Je nachdem, wie der Gradientenabstieg verläuft, wünschen Sie sich manchmal eine größere Lernrate Alpha und manchmal wünschen Sie sich eine kleinere Lernrate Alpha. Der Adam-Algorithmus kann die Lernrate automatisch anpassen.

Adam steht für Adaptive Moment Estimation oder ADAM, und machen Sie sich keine allzu großen Gedanken darüber, was dieser Name bedeutet, es ist einfach so, wie die Autoren diesen Algorithmus genannt haben. Aber interessanterweise verwendet der Adam-Algorithmus keine einzige globale Lernrate Alpha. Es verwendet unterschiedliche Lernraten für jeden einzelnen Parameter Ihres Modells. Wenn Sie die Parameter w_1 bis w_10 haben, wie b, dann hat es tatsächlich 11 Lernratenparameter, Alpha_1, Alpha_2, bis hin zu Alpha_10 für w_1 bis w_10, und ich nenne es auch Alpha_11 für den Parameter b. Die Intuition hinter dem Adam-Algorithmus besteht darin, dass sich ein Parameter w_j oder b scheinbar weiterhin in ungefähr die gleiche Richtung bewegt. Das haben wir im ersten Beispiel auf der vorherigen Folie gesehen. Wenn es jedoch den Anschein hat, dass es sich weiterhin ungefähr in die gleiche Richtung bewegt, erhöhen wir die Lernrate für diesen Parameter. Gehen wir schneller in diese Richtung. Wenn umgekehrt ein Parameter ständig hin und her schwankt, haben Sie dies im zweiten Beispiel auf der vorherigen Folie gesehen. Dann lassen wir es nicht ständig hin und her schwingen oder springen. Lassen Sie uns Alpha_j für diesen Parameter ein wenig reduzieren.




Die Details, wie Adam dies macht, sind etwas kompliziert und würden den Rahmen dieses Kurses sprengen. Wenn Sie jedoch später einige fortgeschrittenere Deep-Learning-Kurse belegen, erfahren Sie möglicherweise mehr über die Details dieses Adam-Algorithmus, aber in Codes ist dies der Fall füge es ein. Das Modell ist genau das gleiche wie zuvor, und die Art und Weise, wie Sie das Modell kompilieren, ist der vorherigen sehr ähnlich, mit der Ausnahme, dass wir der Kompilierungsfunktion jetzt ein zusätzliches Argument hinzufügen, nämlich dass wir den Optimierer angeben, den Sie verwenden möchten ist der Optimierer tf.keras.optimizers.Adam. Der Adam-Optimierungsalgorithmus benötigt eine standardmäßige anfängliche Lernrate Alpha, und in diesem Beispiel habe ich diese anfängliche Lernrate auf 10^ minus 3 festgelegt. Wenn Sie den Adam-Algorithmus jedoch in der Praxis verwenden, lohnt es sich, einige davon auszuprobieren Werte für

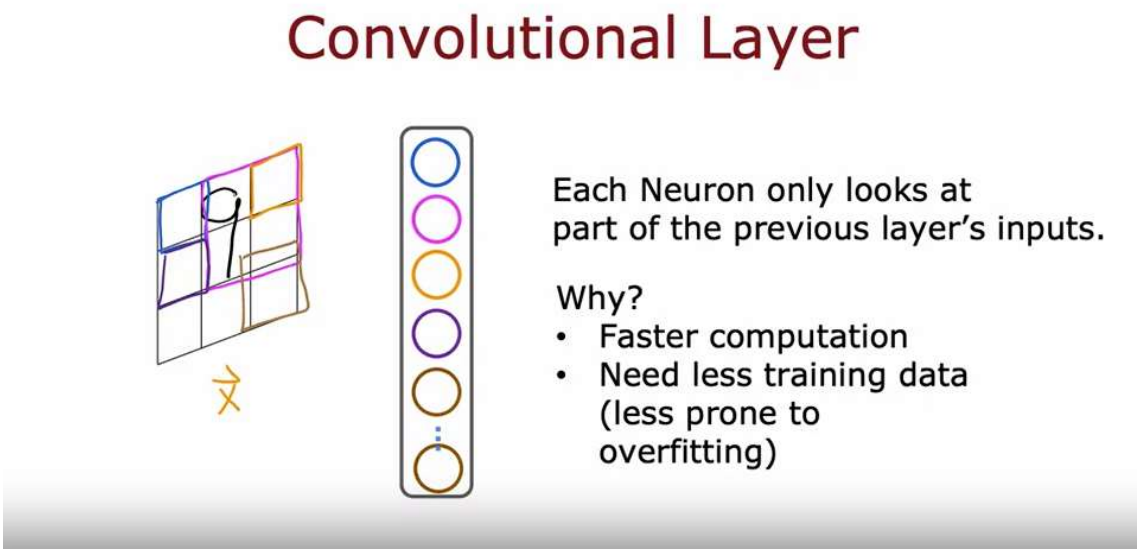
diese standardmäßige globale Lernrate. Probieren Sie einige große und einige kleinere Werte aus, um herauszufinden, was Ihnen die schnellste Lernleistung bietet. Im Vergleich zum ursprünglichen Gradientenabstiegsalgorithmus, den Sie im vorherigen Kurs gelernt haben, dem Adam-Algorithmus, ist er robuster gegenüber der genauen Wahl der von Ihnen gewählten Lernrate, da er die Lernrate ein wenig automatisch anpassen kann. Allerdings gibt es immer noch die Möglichkeit, diesen Parameter ein wenig zu optimieren, um zu sehen, ob Sie etwas schneller lernen können. Das ist alles für den Adam-Optimierungsalgorithmus. Es funktioniert normalerweise viel schneller als der Gradientenabstieg und ist zu einem De-facto-Standard für die Art und Weise geworden, wie Praktiker ihre neuronalen Netze trainieren. Wenn Sie entscheiden möchten, welchen Lernalgorithmus Sie verwenden möchten, welchen Optimierungsalgorithmus Sie zum Trainieren Ihres neuronalen Netzwerks verwenden möchten.




Eine sichere Wahl wäre, einfach den Adam-Optimierungsalgorithmus zu verwenden, und die meisten Praktiker werden heute Adam anstelle des optionalen Gradientenabstiegsalgorithmus verwenden, und ich hoffe, dass Ihre Lernalgorithmen damit viel schneller lernen können. In den nächsten Videos möchte ich nun auf einige fortgeschrittenere Konzepte für neuronale Netze eingehen, und insbesondere im nächsten Video:

## Zusätzliche Layer-Typen

Alle bisher verwendeten neuronalen Netzwerkschichten waren vom dichten Schichttyp, bei dem jedes Neuron in der Schicht seine Eingaben und alle Aktivierungen von der vorherigen Schicht erhält. Und es stellt sich heraus, dass Sie allein mit dem Dense Layer tatsächlich einige ziemlich leistungsstarke Lernalgorithmen erstellen können. Und um Ihnen dabei zu helfen, ein tieferes Verständnis dafür zu entwickeln, was neuronale Netze leisten können. Es stellt sich heraus, dass es auch einige andere Ebenentypen mit anderen Eigenschaften gibt.

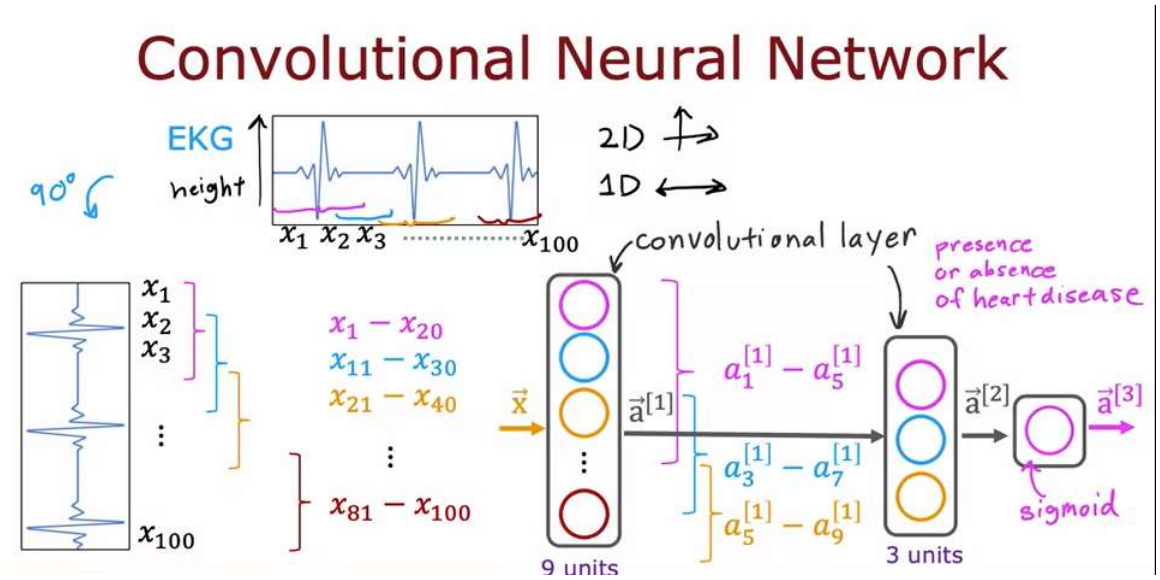



In diesem Video möchte ich kurz darauf eingehen und Ihnen ein Beispiel für eine andere Art von neuronaler Netzwerkschicht geben. Werfen wir einen Blick darauf, um in der dichten Schicht noch einmal zusammenzufassen, dass die Aktivierung eines Neurons beispielsweise in der zweiten verborgenen Schicht eine Funktion jedes einzelnen Aktivierungswerts aus der vorherigen Schicht einer Eins ist. Es stellt sich jedoch heraus, dass sich jemand, der ein neuronales Netzwerk entwirft, für einige Anwendungen für die Verwendung eines anderen Schichttyps entscheidet.

Ein anderer Ebenentyp, den Sie möglicherweise in einigen Arbeiten sehen, wird als Faltungsschicht (convolutional Layer) bezeichnet. Lassen Sie mich dies anhand eines Beispiels verdeutlichen. Was ich also links zeige, ist die Eingabe X. Das ist eine handgeschriebene Ziffer Neun. Und was ich tun werde, ist, eine verborgene Ebene zu konstruieren, die verschiedene Aktivierungen als Funktionen dieses Eingabebildes X berechnet. Aber hier ist etwas, was ich für die erste verborgene Einheit tun kann, die ich blau gezeichnet habe, anstatt dieses Neuron zu sagen kann alle Pixel in diesem Bild betrachten. Ich könnte sagen, dass dieses Neuron nur die Pixel in diesem kleinen rechteckigen Bereich betrachten kann. Das zweite Neuron, das ich in Magenta veranschaulichen werde, wird stattdessen auch nicht das gesamte Eingabebild X betrachten, sondern nur die Pixel in einem begrenzten Bereich des Bildes. Und so weiter für das dritte Neuron und das 4. Neuron und so weiter und so weiter. Bis zum letzten Neuron, das möglicherweise nur diesen Bereich des Bildes betrachtet. Warum sollten Sie das also tun? Warum lässt man nicht jedes Neuron alle Pixel betrachten, sondern nur einige Pixel? Nun, einige der Vorteile sind erstens: Es beschleunigt die Berechnung. Und der zweite Vorteil besteht darin, dass ein neuronales Netzwerk, das diese Art von Schicht verwendet, die als Faltungsschicht bezeichnet wird, möglicherweise weniger Trainingsdaten benötigt oder alternativ auch weniger anfällig für Überanpassung ist. Sie haben gehört, dass ich in ihrem vorherigen Kurs ein wenig über Überanpassung gesprochen habe, aber darauf werde ich nächste Woche auch detaillierter eingehen. Wenn wir über praktische Tipps für die Verwendung von Lernalgorithmen sprechen und diese Art von Schicht, bei der jedes Neuron nur einen Bereich des Eingabebildes betrachtet, als Faltungsschicht bezeichnet wird.

Es war der Forscher John Macoun, der viele Details herausgefunden hatte, wie man Faltungsschichten zum Laufen bringt, und deren Verwendung populär gemacht hatte. Lassen Sie mich eine Faltungsschicht detaillierter veranschaulichen. Und wenn es in einem neuronalen Netzwerk mehrere Faltungsschichten gibt, spricht man manchmal von einem Faltungs-Neuronalen Netzwerk. Um die Faltungsschicht des Faltungs-Neuronalen Netzwerks auf dieser Folie zu veranschaulichen, verwende ich anstelle einer zwei-D-Bildeingabe. Ich werde eine eindimensionale

Eingabe verwenden und das motivierende Beispiel, das ich verwenden werde, ist die Klassifizierung von EKG-Signalen oder Elektrokardiogrammen.




Wenn Sie also zwei Elektroden an Ihrer Brust anbringen, zeichnen Sie die Spannungen auf, die so aussehen und Ihrem Herzschlag entsprechen. Das ist tatsächlich etwas, worüber meine Stanford-Forschungsgruppe geforscht hat. Wir haben tatsächlich EKG-Signale gelesen, die tatsächlich so aussehen, um zu diagnostizieren, ob Patienten möglicherweise ein Herzproblem haben. Also ein EKG-Signal und ein Kardia-Graham-EKG an manchen Stellen EK G. An manchen Orten gibt es nur eine Liste von Zahlen, die der Höhe der Oberfläche zu verschiedenen Zeitpunkten entsprechen. Sie können also beispielsweise 100 Zahlen haben, die der Höhe dieser Kurve zu 100 verschiedenen Zeitpunkten entsprechen. Und die Lernenden werfen anhand dieser Zeitreihe und anhand dieses EKG-Signals eine Klassifizierung vor, um zu sagen, ob dieser Patient eine Herzerkrankung oder diagnostizierbare Herzerkrankungen hat.

Hier ist, was das Convolutional-Neuronal Network tun könnte. Ich nehme also das EKG-Signal und drehe es um 90 Grad, um es auf die Seite zu legen. Wir haben also hier 100 Eingänge X eins x zwei bis hin zu X 100. Wie. Wenn ich also die erste verborgene Ebene konstruiere, müssen wir statt der ersten verborgenen Einheit alle 100 Zahlen eingeben. Geben Sie mir die erste versteckte Einheit. Schauen Sie sich nur X eins bis X 20 an. Das entspricht also der Betrachtung nur eines kleinen Fensters dieses EKG-Signals. Die zweite versteckte Einheit wird hier in einer anderen Farbe angezeigt. Schauen Sie sich X 11 bis X 30 an, sehen Sie sich also ein anderes Fenster in diesem EKG-Signal an. Und das dritte dort versteckte Fenster blickt auf ein weiteres Fenster X21 bis X40 und so weiter. Und die letzten versteckten Einheiten in diesem Beispiel. Schauen Sie sich X 81 bis X100 an. Es sieht also wie ein kleines Fenster gegen Ende dieser EKG-Zeitreihe aus. Dies ist also eine Faltungsschicht, da diese Einheiten in dieser Schicht nur ein begrenztes Fenster der Eingabe betrachten. Nun besteht diese Schicht des neuronalen Netzwerks aus neun Einheiten. Die nächste Schicht kann auch eine Faltungsschicht sein. Lassen Sie mich also in der zweiten verborgenen Ebene meine erste Einheit so konstruieren, dass sie nicht alle neun Aktivierungen der vorherigen Ebene betrachtet, sondern beispielsweise nur die ersten fünf Aktivierungen der vorherigen Ebene. Und dann meine zweite Einheit. In dieser zweiten versteckten Einheit sehen Sie möglicherweise nur weitere fünf Zahlen, sagen wir A3-A7. Und die dritte und letzte verborgene Einheit in dieser Ebene betrachtet nur A5 bis A9. Und dann vielleicht endlich diese Aktivierungen. A2 erhält Eingaben an eine Sigmoideinheit, die alle drei dieser Werte von A2 betrachtet, um eine binäre Klassifizierung hinsichtlich des Vorliegens oder Nichtvorhandenseins einer Herzerkrankung vorzunehmen. Dies ist also das Beispiel eines

neuronalen Netzwerks, bei dem die erste verborgene Schicht eine Faltungsschicht ist. Die zweite verborgene Schicht ist ebenfalls eine Faltungsschicht und die Ausgabeschicht ist eine Sigmoidschicht. Und es stellt sich heraus, dass Sie bei Faltungsschichten viele Architekturoptionen haben, z. B. wie groß das Eingabefenster ist, das ein einzelnes Neuron betrachten soll, und wie viele Neuronen die Schicht haben soll. Und indem Sie diese Architekturparameter effektiv wählen, können Sie neue Versionen neuronaler Netze erstellen, die für einige Anwendungen sogar noch effektiver sein können als die dichte Schicht. Um es noch einmal zusammenzufassen: Das war's für die Faltungsschicht und die Faltungs-Neuronalen Netze. Ich werde in diesem Kurs nicht tiefer auf Faltungsnetzwerke eingehen und Sie müssen nichts über sie wissen, um die Hausaufgaben zu machen und diesen Kurs erfolgreich abzuschließen. Aber ich hoffe, dass Sie diese zusätzliche Einsicht, dass neuronale Netze auch andere Arten von Schichten haben können, nützlich finden. Und tatsächlich, wenn Sie manchmal von den neuesten Spitzenarchitekturen hören, wie einem Transformer-Modell oder einem LS TM oder einem Attention-Modell. Ein Großteil dieser Forschung zu neuronalen Netzen betrifft auch heute noch Forscher, die versuchen, neue Arten von Schichten für neuronale Netze zu erfinden. Und diese verschiedenen Arten von Schichten als Bausteine zusammenzufügen, um noch komplexere und hoffentlich leistungsfähigere neuronale Netze zu bilden. Das sind also die benötigten Videos für diese Woche. Vielen Dank und herzlichen Glückwunsch, dass Sie die ganze Zeit an meiner Seite geblieben sind. Und ich freue mich darauf, Sie nächste Woche wiederzusehen. Dann werden wir beginnen, über praktische Ratschläge zum Aufbau maschineller Lernsysteme zu sprechen. Ich hoffe, dass die Tipps, die Sie nächste Woche lernen, Ihnen dabei helfen werden, nützliche Systeme für maschinelles Lernen viel effektiver zu entwickeln.

## Was ist ein Derivat? (Optional)

Sie haben gesehen, wie Sie in TensorFlow eine neuronale Netzwerkarchitektur angeben können, um die Ausgabe y als Funktion der Eingabe x zu berechnen, und auch eine Kostenfunktion angeben können. TensorFlow verwendet dann automatisch die Rückausbreitung, um Ableitungen zu berechnen und den Gradientenabstieg zu verwenden Adam trainiert die Parameter eines neuronalen Netzwerks. Der Backpropagation-Algorithmus, der Ableitungen Ihrer Kostenfunktion in Bezug auf die Parameter berechnet, ist ein Schlüsselalgorithmus beim Lernen neuronaler Netze. Doch wie funktioniert es eigentlich? In diesem und den nächsten optionalen Videos werden wir versuchen, einen Blick darauf zu werfen, wie Backpropagation Ableitungen berechnet. Diese Videos sind völlig optional und befassen sich nur ein wenig mit der Analysis. Wenn Sie bereits mit Infinitesimalrechnung vertraut sind, hoffe ich, dass Ihnen diese Videos gefallen, aber wenn nicht, ist das völlig in Ordnung. Wir bauen auf den Grundlagen der Analysis auf, um sicherzustellen, dass Sie über die nötige Intuition verfügen, um zu verstehen, wie Backpropagation funktioniert. Lass uns einen Blick darauf werfen. Ich werde eine vereinfachte Kostenfunktion verwenden, J von w ist gleich w im Quadrat. Die Kostenfunktion ist eine Funktion der Parameter w und beispielsweise b. Nehmen wir für diese vereinfachte Kostenfunktion einfach an, dass J von w gleich w im Quadrat ist. Ich werde b für dieses Beispiel ignorieren. Nehmen wir an, der Wert des Parameters w ist gleich 3. J von w ist gleich 9, w im Quadrat von 3 im Quadrat. Nun, wenn wir w um einen kleinen Betrag erhöhen würden, sagen wir Epsilon, den ich auf 0,001 setzen werde. Wie ändert sich der Wert von J von w? Wenn wir w um 0,001 erhöhen, wird w zu 3 plus 0,001, also 3,001. J von w, also dem Quadrat von w, das wir oben definiert haben, beträgt jetzt 3,001 zum Quadrat, also 9,006001. Was wir sehen ist, dass wenn w um 0,001 steigt, ich diesen Aufwärtspfeil hier verwenden werde, um anzuzeigen, dass w um 0,001 steigt, wobei 0,001 dieser kleine Wert Epsilon ist. Dann steigt J von w ungefähr um das Sechsfache, also um das Sechsfache von 0,001. Das ist nicht ganz genau. Es steigt tatsächlich nicht auf 9,006, sondern auf 9,006001. Aber es stellt sich heraus, dass, wenn Epsilon unendlich klein wäre, und mit unendlich klein meine ich sehr klein. Epsilon ist ziemlich klein, aber nicht unendlich klein. Wenn Epsilon 0,00000 wäre, viele Nullen gefolgt von einer Eins, dann wird dies immer genauer. In

diesem Beispiel sehen wir, dass, wenn w um Epsilon steigt, J etwa um das 6-fache Epsilon steigt. In der Analysis würden wir sagen, dass die Ableitung von J von w nach w gleich 6 ist. Das bedeutet nur, dass J von w um das Sechsfache steigt, wenn w um einen winzigen Betrag steigt. Was wäre, wenn Epsilon einen anderen Wert annehmen würde? Was wäre, wenn Epsilon 0,002 wäre? In diesem Fall wäre w 3 plus 0,002, und w im Quadrat wird zu 3,002 im Quadrat, also 9,012004. In diesem Fall kommen wir zu dem Schluss, dass, wenn w um 0,002 steigt, J von w etwa um das Sechsfache von 0,002 steigt. Es steigt ungefähr auf 9,012, und diese 0,012 ist ungefähr 6 mal 0,002. Das ist wieder ein bisschen daneben. Das sind hier zusätzliche 0,00004, da Epsilon nicht ganz unendlich klein ist. Wieder einmal sehen wir dieses Verhältnis von sechs zu eins zwischen dem Anstieg von w und dem Anstieg von J von w. Deshalb ist die Ableitung von J von w nach w gleich sechs. Je kleiner Epsilon, desto genauer wird dies. Übrigens können Sie das Video auch gerne pausieren und diese Berechnung jetzt selbst mit anderen Werten von Epsilon ausprobieren. Der Schlüssel liegt darin, dass, solange Epsilon ziemlich klein ist, das Verhältnis zwischen J von w und dem Betrag, um den w steigt, 6-1 betragen sollte. Probieren Sie es gerne selbst mit anderen Werten von Epsilon aus und prüfen Sie dann, ob dies wirklich zutrifft. Dies führt uns zu einer informellen Definition der Ableitung, die besagt, dass jedes Mal, wenn w um einen winzigen Betrag Epsilon ansteigt, J von w um das k-fache Epsilon ansteigt. In unserem Beispiel war k gleich sechs. Dann sagen wir, dass die Ableitung von J von w nach w gleich k ist, was im Beispiel gerade gleich 6 war. Sie erinnern sich vielleicht, dass Sie bei der Implementierung des Gradientenabstiegs diese Regel wiederholt verwenden, um den Parameter w J zu aktualisieren, wobei Alpha wie üblich die Lernrate ist. Was bewirkt der Gradientenabstieg? Beachten Sie, dass diese Aktualisierung zu einer kleinen Aktualisierung des Parameters W_j führt, wenn die Ableitung klein ist. Wenn dieser Ableitungsterm hingegen groß ist, führt dies zu einer großen Änderung des Parameters W_j. Das macht Sinn, denn das bedeutet im Grunde, dass eine Änderung von w keinen großen Unterschied im Wert von j macht, wenn die Ableitung klein ist. Wir sollten uns also nicht die Mühe machen, eine große Änderung an W_j vorzunehmen. Aber wenn die Ableitung groß ist, bedeutet das, dass selbst eine kleine Änderung von W_j einen großen Unterschied darin machen kann, wie stark Sie die Kostenfunktion j von w ändern oder verringern können. In diesem Fall nehmen wir eine größere Änderung an W_j vor, da dies tatsächlich einen großen Unterschied darin macht, wie stark wir die Kostenfunktion J reduzieren können. Schauen wir uns noch ein paar weitere Beispiele für Ableitungen an. Was Sie gerade im Beispiel gesehen haben, war, dass, wenn w gleich 3 ist und j von w gleich w zum Quadrat gleich 9 ist, dann, wenn w um Epsilon um 0,01 steigt, j von w zu j von 3,01 wird, was jetzt 9,006001 ist. Mit anderen Worten: j ist um etwa 0,006 gestiegen, was dem Sechsfachen von 0,001 oder dem Sechsfachen von Epsilon entspricht, weshalb die Ableitung von w nach W gleich 6 ist. Schauen wir uns an, wie die Ableitung für andere Werte aussehen wird von w, nehme w gleich 2. In diesem Fall ist j von w, w im Quadrat ist jetzt gleich 4, und wenn w um 0,001 steigt, dann wird J von w zu j von 2,001, was gleich 4,004001 ist und so weiter j von w ist hier von vier auf diesen Wert gestiegen, was etwa viermal Epsilon größer als vier ist, weshalb die Ableitung jetzt vier beträgt. Denn der Anstieg von w um Epsilon hat dazu geführt, dass j von w um das Vierfache gestiegen ist. Auch hier gibt es zusätzliche 0,001, weil es nicht ganz genau ist, weil Epsilon unendlich klein ist. Oder schauen wir uns ein anderes Beispiel an. Was wäre, wenn w gleich minus 3 wäre? J von w, das w zum Quadrat ist, ist immer noch gleich 9, weil minus 3 zum Quadrat 9 ist. Wenn w wieder um Epsilon steigen würde, dann wäre w jetzt gleich minus 2,999, das ist also j von minus 2,999. Das Quadrat von minus 2,999 ist gleich 8,994001, weil w minus 3 plus 0,001 ist. Beachten Sie hier, dass j von w um etwa 0,006 gesunken ist, was dem Sechsfachen von Epsilon entspricht. Was wir in diesem Beispiel haben, ist, dass j mit 9 beginnt, aber jetzt gesunken ist. Beachten Sie diesen Abwärtspfeil hier [unhörbar] um das 6-fache Epsilon, bzw. er ist um das 6-fache Epsilon nach oben gestiegen. Aus diesem Grund ist die Ableitung in diesem Fall gleich minus 6. Denn wenn w um Epsilon steigt, steigt j von w um minus 6 mal Epsilon, wenn Epsilon klein ist. Eine andere Möglichkeit, dies zu veranschaulichen, besteht darin, die Funktion J von w so darzustellen, dass die

horizontale Achse w ist und dies J von w ist. Wenn w gleich 3 ist, ist J von w gleich 9. Wenn es negativ 3 ist, ist es auch so gleich 9, und wenn es 2 ist, ist J von w gleich 4. Lassen Sie mich eine Beobachtung machen, die relevant sein könnte, wenn Sie schon einmal einen Mathematikkurs besucht haben. Aber wenn nicht, ergibt das, was ich in den nächsten 60 Sekunden sage, möglicherweise keinen Sinn, aber machen Sie sich darüber keine Sorgen. Sie müssen es verstehen, um den Rest dieser Videos vollständig verfolgen zu können. Wenn Sie irgendwann einen Kurs in Analysis besucht haben, werden Sie vielleicht erkennen, dass die Ableitungen der Steigung einer Geraden entsprechen, die gerade die Funktion J von w an diesem Punkt berührt, sagen wir, wo w gleich 3 ist. Die Steigung dieser Geraden bei Dieser Punkt, und die Steigung dieser Höhe über dieser Breite ergibt sich als gleich 6, wenn w gleich 3 ist, die Steigung dieser Linie ergibt sich als 4, wenn w gleich 2 ist, und die Steigung dieser Linie ergibt sich als negativ 6 wenn w gleich minus 3 ist. In der Analysis stellt sich heraus, dass die Steigung dieser Geraden der Ableitung der Funktion entspricht. Aber wenn Sie noch nie einen Mathematikkurs besucht und dieses Steigungskonzept noch nie gesehen haben, machen Sie sich darüber keine Sorgen. Nun möchte ich noch eine letzte Beobachtung machen, bevor ich fortfahre: Sie sehen in allen drei Beispielen, dass J von w dieselbe Funktion ist und J von w gleich w im Quadrat ist. Die Ableitung von J von w hängt jedoch von w ab. Wenn w drei ist, beträgt die Ableitung sechs. Wenn w zwei ist, ist die Ableitung vier. Wenn w negativ 3 ist, ist die Ableitung negativ 6. Es stellt sich heraus, dass wir mit der Analysis die Ableitung von J von w in Bezug auf berechnen können, wenn Sie mit Analysis vertraut sind – und auch das ist völlig in Ordnung, wenn Sie nicht damit vertraut sind w wie 2 mal w. Gleich zeige ich Ihnen, wie Sie Python verwenden können, um diese Ableitungen mithilfe eines raffinierten Python-Pakets namens SymPy selbst zu berechnen. Aber weil uns die Infinitesimalrechnung sagt, dass die Ableitung von w zum Quadrat von J von w 2w ist, ist die Ableitung, wenn w drei ist, 2 mal 3, oder wenn zwei ist, ist sie 2 mal 2, oder wenn negativ 3 ist, ist sie 2 mal negativ 3, weil dies der Fall ist Der Wert von w mal 2 ergibt die Ableitung. Lassen Sie uns noch ein paar Beispiele durchgehen, bevor wir zum Schluss kommen. Für diese Beispiele setze ich w gleich 2. Sie haben auf der letzten Folie gesehen, dass, wenn J von w w zum Quadrat ist, die Ableitung, die ich gesagt habe, 2 mal w wäre, also 4. Wenn w um 0,01 steigt Da dies Epsilon ist, wird J von w zu diesem Wert, sodass J von w ungefähr um das Vierfache von Epsilon steigt. Schauen wir uns noch ein paar andere Funktionen an. Was ist, wenn J von w gleich w kubisch ist? In diesem Fall wäre w kubisch, 2 kubisch wäre gleich 8, oder was wäre, wenn J von w gerade gleich w wäre? Hier ist w gleich 2. Oder was wäre, wenn J von w 1 über w wäre? In diesem Fall wäre 1 über w, 1 über 2 1/2 oder 0,5. Was ist die Ableitung von J von w nach w, wenn die Kostenfunktion J von w entweder kubisch w oder w oder 1 über w ist? Lassen Sie mich Ihnen zeigen, wie Sie diese Ableitungen mithilfe einer Bibliothek und eines Pakets namens SymPy selbst berechnen können. Lassen Sie mich zunächst SymPy importieren. Ich werde SymPy mitteilen, dass ich J und w als Symbole für die Berechnung von Ableitungen verwenden werde. Für unser erstes Beispiel hatten wir, dass die Kostenfunktion J gleich w im Quadrat war. Beachten Sie auch hier, wie SymPy es tatsächlich in dieser raffinierten Schriftart darstellt. Wenn wir SymPy verwenden würden, um die Ableitung von J nach w zu ermitteln, sollten wir wie folgt vorgehen. Sie sehen, dass SymPy Ihnen sagt, dass diese Ableitung 2w ist. Lassen Sie mich tatsächlich eine Variable auswählen, dJ, dw , wir setzen sie auf diesen Wert, geben Sie sie einfach hier noch einmal ein. Drucke es aus. Es gibt 2 W. Wenn Sie den Wert von w in diesen Ausdruck einfügen möchten, um ihn auszuwerten, können Sie derivative.subs w, 2 ausführen. Dies bedeutet, dass Sie einen Wert von w, der gleich 2 ist, in diesen Ausdruck einfügen und ihn auswerten. Das ergibt den Wert vier, weshalb wir sahen, dass die Ableitung von J gleich 4 war, wenn w gleich 2 war. Schauen wir uns einige andere Beispiele an. Was wäre, wenn J w-gewürfelt wäre? Dann beträgt die Ableitung das Dreifache von w im Quadrat. Aus der Infinitesimalrechnung geht hervor, und das ist es, was SymPy für uns berechnet: Wenn J die Kubikzahl w ist, dann beträgt die Ableitung von J nach w 3w im Quadrat. Je nachdem, was w ist, ändert sich auch der Wert der Ableitung . Wir können einstecken, wenn w gleich 2 ist, in diesem Fall erhalten Sie 12. Oder was

wäre, wenn J gleich w wäre? In diesem Fall ist die Ableitung gerade gleich 1. Oder das letzte Beispiel, das wir haben, war, was wäre, wenn J gleich 1 über w wäre? In diesem Fall ergibt sich, dass die Ableitung über w zum Quadrat negativ 1 ist. Das ist minus 1 zu 4. Ich werde die Ableitungen verwenden, die wir berechnet haben. Denken Sie daran, dass es für w im Quadrat 2w war, für w im Quadrat 3w im Quadrat. Da w einfach 1 und 1 über w ist, ist es negativ 1 über w im Quadrat. Kopieren wir dies zurück auf unsere andere Folie. Was SymPy oder eigentlich die Infinitesimalrechnung uns gezeigt hat, ist, dass, wenn J von w gleich w ist, die Ableitung 3w zum Quadrat beträgt, was gleich 12 ist, wenn w gleich 2 ist, und wenn J von w gleich w ist, ist die Ableitung gerade gleich 1. Wenn J von w ist 1 über w ist negativ 1 über w im Quadrat, was negativ 1/4 ist, wenn w gleich 2 ist. Fangen wir an. Wir prüfen, ob diese Ausdrücke, die wir von SymPy erhalten haben, korrekt sind. Versuchen wir, w um Epsilon zu erhöhen, in diesem Fall J von w. Wenn Sie möchten, können Sie das Video jederzeit anhalten und diese Berechnung auf Ihrem eigenen Taschenrechner überprüfen. Aber in diesem Fall wird J von w auf 0,001 kubisch zu diesem. J ist also ungefähr von 8 auf 8,012 gestiegen. Es ist um etwa das Zwölffache des Epsilon gestiegen. Somit ist die Ableitung tatsächlich 12. Oder wenn J von w gleich w ist und w um Epsilon zunimmt, dann ist J von w, das einfach w ist, jetzt 2,001. Er ist also um 0,01 gestiegen, was genau dem Wert von Epsilon entspricht. J von w ist also um das 1-fache Epsilon gestiegen. Die Ableitung ist tatsächlich gleich 1. Beachten Sie, dass dies hier tatsächlich genau Epsilon ist, obwohl Epsilon unendlich klein ist. Wenn in unserem letzten Beispiel J von w gleich 1/w ist, wenn w um Epsilon steigt, dann ist w 1/2,001, dann stellt sich heraus, dass J von w ungefähr 4,9975 ist, wobei einige zusätzliche Ziffern abgeschnitten sind. Aber das ergibt 0,5 minus 0,00025. J von w begann bei 0,5 und ist um 0,00025 gesunken. Diese 0,00025 ist das 0,25-fache Epsilon. Es ist um diesen Betrag gesunken oder um das negative 0,25-fache Epsilon gestiegen, weil das negative 0,25-fache Epsilon dieser Summe hier entspricht. Wir sehen, dass, wenn w um Epsilon steigt, J von w um minus 1/4 oder minus 0,25 mal Epsilon steigt, weshalb die Ableitung in diesem Fall minus 1/4 ist. Ich hoffe, dass Sie anhand dieser Beispiele eine Vorstellung davon bekommen, was die Ableitung nach w von J von w bedeutet. Es ist nur so: Wenn w um Epsilon steigt, um wie viel erhöht sich dann J von w um eine Konstante k mal Epsilon? Diese Konstante k ist die Ableitung. Der Wert von k hängt sowohl von der Funktion J von w als auch vom Wert von w ab. Bevor wir dieses Video abschließen, möchte ich kurz auf die Notation eingehen, die zum Schreiben von Ableitungen verwendet wird, die Sie möglicherweise in anderen Texten sehen. Das heißt, wenn J von w eine Funktion einer einzelnen Variablen ist, sagen wir w, dann schreiben Mathematiker die Ableitung manchmal als d/dw von J von w. Beachten Sie, dass diese Notation hier den Kleinbuchstaben d verwendet. Wenn dagegen J eine Funktion von mehr als einer Variablen ist, verwenden Mathematiker manchmal diese verschnörkelte Alternative d, um die Ableitung von J nach einem der Parameter w_i zu bezeichnen. Meiner Meinung nach macht diese Notation, die zwischen diesem regulären Buchstaben d und diesem stilisierten Kalkül-Ableitungssymbol d unterscheidet, für mich wenig Sinn, diese Unterscheidung vorzunehmen und diese Notation meiner Meinung nach die Infinitesimalrechnung zu kompliziert zu machen und diese Notation abzuleiten. Aber aus historischen Gründen werden in Kalkültexten diese beiden unterschiedlichen Notationen verwendet, je nachdem, ob J eine Funktion einer einzelnen Variablen oder eine Funktion mehrerer Variablen ist. Aber ich denke, aus praktischen Gründen führt diese Notationskonvention dazu, die Dinge einfach zu kompliziert zu machen, und zwar auf eine Weise, die ich nicht für notwendig halte. Für diese Klasse werde ich diese Notation einfach überall verwenden, auch wenn es nur eine einzige Variable gibt. Tatsächlich ist die Funktion J für die meisten unserer Anwendungen eine Funktion von mehr als einer Variablen. Diese andere Notation, die manchmal als partielle Ableitungsnotation bezeichnet wird, ist eigentlich fast immer die richtige Notation, da J normalerweise mehr als eine Variable hat. Aber ich hoffe, dass die Verwendung dieser Notation in diesen Vorlesungen die Darstellung vereinfacht und Ableitungen ein wenig verständlicher macht. Tatsächlich ist diese Notation die, die Sie in den bisherigen Videos gesehen haben. Der Einfachheit halber wird dieser Ausdruck hier manchmal nicht

vollständig aufgeschrieben, sondern manchmal auch als Ableitung oder partielle Ableitung von J in Bezug auf w_i gekürzt oder so geschrieben. Dies sind hier nur vereinfachte Kurzformen dieses Ausdrucks. Ich hoffe, das gibt Ihnen einen Eindruck davon, was Derivate sind. Es stellt sich nur die Frage, wie stark sich J von w als Folge davon ändert, wenn w um ein wenig ansteigt, bei Epsilon. Schauen wir uns als Nächstes an, wie Sie Ableitungen in einem neuronalen Netzwerk berechnen können. Dazu müssen wir uns etwas ansehen, das als Berechnungsdiagramm bezeichnet wird. Schauen wir uns das im nächsten Video an.

## Berechnungsdiagramm (optional)

Der Berechnungsgraph ist eine Schlüsselidee beim Deep Learning und dient auch dazu, wie Programmier-Frameworks wie TensorFlow automatisch Ableitungen Ihrer neuronalen Netze berechnen. Werfen wir einen Blick darauf, wie es funktioniert. Lassen Sie mich das Konzept eines Berechnungsgraphen anhand eines kleinen Beispiels eines neuronalen Netzwerks veranschaulichen. Dieses neuronale Netzwerk hat nur eine Schicht, die auch die Ausgabeschicht ist, und nur eine Einheit in der Ausgabeschicht. Es nimmt uns Eingaben x, wendet eine lineare Aktivierungsfunktion an und gibt Deaktivierung a aus. Genauer gesagt ist diese Ausgabe a gleich wx plus b. Dies ist im Grunde eine lineare Regression, wird jedoch als neuronales Netzwerk mit einer Ausgabeeinheit ausgedrückt. Angesichts der Ausgabe ist die Ursachenfunktion dann 1/ 2a, also der vorhergesagte Wert minus dem tatsächlich beobachteten Wert von y. Für dieses kleine Beispiel werden wir nur ein einziges Trainingsbeispiel haben, wobei das Trainingsbeispiel darin besteht, dass die Eingabe x gleich minus 2 ist. Der Ground-Truth-Ausgabewert y ist gleich 2 und die Parameter dieses Netzwerks sind; w ist gleich 2 und b ist gleich 8. Ich möchte zeigen, wie die Berechnung der Ursachenfunktion J Schritt für Schritt mithilfe eines Berechnungsgraphen berechnet werden kann. Zur Erinnerung: Beim Lernen betrachten wir die Ursachenfunktion J gerne als Funktion der Parameter w und b. Nehmen wir die Berechnung von J und zerlegen sie in einzelne Schritte. Zuerst haben Sie den Parameter w, der eine Eingabe für die Ursachenfunktion J ist, und dann müssen wir zuerst w mal x berechnen. Lassen Sie mich das c einfach wie folgt nennen; w ist gleich 2, x ist gleich -2 und c wäre also -4. Ich schreibe den Wert hier einfach über diesen Pfeil, um den Wert anzuzeigen, den Sie auf diesem Pfeil ausgeben. Der nächste Schritt besteht dann darin, a zu berechnen, das wx plus b ist. Lassen Sie mich hier einen weiteren Knoten erstellen. Dazu muss b eingegeben werden, der andere Parameter, der in die Ursachenfunktion J eingegeben wird, und a ist gleich wx plus b ist gleich c plus b. Wenn man diese addiert, ergibt das 4. Damit beginnt der Aufbau eines Berechnungsdiagramms, in dem die Schritte, die wir zur Berechnung der Ursachenfunktion J benötigen, in kleinere Schritte zerlegt werden. Der nächste Schritt besteht darin, ein minus y zu berechnen, das ich d nennen werde. Lassen Sie mich diesen Knoten d haben, der ein Minus y berechnet. Y ist gleich 2, also ist 4 minus 2 2. Dann schließlich ist J die Ursache und beträgt 1/2 minus y zum Quadrat oder 1/2 von d zum Quadrat, was genau gleich 2 ist. Was wir gerade haben Fertig ist die Erstellung eines Berechnungsdiagramms. Dies ist ein Graph, nicht im Sinne von Darstellungen mit x- und y-Achsen, sondern dies ist die andere Bedeutung des Wortes Graph in der Informatik, nämlich dass es sich um eine Menge von Knoten handelt, die durch Kanten oder durch Pfeile verbunden sind dieser Fall. Dieses Berechnungsdiagramm zeigt den Vorwärtsschritt, mit dem wir die Ausgabe a des neuronalen Netzwerks berechnen. Gehen Sie dann aber noch weiter und berechnen Sie auch den Wert der Ursachenfunktion J. Die Frage ist nun, wie finden wir die Ableitung von J nach den Parametern w und b? Schauen wir uns das als nächstes an. Hier ist das Berechnungsdiagramm aus der vorherigen Folie und wir haben es für ein Problem vervollständigt, bei dem wir berechnet haben, dass J, die Ursachenfunktion, durch alle diese Schritte von links nach rechts für eine Requisite im Berechnungsdiagramm gleich 2 ist. Was wir jetzt tun möchten, ist die Ableitung von J nach w und die

Ableitung von J nach b zu berechnen. Es stellt sich heraus, dass es sich bei einem Prop um eine Berechnung von links nach rechts handelte, bei der Berechnung der Ableitungen jedoch um eine Berechnung von rechts nach links, weshalb es Backprop genannt wird, also von rechts nach links rückwärts ging. Der letzte Berechnungsknoten dieses Diagramms ist dieser hier, der J gleich 1/2 von d im Quadrat berechnet. Der erste Schritt von Backprop fragt, ob sich der Wert von d, der die Eingabe in diesen Knoten war, ein wenig geändert hat. Wie stark ändert sich der Wert von j? Konkret möchte ich fragen, wie sich der Wert von j ändern würde, wenn d um ein wenig steigen würde, sagen wir 0,001, und das wäre in diesem Fall unser Wert von Epsilon. In diesem Fall stellt sich heraus, dass, wenn d von 2-2,01 reicht, j von 2-2,02 geht. Wenn also d um Epsilon steigt, steigt j um etwa das Zweifache von Epsilon. Wir schließen daraus, dass die Ableitung von J in Bezug auf diesen Wert d, der an diesem letzten Knoten eingegeben wird, gleich zwei ist. Der erste Schritt von Backprop wäre, diesen Wert zwei hier einzutragen, wobei dieser Wert die Ableitung von j in Bezug auf diesen Eingabewert d ist. Wir wissen, dass sich j um das Doppelte ändert, wenn sich d ein wenig ändert, da diese Ableitung gleich zwei ist. Der nächste Schritt besteht darin, den Knoten davor zu betrachten und zu fragen, was die Ableitung von j nach a ist. Um das zu beantworten, müssen wir fragen: Wenn a um 0,001 steigt, wie ändert sich dann j? Nun, wir wissen, dass d nur ein Minus von y ist, wenn a um 0,001 steigt. Wenn a 4,001 wird, d, das a minus y ist, zu 4,001 minus y gleich 2 wird, also 2,001 wird, steigt sub a um 0,001, d steigt ebenfalls um 0,001. Aber wir waren bereits zuvor zu dem Schluss gekommen, dass d um 0,001 steigt, j um das Doppelte. Jetzt wissen wir, wenn a um 0,001 steigt, d um 0,001 steigt und j ungefähr um das Doppelte von 0,001 steigt. Dies sagt uns, dass die Ableitung von j nach a ebenfalls gleich zwei ist. Deshalb werde ich diesen Wert hier eingeben. Dass dies die Ableitung von j nach a ist. Genauso wie dies die Ableitung von j nach d war. Wenn Sie schon einmal an einem Analysis-Kurs teilgenommen haben und von der Kettenregel gehört haben, erkennen Sie vielleicht, dass dieser Rechenschritt, den ich gerade durchgeführt habe, tatsächlich auf der Kettenregel für die Analysis beruht. Wenn Sie mit der Kettenregel nicht vertraut sind, machen Sie sich darüber keine Sorgen. Für den Rest dieser Videos müssen Sie es nicht wissen. Aber wenn Sie die Kettenregel gesehen haben, werden Sie vielleicht erkennen, dass die Ableitung von j nach a die Frage ist, wie stark sich d nach a ändert, was die Ableitung von d nach a mal der Ableitung von j nach d ist , und macht darüber hinaus wenig Berechnung, zeigte, dass der Teil von t nach a eins ist, und wir würden EZ zeigen, dass die Ableitung von J nach d gleich zwei ist, weshalb die Ableitung von J nach a ist eins mal zwei, was gleich zwei ist. Das ist der Wert, den wir haben. Aber auch hier gilt: Wenn Sie mit der Kettenregel nicht vertraut sind, machen Sie sich darüber keine Sorgen. Die Logik, die wir hier gerade durchgegangen sind, ist der Grund, warum wir wissen, dass j doppelt so stark steigt wie a. Deshalb ist dieser Ableitungsterm gleich zwei. Der nächste Schritt besteht dann darin, weiterhin von rechts nach links zu gehen, wie wir es beim Backprop tun. Wir fragen uns: Wie stark führt eine kleine Änderung in c dazu, dass sich j ändert, und wie stark führt eine Änderung von y in b dazu, dass sich j ändert? Um das herauszufinden, stellen wir uns die Frage: Was ändert sich, wenn c um Epsilon 0,001 steigt? Nun, a ist gleich c plus b. Es stellt sich heraus, dass, wenn c am Ende minus 3,999 ist, a, das minus 3,999 plus 8 ist, zu 4,001 wird. Wenn c um Epsilon steigt, steigt a um Epsilon. Wir wissen, dass, wenn a um Epsilon ansteigt, wir wissen, dass dies wiederum dazu führt, dass j um das Zweifache von Epsilon ansteigt, da die Ableitung von J nach a zwei beträgt. Wir können daraus schließen, dass J um das Doppelte steigt, wenn c ein wenig steigt. Wir wissen das, weil wir wissen, dass die Ableitung von J nach a 2 ist. Daraus können wir schließen, dass die Ableitung von J nach c ebenfalls gleich 2 ist. Ich werde diesen Wert hier eingeben. Auch hier gilt: Nur wenn Sie mit der Kettenregel vertraut sind, gibt es eine andere Möglichkeit, dies zu schreiben: Ableitung von J in Bezug auf c, ist die Ableitung von a in Bezug auf c. Es stellt sich heraus, dass dies das 1-fache der Ableitung von J nach a ist, von der wir zuvor herausgefunden haben, dass sie gleich 2 ist, weshalb sie am Ende gleich 2 ist. Durch eine ähnliche Berechnung erhöht sich b um 0,001 a steigt ebenfalls um 0,001 und J steigt um das Zweifache von 0,001, weshalb diese Ableitung auch gleich 2 ist. Wir haben hier die Ableitung von J

nach b und hier die Ableitung von J nach c eingetragen. Nun ein letzter Schritt: Was ist die Ableitung von J nach w? W steigt um 0,001. Was geschieht? C, das w mal x ist, wenn w 2,001 wäre, wird c, das w mal x ist, negativ 2 mal 2,001, also wird es negativ 4,002. Wenn w um Epsilon steigt, sinkt c um 2 mal 0,001, oder entsprechend steigt c um minus 2 mal 0,001. Wir wissen, dass, wenn c um minus 2 mal 0,001 steigt, weil die Ableitung von J nach c 2 ist, dies bedeutet, dass J um minus 4 mal 0,001 steigt, denn wenn c um einen bestimmten Betrag steigt, ändert sich J um das 2-fache, also negativ 2-mal so viel negativ ist 4-mal so viel. Daraus können wir schließen, dass, wenn w um 0,001 steigt, J um minus 4 mal 0,001 steigt. Die Ableitung von J nach w ist minus 4. Ich schreibe hier minus 4, weil die Ableitung von J nach w ist. Noch einmal, die Kettenregelberechnung ist, wenn Sie damit vertraut sind, folgende. Es ist die Ableitung von c nach w mal die Ableitung von J nach c. Das ist 2 und das ist minus 2, weshalb wir am Ende minus 4 haben, aber machen Sie sich darüber keine Sorgen, wenn Sie mit der Kettenregel nicht vertraut sind. Zum Abschluss dessen, was wir gerade getan haben, führen Sie in diesem Berechnungsdiagramm manuell Backprop aus. Während die Vorwärtsstütze eine Berechnung von links nach rechts war, bei der w gleich 2 war, konnten wir c berechnen. Dann hatten wir b und das erlaubt uns, a und dann d zu berechnen, und dann ging J backprop von rechts nach links und wir berechneten zuerst die Ableitung von J nach d und gingen dann zurück, um die Ableitung von J mit zu berechnen nach a, dann die Ableitung von J nach b, die Ableitung von J nach c und schließlich die Ableitung von J nach w. Deshalb ist Backprop eine Berechnung von rechts nach links, während Forward Prop eine Berechnung von links nach rechts war. Lassen Sie uns die Berechnung, die wir gerade durchgeführt haben, noch einmal überprüfen. J mit diesen Werten von w, b, x und y ist gleich der Hälfte mal wx plus b minus y im Quadrat, was der Hälfte mal 2 mal minus 2 plus 8 minus 2 im Quadrat entspricht, was gleich 2 ist. Nun, wenn Würde w um 0,001 steigen, dann wird J halbiert, w ist jetzt 2,001 mal x, was minus 2 ist, plus b, was 8 minus y im Quadrat ist. Wenn man das ausrechnet, ergibt sich ein Wert von 1,996002. Ungefähr ist J von 2 auf 1,996 gesunken, und dann noch einmal auf 002, und J ist somit um das Vierfache des Epsilons gesunken. Dies zeigt, dass, wenn W um Epsilon steigt, J um das Vierfache des Epsilon-Ball-Äquivalents sinkt und J um das negative Vierfache des Epsilon steigt, also y. Die Ableitung von j nach w ist minus 4, was wir hier ermittelt haben. Wenn Sie möchten, können Sie das Video gerne anhalten und diese Berechnung auch selbst noch einmal überprüfen, um herauszufinden, was in b passiert. Der andere Parameter steigt um Epsilon, und Sie werden hoffentlich feststellen, dass die Ableitung von j nach b tatsächlich ist zwei. Dass b um Epsilon steigt, j steigt um das Doppelte von Epsilon, wie durch diese Ableitungsrechnung vorhergesagt. Warum verwenden wir den Backprop-Algorithmus zur Berechnung von Ableitungen? Es stellt sich heraus, dass der Hintergrund eine effiziente Methode zur Berechnung von Derivaten ist. Der Grund, warum wir dies als Rechts-nach-Links-Berechnung anordnen, ist, wenn Sie zunächst fragen würden, was die Ableitung von j nach w ist. Um dann zu wissen, wie stark sich die Änderung von w auf die Änderung von j auswirkt: Wenn w um Epsilon steigen würde, um wie viel würde j um Epsilon steigen? Nun, das erste, was wir wissen wollen, ist: Was ist die Ableitung von j nach c? Weil eine Änderung von w diese erste Größe hier ändern wird. Um zu wissen, wie stark sich eine Änderung in w auf j auswirkt, möchten wir wissen, wie stark sich eine Änderung in c auf j auswirkt. Um jedoch zu wissen, wie stark sich eine Änderung in c auf j auswirkt, ist es am hilfreichsten zu wissen, wie sich die Änderung in c auf a auswirkt. Sie möchten wissen, wie stark sich dieser Effekt j usw. ändert. Aus diesem Grund wird eine Sequenz als Rechts-nach-Links-Berechnung zurückgesetzt. Denn wenn Sie die Rechnung von rechts nach links durchführen, können Sie herausfinden, wie sich die Änderung von d auf die Änderung von j auswirkt. Dann können Sie herausfinden, wie stark sich diese Änderung in einem Effekt j usw. auswirkt. Bis Sie die Ableitungen jeder dieser Zwischengrößen c, a und d sowie die Parameter w und b finden. Dass Sie mit einer Berechnung von rechts nach links herausfinden können, wie stark sich eine dieser Zwischengrößen c, a oder d sowie die Eingabeparameter w und b ändern. Wie viel Änderung in einem dieser Dinge wirkt sich auf den endgültigen Ausgabewert j aus? Eine Sache, die Backprop

effizient macht, ist, dass wir bei der Berechnung von rechts nach links diesen Term, die Ableitung von j nach a, nur einmal berechnen mussten. Diese Größe wird dann verwendet, um sowohl die Ableitung von g nach w als auch die Ableitung von j nach b zu berechnen. Es stellt sich heraus, dass wir in diesem Fall zwei Parameter haben, wenn ein Berechnungsgraph n Knoten hat, also Ihre n dieser Felder und p Parameter. Dieses Verfahren ermöglicht es uns, alle Ableitungen von j in Bezug auf alle Parameter in ungefähr n plus p Schritten zu berechnen, statt in n mal p Schritten. Wenn Sie ein neuronales Netzwerk mit beispielsweise 10.000 Knoten und vielleicht 100.000 Parametern haben. Nach modernen Maßstäben wäre dies nicht einmal ein sehr großes neuronales Netzwerk. In der Lage zu sein, die Ableitungen und 10.000 plus 100.000 Schritte zu berechnen, was Punch und 10.000 ist, ist viel besser, als 10.000 mal 100.000 Schritte zu erfüllen, was einer Milliarde Schritte entsprechen würde. Der Backpropagation-Algorithmus, der mithilfe des Berechnungsdiagramms erstellt wird, bietet Ihnen eine sehr effiziente Möglichkeit, alle Ableitungen zu berechnen. Deshalb ist es ein so zentraler Gedanke bei der heutigen Implementierung von Deep-Learning-Algorithmen. In diesem Video haben Sie gesehen, wie das Berechnungsdiagramm alle Schritte der Berechnung ausführt, die zur Berechnung der Ausgabe eines neuronalen Netzwerks a sowie der Kostenfunktion j erforderlich sind. Führt Schritt-für-Schritt-Berechnungen durch und unterteilt sie in die verschiedenen Knoten des Berechnungsdiagramms. Anschließend wird eine Links-nach-rechts-Berechnung für eine Requisite verwendet, um die Kostenfunktion J zu berechnen. Anschließend wird eine Rechts-nach-links- oder Backpropagation-Berechnung zur Berechnung aller Ableitungen verwendet. In diesem Video haben Sie gesehen, wie diese Ideen auf ein kleines Beispiel eines neuronalen Netzwerks angewendet werden. Lassen Sie uns im nächsten Video diese Ideen aufgreifen und auf ein größeres neuronales Netzwerk anwenden. Kommen wir zum nächsten Video.

## Beispiel für ein größeres neuronales Netzwerk (optional)

In diesem letzten Video zur Intuition für Backprop werfen wir einen Blick darauf, wie der Berechnungsentwurf an einem größeren Beispiel eines neuronalen Netzwerks funktioniert. Hier ist das Netzwerk, das wir mit einer einzelnen verborgenen Schicht verwenden werden, mit einer einzelnen verborgenen Einheit, die a1 ausgibt und die in die Ausgabeschicht einspeist, die die endgültige Vorhersage a2 ausgibt. Um die Mathematik verständlicher zu machen, werde ich weiterhin nur ein einziges Trainingsbeispiel mit den Eingaben x = 1, y = 5 verwenden. Und das werden die Parameter des Netzwerks sein. Und wir werden durchgehend die ReLU-Aktivierungsfunktionen von g( z) = max(0, z) verwenden. Die Requisite in Ihrem Netzwerk sieht also so aus. Wie üblich ist a1 gleich g( w1 mal x + b1). Und so stellt sich heraus, dass w1x + b positiv sein wird. Wir befinden uns also in den max( 0, z) = z, Teilen dieser Aktivierungsfunktion. Das ist also gleich 2 mal 1, das ist w1 ist 2 mal x1 + 0, das ist b1, was gleich 2 ist. Und dann ist a2 gleich diesem, g(w2a1 + b2), also w2 mal a1 + b. Auch hier, weil wir uns im positiven Teil der ReLU-Aktivierungsfunktion befinden, der 3 x 2 + 1 = 7 ist. Zum Schluss verwenden wir die quadratische Fehlerkostenfunktion. Also ist j( w, b) 1/2(a2- y) im Quadrat = 1/2(7-5) im Quadrat, was 1/2 von 2 im Quadrat ist, was genau gleich 2 ist. Nehmen wir also diese Berechnung das haben wir gerade gemacht und schreiben es in Form eines Berechnungsdiagramms auf. Um die Berechnung Schritt für Schritt durchzuführen, müssen wir zunächst w1 nehmen und mit x multiplizieren. Wir haben also w1, das in den Berechnungsknoten eingespeist wird, der w1 mal x berechnet. Und ich werde dies eine temporäre Variable t1 nennen. Als nächstes berechnen wir z1, was hier dieser Term ist, der t1 + b1 ist. Wir haben hier also auch diesen Eingang b1. Und schließlich ist a1 gleich g( z1). Wir wenden die Aktivierungsfunktion an und erhalten hier wieder den Wert 2. Als nächstes müssen wir t2 berechnen, also w2 mal a1. Und mit w2 erhalten wir diesen Wert, der 6 ist. Dann mussten wir z2, das ist diese

Größe, mit b2 versehen, und das ergibt 7. Und schließlich wenden wir die Aktivierungsfunktion g an. Am Ende haben wir immer noch 7. Und schließlich ist j 1/2(a2-y) im Quadrat. Und das ergibt 2. Das war hier diese Kostenfunktion. So führen Sie die Berechnungen für größere neuronale Netzwerke Schritt für Schritt durch und schreiben sie in das Berechnungsdiagramm. Sie haben bereits im letzten Video die Mechanik zur Durchführung von Backprop gesehen. Ich werde die Berechnungen hier nicht Schritt für Schritt durchgehen. Wenn Sie jedoch Backprop durchführen würden, fragen Sie sich zunächst: Was ist die Ableitung der Kostenfunktion j nach a2? Und wenn man das ausrechnet, ergibt sich ein Wert von 2. Also tragen wir das hier ein. Und im nächsten Schritt wird gefragt, was die Ableitung der Kosten j nach z2 ist . Und mithilfe dieser Ableitung, die wir zuvor berechnet haben, können Sie herausfinden, dass dies 2 ist. Denn wenn z um Epsilon steigt, können Sie zeigen, dass a2 bei der aktuellen Einstellung aller Parameter um Epsilon steigt. Und deshalb wird j um das 2-fache Epsilon steigen. Diese Ableitung ist also gleich 2 und so weiter. Schritt für Schritt. Wir können dann herausfinden, dass die Ableitung von j bzw. b2 ebenfalls gleich 2 ist. Die Ableitung nach t2 ist gleich 2 und so weiter und so weiter. Bis Sie schließlich die Ableitung von j nach allen Parametern w1, b1, w2 und b2 berechnet haben. Und das ist Backprop. Und wieder einmal habe ich nicht die mechanischen Schritte jedes einzelnen Backprop-Schritts durchgegangen. Aber es ist im Grunde der Prozess, den Sie im vorherigen Video gesehen haben. Lassen Sie mich eines dieser Beispiele noch einmal überprüfen. Wir haben hier also gesehen, dass die Ableitung von j in Bezug auf w1 gleich 6 ist. Dies sagt also voraus, dass, wenn w1 um Epsilon steigt, j um ungefähr das Sechsfache von Epsilon steigen sollte. Lassen Sie uns die Karte durchgehen und sehen, ob das wirklich stimmt. Dies sind wiederum die Berechnungen, die wir durchgeführt haben. Wenn also w, das 2 war, 2,001 wäre und um Epsilon steigt, dann wird a1, mal sehen, anstelle von 2 ist dies ebenfalls 2,001. Also ist a1 statt 2 jetzt 2,001. Also 3 x 2,001 + 1, das ergibt 7,003. Und wenn a2 7,003 ist, dann wird es einfach 7,003-5 zum Quadrat. Daraus ergibt sich also 2,003 zum Quadrat von 2, was 2,006005 ergibt. Wenn man also einige der zusätzlichen Ziffern außer Acht lässt, sieht man aus dieser kleinen Rechnung, dass, wenn w1 um 0,001 steigt, j von w ungefähr von 2 auf 2,006 gestiegen ist. Also 6 mal so viel. Die Ableitung von j nach w1 ist also tatsächlich gleich 6. Das Backprop-Verfahren bietet Ihnen also eine sehr effiziente Möglichkeit, alle diese Ableitungen zu berechnen. Diese können Sie dann in den Gradientenabstiegsalgorithmus oder den Adam-Optimierungsalgorithmus einspeisen, um dann die Parameter Ihres neuronalen Netzwerks zu trainieren. Und noch einmal: Der Grund dafür, dass wir den Hintergrund verwenden, ist, dass er eine sehr effiziente Möglichkeit ist, alle Ableitungen von j in Bezug auf w1, j in Bezug auf b1, j in Bezug auf w2 und j in Bezug auf b2 zu berechnen. Ich habe gerade gezeigt, wie wir w1 ein wenig erhöhen und sehen können, wie sehr sich j ändert. Aber das war eine Berechnung von links nach rechts. Und dann mussten wir dieses Verfahren für jeden Parameter einzeln durchführen. Wenn wir w um 0,001 erhöhen müssten, um zu sehen, wie sich dadurch j ändert. Erhöhen Sie b1 ein wenig, um zu sehen, wie sich j dadurch ändert, und erhöhen Sie jeden Parameter einzeln um ein wenig, um zu sehen, wie sich j dadurch ändert. Dann wird dies zu einer sehr ineffizienten Berechnung. Und wenn Sie N Knoten in Ihrem Berechnungsdiagramm und P Parameter hätten, würde dieses Verfahren am Ende N mal P Schritte erfordern, was sehr ineffizient ist. Wohingegen wir alle vier dieser Ableitungen N + P und nicht N mal P Schritte erhalten. Und das macht einen großen Unterschied in praktischen neuronalen Netzen, wo die Anzahl der Knoten und die Anzahl der Parameter sehr groß sein können. Das ist also das Ende des Videos für diese Woche. Vielen Dank, dass Sie bis zum Ende dieser optionalen Videos bei mir geblieben sind. Und ich hoffe, dass Sie jetzt ein Gespür dafür haben, wann Sie Programm-Frameworks wie Tensorflow verwenden, um ein neuronales Netzwerk zu trainieren. Was passiert eigentlich unter der Haube und wie nutzt man den Berechnungsgraphen, um Ableitungen effizient für Sie zu berechnen? Vor vielen Jahren, vor dem Aufkommen von Frameworks wie Tensorflow und Pytorch, mussten Forscher die Ableitungen der neuronalen Netze, die sie trainieren wollten, manuell mithilfe von Analysis berechnen. Und so können Sie in modernen Programm-Frameworks „forwardprop“ angeben und Backprop für Sie

übernehmen lassen. Vor vielen Jahren schrieben Forscher das neuronale Netzwerk manuell auf und berechneten die Ableitungen manuell mithilfe von Analysis. Und dann implementierten die Neuronen eine Reihe von Gleichungen, die sie mühsam auf dem Papier abgeleitet hatten, um Backprop zu implementieren. Dank des Berechnungsgraphen und dieser Techniken zur automatischen Durchführung von Ableitungsberechnungen. Wird manchmal Autodiff genannt, was automatische Differenzierung bedeutet. Dieser Prozess, bei dem Forscher manuell Kalkulationen verwenden, um Ableitungen zu ermitteln, wird nicht mehr wirklich durchgeführt. Zumindest musste ich das seit vielen Jahren nicht mehr selbst tun, dank Autodiff. Vor vielen Jahren lag die Messlatte für den Rechenaufwand, den man tatsächlich kennen muss, bei der Nutzung neuronaler Netze höher. Aber dank automatischer Differenzierungsalgorithmen, die normalerweise auf dem Berechnungsgraphen basieren, können Sie jetzt ein neuronales Netzwerk implementieren und Ableitungen einfacher als zuvor für Sie berechnen lassen. Vielleicht ist mit der Weiterentwicklung neuronaler Netze die Menge an Berechnungen, die Sie wissen müssen, damit diese Algorithmen funktionieren, tatsächlich zurückgegangen. Und das hat vielen Menschen Mut gemacht. Und damit sind es die Videos für diese Woche. Ich hoffe, dass Ihnen die Labore gefallen und freue mich darauf, Sie nächste Woche wiederzusehen.

## Überlegen Sie, was Sie als Nächstes versuchen möchten

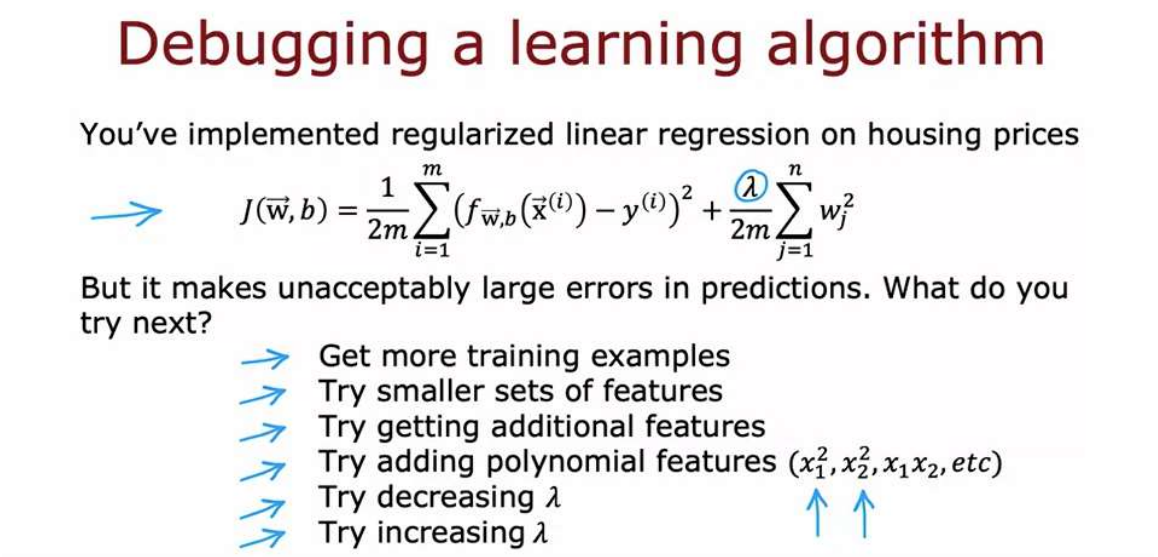
### Learning Objectives

---

- Evaluate and then modify your learning algorithm or data to improve your model's performance
- Evaluate your learning algorithm using cross validation and test datasets.
- Diagnose bias and variance in your learning algorithm
- Use regularization to adjust bias and variance in your learning algorithm
- Identify a baseline level of performance for your learning algorithm
- Understand how bias and variance apply to neural networks
- Learn about the iterative loop of Machine Learning Development that's used to update and improve a machine learning model
- Learn to use error analysis to identify the types of errors that a learning algorithm is making
- Learn how to add more training data to improve your model, including data augmentation and data synthesis
- Use transfer learning to improve your model's performance.
- Learn to include fairness and ethics in your machine learning model development
- Measure precision and recall to work with skewed (imbalanced) datasets

Hallo und willkommen zurück. Mittlerweile haben Sie viele verschiedene Lernalgorithmen gesehen, darunter lineare Regression, logistische Regression, sogar Deep Learning oder neuronale Netze, und nächste Woche werden Sie auch Entscheidungsbäume sehen. Sie verfügen mittlerweile über viele leistungsstarke Werkzeuge des maschinellen Lernens, aber wie nutzen Sie diese Werkzeuge effektiv? Ich habe manchmal Teams gesehen, die, sagen wir, sechs Monate Zeit hatten, um ein System für maschinelles Lernen aufzubauen, von denen ich denke, dass ein kompetenteres Team sie in nur ein paar Wochen hätte übernehmen oder schaffen können. Die Effizienz, mit der Sie ein maschinelles Lernsystem schnell zum Laufen bringen können, hängt zu einem großen Teil davon ab, wie gut Sie im

Laufe eines maschinellen Lernprojekts immer wieder gute Entscheidungen darüber treffen können, was als nächstes zu tun ist. In dieser Woche möchte ich Ihnen eine Reihe von Tipps geben, wie Sie Entscheidungen darüber treffen können, was als nächstes in einem maschinellen Lernprojekt zu tun ist, von denen ich hoffe, dass sie Ihnen am Ende viel Zeit sparen. Werfen wir einen Blick auf einige Ratschläge zum Aufbau maschineller Lernsysteme.



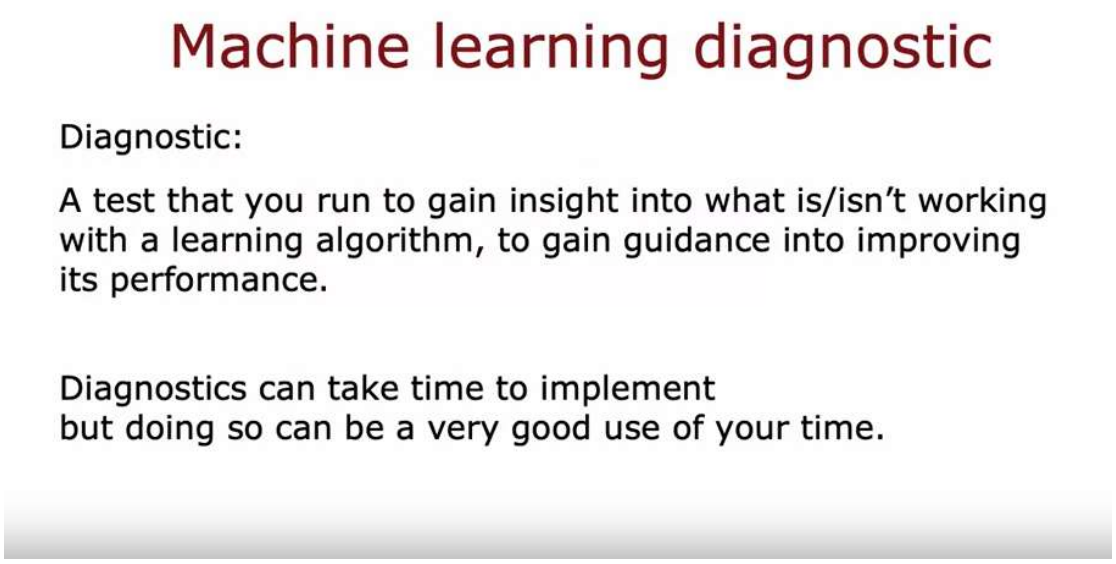

Beginnen wir mit einem Beispiel: Angenommen, Sie haben eine regulierte lineare Regression implementiert, um Immobilienpreise vorherzusagen. Sie haben also die übliche Kostenfunktion für Ihren Lernalgorithmus, den quadrierten Fehler plus diesen Regularisierungsterm. Aber wenn Sie das Modell trainieren und feststellen, dass es in seinen Vorhersagen unannehmbar große Fehler macht, was versuchen Sie dann als Nächstes? Wenn Sie einen Algorithmus für maschinelles Lernen erstellen, können Sie normalerweise viele verschiedene Dinge ausprobieren.

Beispielsweise könnten Sie sich dafür entscheiden, mehr Trainingsbeispiele zu erhalten, da es den Anschein hat, dass mehr Daten helfen sollten, oder vielleicht denken Sie, dass Sie vielleicht zu viele Funktionen haben, also könnten Sie einen kleineren Satz an Funktionen ausprobieren. Oder vielleicht möchten Sie zusätzliche Funktionen erhalten, wie zum Beispiel endlich zusätzliche Eigenschaften der Häuser, die Sie in Ihre Daten einfügen können, und vielleicht hilft Ihnen das dabei, es besser zu machen. Oder Sie nehmen die vorhandenen Merkmale x_1, x_2 usw. und versuchen, die Polynommerkmale x_1 im Quadrat, x_2 im Quadrat, x_1, x_2 usw. hinzuzufügen. Oder Sie fragen sich vielleicht, ob der Wert von Lambda gut gewählt ist, und Sie könnten sagen, vielleicht ist er zu groß, ich möchte ihn verringern. Oder Sie sagen vielleicht, es ist zu klein, ich möchte versuchen, es zu vergrößern. Bei jeder Anwendung für maschinelles Lernen stellt sich häufig heraus, dass einige dieser Dinge fruchtbar sein könnten, andere jedoch nicht.

Der Schlüssel zur effektiven Entwicklung eines Algorithmus für maschinelles Lernen liegt darin, dass Sie eine Möglichkeit finden, gute Entscheidungen darüber zu treffen, wo Sie Ihre Zeit investieren. Ich habe zum Beispiel gesehen, dass Teams buchstäblich viele, viele Monate damit verbracht haben, weitere Trainingsbeispiele zu sammeln, weil sie dachten, dass mehr Trainingsdaten helfen würden, aber manchmal stellte sich heraus, dass es sehr hilfreich war, und manchmal nicht.

In dieser Woche erfahren Sie, wie Sie eine Reihe von Diagnosen durchführen. Mit Diagnose meine ich einen Test, den Sie durchführen können, um Erkenntnisse darüber zu gewinnen, was mit dem Lernalgorithmus funktioniert und was nicht, um Hinweise zur Verbesserung seiner Leistung zu erhalten. Einige dieser Diagnosen verraten Ihnen beispielsweise, ob es sich lohnt, wochen- oder

sogar monatelang mehr Trainingsdaten zu sammeln, denn wenn ja, können Sie die Investition in mehr Daten tätigen, was hoffentlich zu einer verbesserten Leistung führt , oder wenn dies nicht der Fall ist, hätte Ihnen die Ausführung dieser Diagnose möglicherweise Monate Zeit gespart. Eine Sache, die Sie diese Woche ebenfalls sehen, ist, dass die Implementierung von Diagnosen einige Zeit in Anspruch nehmen kann, aber ihre Ausführung kann eine sehr gute Zeitnutzung sein.

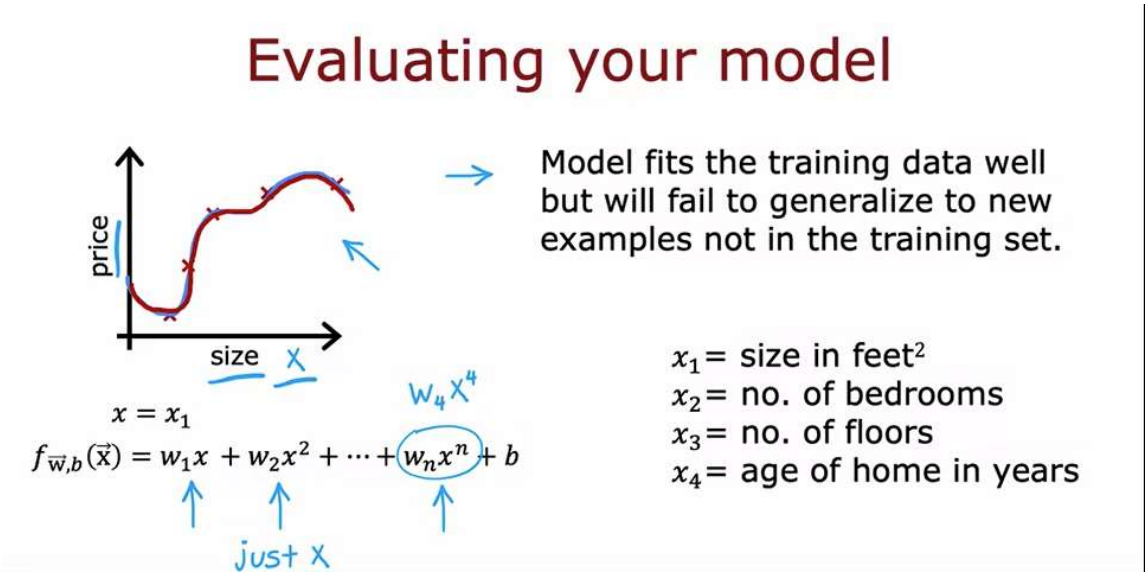



Diese Woche werden wir viel Zeit damit verbringen, über verschiedene Diagnosemethoden zu sprechen, die Sie verwenden können, um Ihnen Anleitungen zu geben, wie Sie die Leistung Ihres Lernalgorithmus verbessern können. Aber werfen wir zunächst einen Blick darauf, wie Sie die Leistung Ihres Lernalgorithmus bewerten können. Machen wir das im nächsten Video.

## Bewertung eines Modells

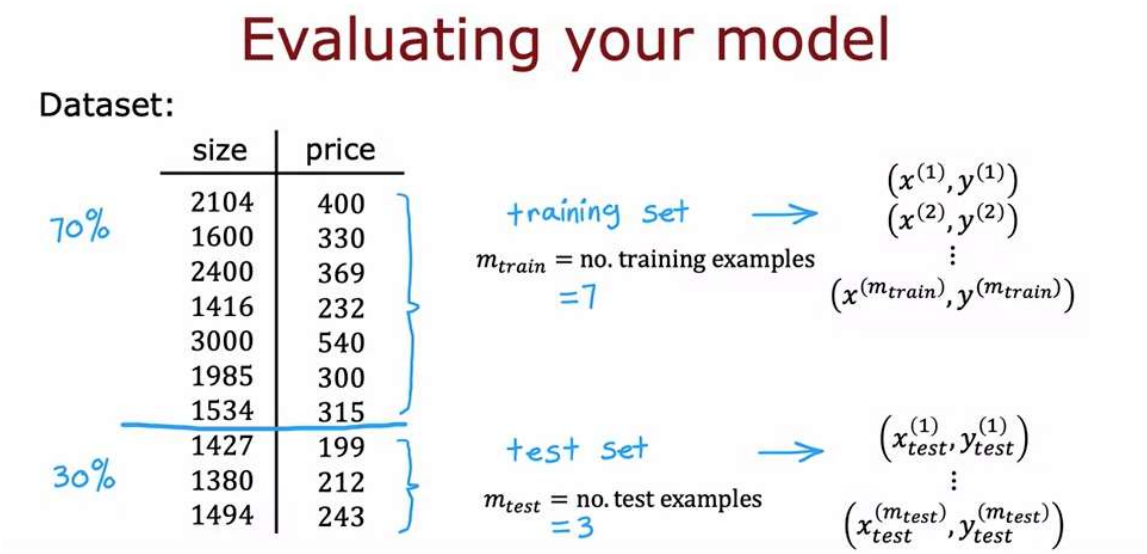
Mal sehen, Sie haben ein Modell für maschinelles Lernen trainiert. Wie bewerten Sie die Leistung dieses Modells? Sie stellen fest, dass eine systematische Methode zur Leistungsbewertung auch einen klareren Weg zur Verbesserung seiner Leistung aufzeigen wird. Schauen wir uns also an, wie das Modell bewertet wird. Nehmen wir das Beispiel des Lernens, Immobilienpreise als Funktion der Größe vorherzusagen. Nehmen wir an, Sie haben das Modell darauf trainiert, Immobilienpreise als Funktion der Größe x vorherzusagen.

Und für das Modell ist das ein Polynom vierter Ordnung. Also Funktionen x, x quadriert, ausgeführt und x zur 4. Da wir ein Polynom 1/4-Ordnung an einen Trainingssatz mit fünf Datenpunkten anpassen, passt dies sehr gut zu den Trainingsdaten. Aber dieses Modell gefällt uns nicht besonders, denn obwohl das Modell gut zu den Trainingsdaten passt, gehen wir davon aus, dass es sich nicht auf neue Beispiele verallgemeinern lässt, die nicht im Trainingssatz enthalten sind.




Wenn Sie also Preise vorhersagen, also nur ein einzelnes Merkmal der Größe des Hauses, könnten Sie das Modell wie folgt darstellen und wir könnten sehen, dass die Kurve sehr schwach ist, sodass wir wissen, dass diese Parodie kein gutes Modell ist. Aber wenn Sie dieses Modell mit noch mehr Funktionen ausstatten würden, sagen wir, wir hätten x1 die Größe des Hauses, die Anzahl der Schlafzimmer, die Anzahl der Stockwerke des Hauses und auch das Alter des Hauses in Jahren, dann wird es viel schwieriger, f zu plotten weil f nun eine Funktion von x1 bis x4 ist.

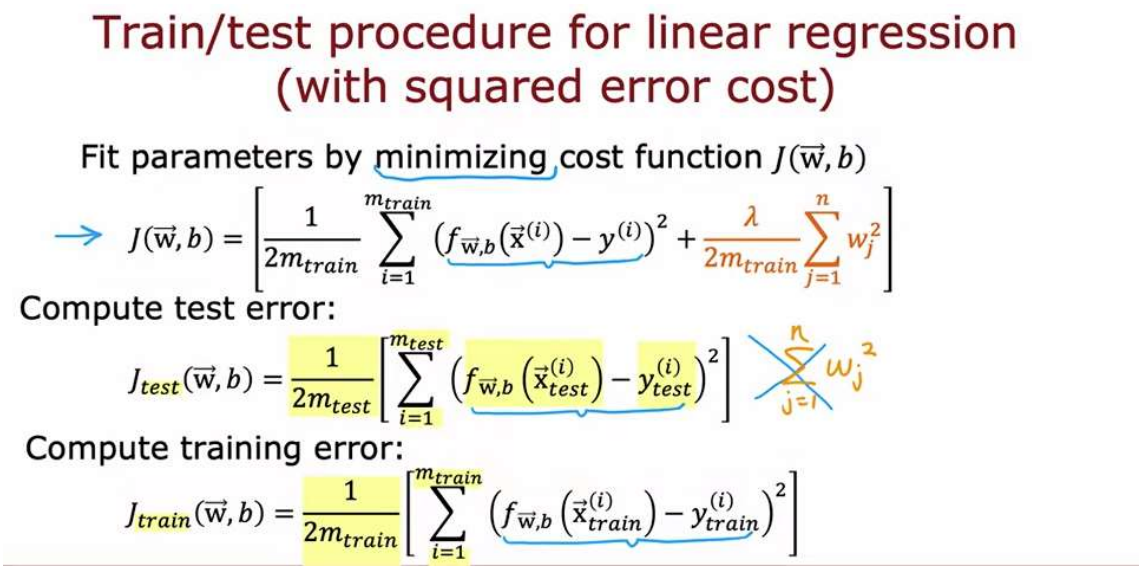
Und wie zeichnet man eine vierdimensionale Funktion? Um also festzustellen, ob Ihr Modell gut funktioniert, insbesondere bei Anwendungen, bei denen Sie mehr als ein oder zwei Features haben, was es schwierig macht, f von x darzustellen. Wir brauchen eine systematischere Methode, um zu bewerten, wie gut Ihr Modell funktioniert. Hier ist eine Technik, die Sie verwenden können. Wenn Sie über einen Trainingssatz verfügen und es sich um einen kleinen Trainingssatz mit nur 10 hier aufgeführten Beispielen handelt, können Sie den Trainingssatz stattdessen in zwei Teilmengen aufteilen, anstatt alle Ihre Daten zum Trainieren der Parameter w und p des Modells zu verwenden. Ich werde hier eine Linie ziehen und 70 % der Daten in den ersten Teil einfügen, den ich Trainingssatz nennen werde. Und der zweite Teil der Daten, sagen wir 30 % der Daten, die ich hineinstecke, ist festgelegt.



Und was wir tun werden, ist, die Modelle und Parameter des Trainingssatzes auf diesen ersten etwa 70 % der Daten zu trainieren, und dann werden wir seine Leistung auf diesem Testsatz testen.

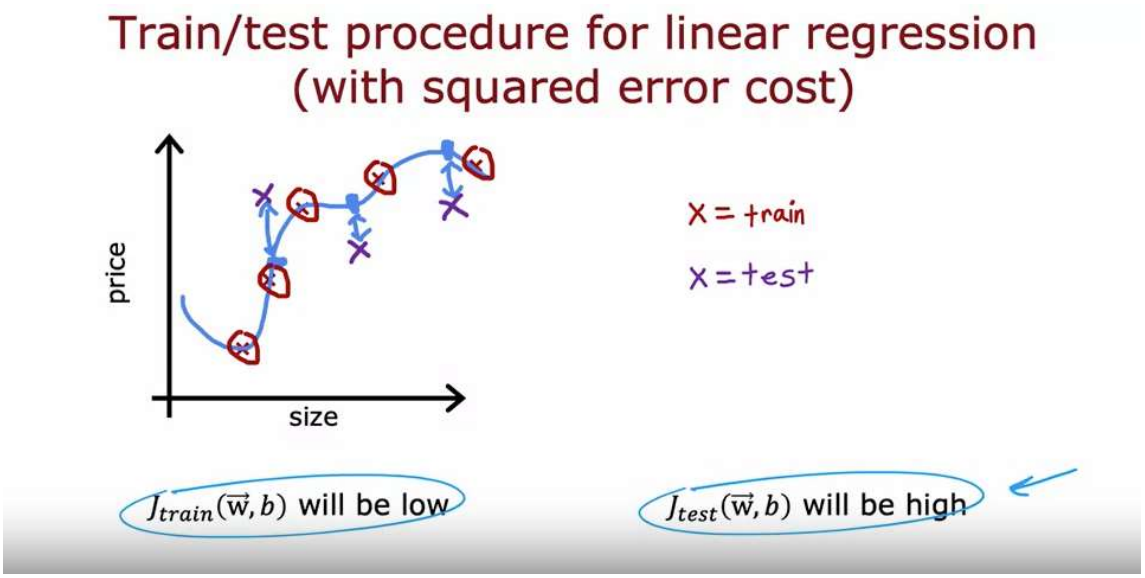
In der Notation verwende ich x1, y1? Dasselbe wie zuvor, um die Trainingsbeispiele durch xm, ym zu bezeichnen, außer dass dies jetzt explizit gemacht wird. In diesem kleinen Beispiel hätten wir also sieben Trainingsbeispiele. Und um eine neue Notation einzuführen, werde ich m subscript train verwenden. Der M-Zug ist eine Anzahl von Trainingsbeispielen, die in diesem kleinen Datensatz 7 beträgt. Der tiefgestellte Zug betont also nur, wenn wir den Trainingssatzteil der Daten betrachten. Und für den Testsatz verwende ich die Notation x1 subscript test, Komma y1, subscript test, um das erste Testbeispiel zu bezeichnen, und das geht bis hin zu x mtest subscript tests, y mtest subscript tests und m tests Anzahl der Testbeispiele, in diesem Fall 3. Und es ist nicht ungewöhnlich, Ihren Datensatz nach einer 70, 30-Aufteilung oder 80, 20-Aufteilung aufzuteilen, wobei die meisten Ihrer Daten in den Trainingssatz gehen und dann ein kleinerer Bruchteil in das Testset ein.

Um ein Modell zu trainieren und auszuwerten, würde es also so aussehen, wenn Sie eine lineare Regression mit quadrierten Fehlerkosten verwenden. Beginnen Sie mit der Anpassung der Parameter durch Minimierung der Kostenfunktion j von w ,b . Dies ist also die übliche Kostenfunktion, die über w,b dieser quadratischen Fehlerkosten minimiert wird, plus einem Regularisierungsterm, der länger als 2m ist und einen Teil des w,j-Quadrats darstellt. Und um dann festzustellen, wie gut dieses Modell abschneidet, würden Sie J test von w ,b berechnen , was dem durchschnittlichen Fehler im Testsatz entspricht, und das entspricht gerade mal der Hälfte des m-Tests. Das ist die Anzahl der Testbeispiele. Und dann sind von einigen insgesamt die Beispiele von r gleich 1, bis hin zur Anzahl der Testbeispiele des quadrierten Zeitalters auf jedem der Testbeispiele wie folgt. Es handelt sich also um eine Vorhersage über die Eingabe des Testbeispiels abzüglich des tatsächlichen Preises des Hauses im Quadrat des Testbeispiels. Und beachten Sie, dass die Testfehlerformel J test diesen Regularisierungsterm nicht enthält.



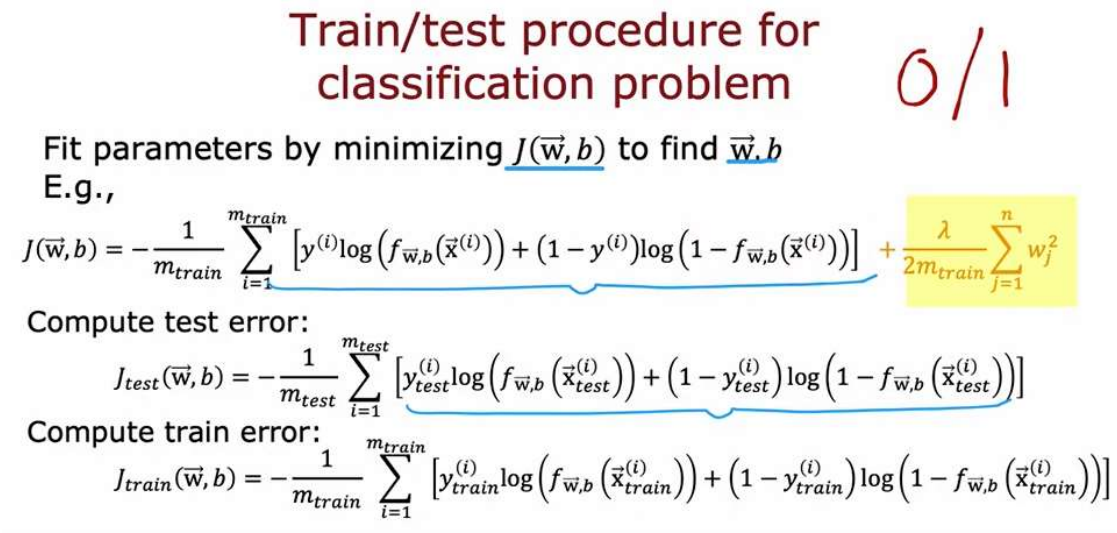

Dadurch erhalten Sie einen Eindruck davon, wie gut Ihr Lernalgorithmus funktioniert. Eine der für den Computer häufig nützlichen Größen sowie der Trainingsfehler, der ein Maß dafür ist, wie gut Ihr Lernalbum auf dem Trainingssatz abschneidet. Lassen Sie mich also den J-Zug von w ,b so definieren , dass er dem Durchschnitt über den Trainingssatz entspricht. 1 bis 2 m oder 1/2 m tiefgestellter Zug von einigen über Ihren Trainingssatz dieses quadrierten Fehlerterms. Und auch hier ist der Regularisierungsterm im Gegensatz zur Kostenfunktion, die Sie minimieren, um sie an die Parameter anzupassen, nicht enthalten. In dem Modell, wie wir es weiter oben in diesem Video gesehen haben, wird der J-Zug von w ,b also niedrig sein, da die durchschnittliche Ära in Ihren Trainingsbeispielen

Null oder sehr nahe bei Null liegt. Der J-Zug wird also sehr nahe bei Null liegen. Aber wenn Sie ein paar zusätzliche Beispiele in Ihrem Testset haben, auf die das Album nicht trainiert wurde, dann diese Testbeispiele, mein Liebesleben diese. Und es gibt eine große Lücke zwischen dem, was das Album als geschätzten Immobilienpreis vorhersagt, und dem tatsächlichen Wert dieser Immobilienpreise.




Daher werden die J-Tests hoch sein. Wenn Sie also sehen, dass der J-Test bei diesem Modell hoch ist, können Sie erkennen, dass er zwar im Trainingssatz gut abschneidet, sich aber nicht so gut auf neue Beispiele und neue Datenpunkte verallgemeinern lässt, die nicht im Trainingssatz enthalten waren.

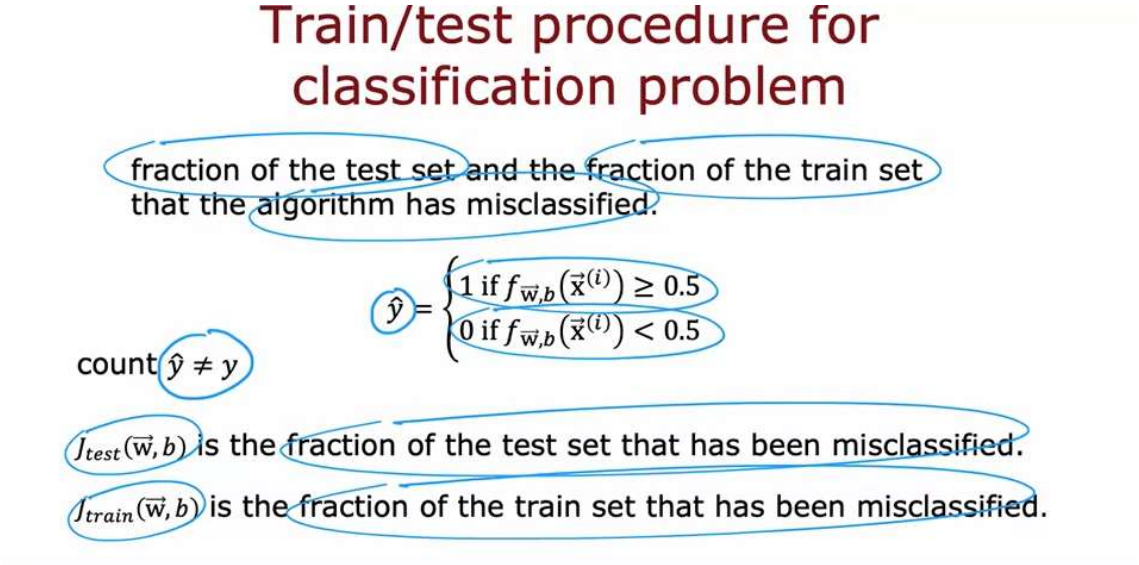
Das war also eine Regression mit quadrierten Fehlerkosten. Schauen wir uns nun an, wie Sie dieses Verfahren auf ein Klassifizierungsproblem anwenden. Wenn Sie beispielsweise zwischen handgeschriebenen Ziffern klassifizieren, die entweder 0 oder 1 sind, passen Sie die Parameter wie zuvor an, indem Sie die Kostenfunktion minimieren, um die Parameter w,b zu finden.




Wenn Sie beispielsweise die logistische Regression trainieren würden, wäre dies die Kostenfunktion J von w ,b , wobei dies die übliche logistische Verlustfunktion ist, und dann noch der Regularisierungsterm. Und um den Testfehler zu berechnen, ist J test dann der Durchschnitt Ihrer Testbeispiele, das heißt die 30 % Ihrer Daten, die nicht im Trainingssatz des Logistikverlusts Ihres Testsatzes enthalten waren. Und der Trainingsfehler, den Sie auch mit dieser Formel berechnen

können, ist der durchschnittliche logistische Verlust Ihrer Trainingsdaten, den das Album verwendet hat, um die Kostenfunktion J von w, b zu minimieren. Nun, wenn ich es hier beschreibe, wird es funktionieren, okay, um herauszufinden, ob Ihr Lernalgorithmus gut funktioniert, indem ich sehe, wie ich in Bezug auf Testfehler abgeschnitten habe.

Wenn man maschinelles Lernen auf Klassifizierungsprobleme anwendet, gibt es tatsächlich eine andere Definition von J-Tests und J-Train, die vielleicht sogar noch häufiger verwendet wird. Das heißt, anstatt den logistischen Verlust zur Berechnung des Testfehlers und des Trainingsfehlers zu verwenden, um stattdessen zu messen, wie groß der Anteil des Testsatzes und der Anteil des Trainingssatzes ist, den der Algorithmus falsch klassifiziert hat. Speziell für den Testsatz können Sie den Algorithmus also für jedes Testbeispiel eine Vorhersage 1 oder 0 treffen lassen. Erinnern Sie sich also daran, dass wir 1 vorhersagen würden, wenn f von x größer als 0,5 ist, und Null, wenn es kleiner als 0,5 ist. Und Sie können dann im Testsatz den Anteil der Beispiele zählen, bei denen y_hat im Testsatz nicht mit der tatsächlichen Grundwahrheitsbezeichnung übereinstimmt.

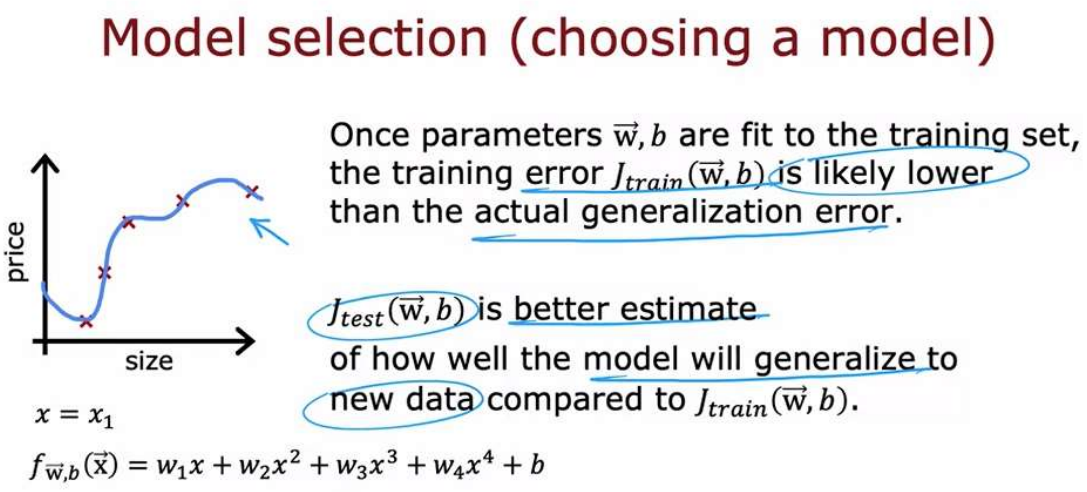



Wenn Sie also konkret die handgeschriebenen Ziffern 0, 1 durch einen neuen Klassifizierungswurf klassifizieren, dann wären J-Tests der Bruchteil dieses Testsatzes, wobei 0 als 1 von 1 klassifiziert wurde, also als 0 klassifiziert wurde. Und ähnlich ist J-Zug ein Bruchteil des Trainingssatzes, der falsch klassifiziert wurde. Durch die Aufteilung eines Datensatzes in einen Trainingssatz und einen separaten Testsatz können Sie systematisch bewerten, wie gut Ihre Lernergebnisse sind. Indem Sie sowohl J-Tests als auch J-Train berechnen, können Sie nun messen, wie es mit dem Testsatz und dem Trainingssatz abgeschnitten hat.

Dieses Verfahren ist ein Schritt dazu, wie Sie automatisch auswählen können, welches Modell für eine bestimmte Anwendung für maschinelles Lernen verwendet werden soll. Wenn Sie beispielsweise versuchen, Immobilienpreise vorherzusagen, sollten Sie dann eine gerade Linie an Ihre Daten anpassen oder ein Polynom zweiter Ordnung oder ein Polynom dritter Ordnung vierter Ordnung? Es stellt sich heraus, dass Sie mit einer weiteren Verfeinerung der Idee, die Sie in diesem Video gesehen haben, in der Lage sein werden, einen Algorithmus zu verwenden, der Ihnen dabei hilft, diese Art von Entscheidung automatisch gut zu treffen. Sehen wir uns im nächsten Video an, wie das geht.

## Modellauswahl und Training/Kreuzvalidierung/Testsätze

Im letzten Video haben Sie gesehen, wie Sie mit dem Testsatz die Leistung eines Modells bewerten. Lassen Sie uns diese Idee in diesem Video noch weiter verfeinern, damit Sie die Technik nutzen können, um automatisch ein gutes Modell für Ihren maschinellen Lernalgorithmus auszuwählen. Eine Sache, die wir gesehen haben, ist, dass die Parameter w und b des Modells einmal an den Trainingssatz angepasst wurden.



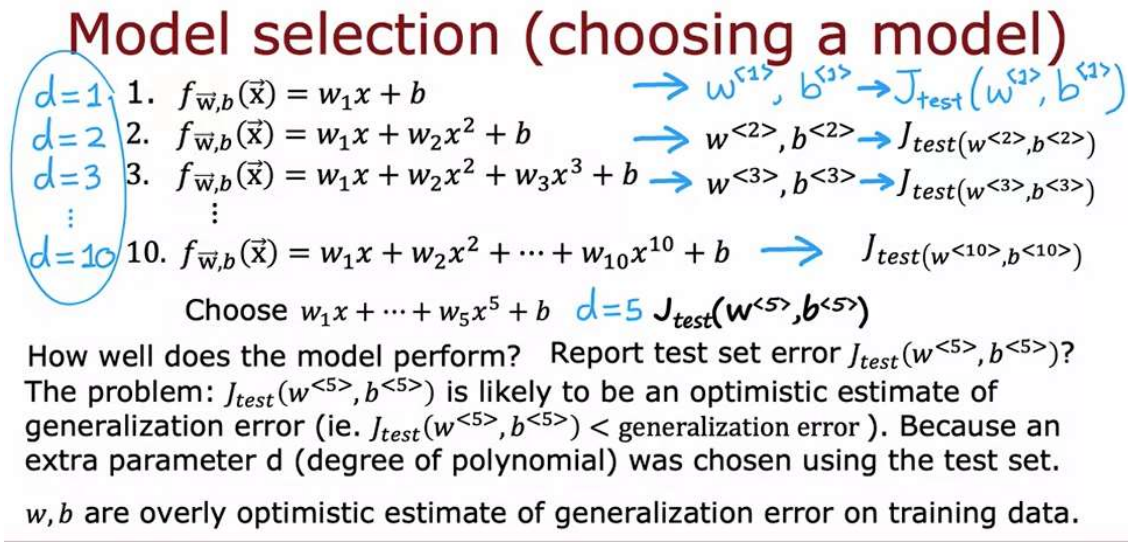

Der Trainingsfehler ist möglicherweise kein guter Indikator dafür, wie gut der Algorithmus funktionieren wird oder wie gut er sich auf neue Beispiele verallgemeinern lässt, die nicht im Trainingssatz enthalten waren, und insbesondere für dieses Beispiel wird der Trainingsfehler so gut wie Null sein.

Das ist wahrscheinlich viel niedriger als der tatsächliche Generalisierungsfehler, und damit meine ich den durchschnittlichen Fehler bei neuen Beispielen, die nicht im Trainingssatz enthalten waren. Was Sie im letzten Video gesehen haben, ist, dass J die Leistung des Algorithmus anhand von Beispielen testet, an denen nicht trainiert wird. Dies ist ein besserer Indikator dafür, wie gut das Modell wahrscheinlich bei neuen Daten abschneiden wird. Damit meine ich andere Daten, die nicht im Trainingssatz enthalten sind. Werfen wir einen Blick darauf, welche Auswirkungen dies hat und wie wir einen Testsatz verwenden könnten, um ein Modell für eine bestimmte Anwendung des maschinellen Lernens auszuwählen.

Wenn Sie eine Funktion zur Vorhersage von Immobilienpreisen oder einem anderen Regressionsproblem anpassen möchten, könnten Sie die Anpassung eines linearen Modells wie dieses in Betracht ziehen. Dies ist ein Polynom erster Ordnung und wir werden auf dieser Folie d gleich 1 verwenden, um die Anpassung an ein Polynom erster Ordnung oder eins zu bezeichnen. Wenn Sie ein solches Modell an Ihren Trainingssatz anpassen, erhalten Sie einige Parameter, w und b, und können dann J-Tests berechnen, um abzuschätzen, wie gut sich dies auf neue Daten verallgemeinern lässt.

Auf dieser Folie verwende ich w^1, b^1, um anzuzeigen, dass dies die Parameter sind, die Sie erhalten, wenn Sie ein Polynom erster Ordnung, ein Polynom vom Grad eins, d gleich 1, anpassen würden. Nun könnten Sie auch darüber nachdenken, ein Polynom zweiter Ordnung oder ein quadratisches Modell anzupassen, also ist dies das Modell. Wenn Sie dies an Ihren Trainingssatz anpassen würden, würden Sie einige Parameter w^2, b^2 erhalten, und Sie können diese Parameter

dann auf ähnliche Weise in Ihrem Testsatz auswerten und J-Test w^2, b^2 und erhalten Dadurch erhalten Sie einen Eindruck davon, wie gut das Polynom zweiter Ordnung funktioniert. Sie können dann d gleich 3 ausprobieren, das ist ein Polynom dritter Ordnung oder dritten Grades, das so aussieht, und die Parameter anpassen und auf ähnliche Weise den J-Test erhalten. Sie können so weitermachen, bis Sie, sagen wir, bis zu einem Polynom 10. Ordnung versuchen und am Ende einen J-Test für w^10, b^10 erhalten. Das gibt Ihnen einen Eindruck davon, wie gut das Polynom 10. Ordnung funktioniert.



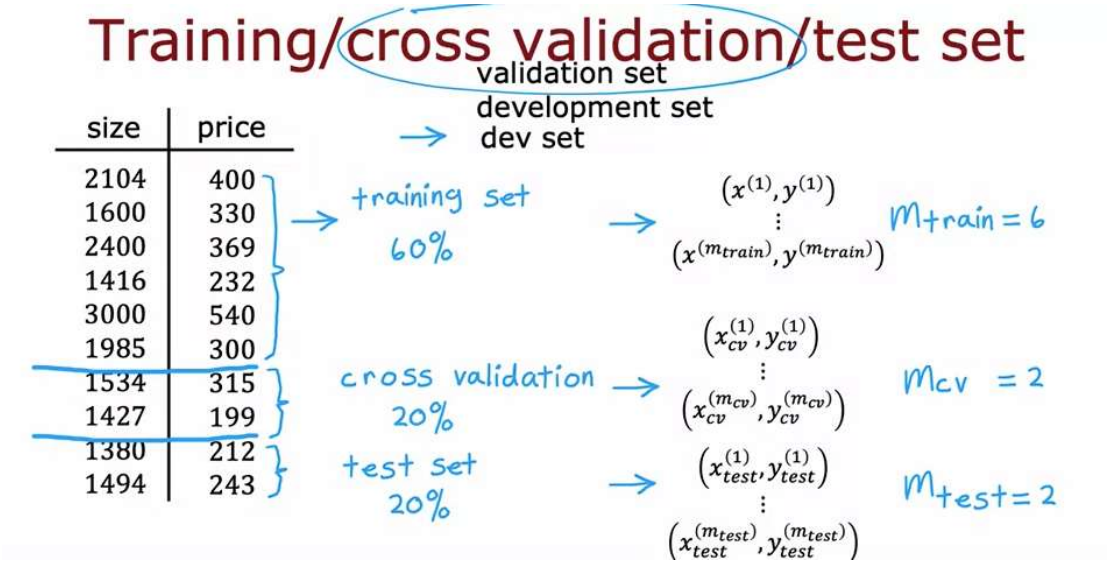

Ein Verfahren, das Sie ausprobieren könnten, ist zwar nicht das beste Verfahren, aber Sie könnten es versuchen: Schauen Sie sich alle diese J-Tests an und finden Sie heraus, welcher den niedrigsten Wert liefert. Angenommen, Sie stellen fest, dass der J-Test für das Polynom fünfter Ordnung für w^5, b^5 am niedrigsten ist.

Wenn dies der Fall ist, können Sie entscheiden, dass das Polynom fünfter Ordnung d gleich 5 am besten geeignet ist, und dieses Modell für Ihre Anwendung auswählen. Wenn Sie abschätzen möchten, wie gut dieses Modell funktioniert, können Sie den Testsatzfehler J test w^5, b^5 melden, was sich jedoch als leicht fehlerhaftes Verfahren herausstellt. Der Grund für die Fehlerhaftigkeit dieses Verfahrens liegt darin, dass der J-Test von w^5, b^5 wahrscheinlich eine optimistische Schätzung des Generalisierungsfehlers ist.

Mit anderen Worten, er ist wahrscheinlich geringer als der tatsächliche Generalisierungsfehler, und der Grund dafür ist, dass wir in dem Verfahren, über das wir auf dieser Folie mit Basisanpassungen gesprochen haben, einen zusätzlichen Parameter gewählt haben, nämlich d, den Grad des Polynoms diesen Parameter mithilfe des Testsatzes. Auf der vorherigen Folie haben wir gesehen, dass die Trainingsdaten eine zu optimistische Schätzung des Generalisierungsfehlers wären, wenn man w, b an die Trainingsdaten anpassen würde. Es stellt sich auch heraus, dass, wenn Sie den Parameter d mithilfe des Testsatzes auswählen möchten, der J-Test des Testsatzes nun zu optimistisch ist, d. h. niedriger als die tatsächliche Schätzung des Generalisierungsfehlers ist.

Das Verfahren auf dieser speziellen Folie ist fehlerhaft und ich empfehle die Verwendung nicht. Wenn Sie stattdessen automatisch ein Modell auswählen möchten, entscheiden Sie beispielsweise, welches Polynom Grad verwendet werden soll. So ändern Sie das Trainings- und Testverfahren, um die Modellauswahl durchzuführen. Unter Modellauswahl verstehe ich die Auswahl zwischen verschiedenen Modellen, wie zum Beispiel diesen 10 verschiedenen Modellen, die Sie möglicherweise für Ihre Anwendung für maschinelles Lernen verwenden möchten. Wir ändern das

Verfahren dahingehend, dass wir Ihre Daten nicht nur in zwei Teilmengen, den Trainingssatz und den Testsatz, aufteilen, sondern Ihre Daten in drei verschiedene Teilmengen aufteilen, die wir Trainingssatz nennen, der Kreuzvalidierungssatz und dann auch der Testsatz. Anhand unseres vorherigen Beispiels dieser 10 Trainingsbeispiele könnten wir es so aufteilen, dass 60 Prozent der Daten in den Trainingssatz eingefügt werden. Daher wird die Notation, die wir für den Trainingssatzteil verwenden, dieselbe wie zuvor sein, mit der Ausnahme, dass jetzt M Train wird die Anzahl der Trainingsbeispiele sechs betragen und wir könnten 20 Prozent der Daten in den Kreuzvalidierungssatz einfügen und eine Notation, die ich verwenden werde, ist x_cv von eins, y_cv von eins für das erste Kreuzvalidierungsbeispiel. cv steht also für Kreuzvalidierung, bis hin zu x_cv von m_cv und y_cv von m_cv. Wo hier m_cv in diesem Beispiel gleich 2 ist, ist die Anzahl der Kreuzvalidierungsbeispiele. Dann haben wir schließlich den gleichen Testsatz wie zuvor, also x1 bis xm Tests und y1 bis ym, wobei m Tests gleich 2 sind.




Dies ist die Anzahl der Testbeispiele. Auf der nächsten Folie erfahren Sie, wie Sie das Kreuzvalidierungsset verwenden.

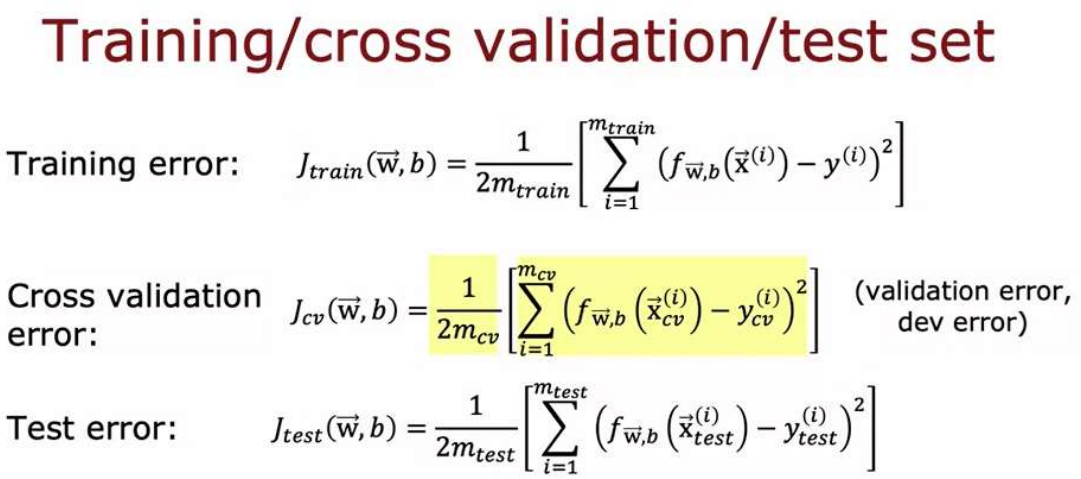
Wir werden das Verfahren so ändern, dass Sie den Trainingssatz und den Testsatz bereits gesehen haben und wir einen neuen Teilsatz der Daten einführen werden, der als Kreuzvalidierungssatz bezeichnet wird. Der Name Kreuzvalidierung bezieht sich darauf, dass es sich um einen zusätzlichen Datensatz handelt, den wir verwenden werden, um die Gültigkeit oder tatsächlich die Genauigkeit verschiedener Modelle zu überprüfen .

Ich glaube nicht, dass es ein toller Name ist, aber die Leute, die sich mit maschinellem Lernen befassen, haben diesen zusätzlichen Datensatz so genannt. Möglicherweise hören Sie auch, dass die Leute dies kurz „Validierungssatz“ nennen, es sind nur weniger Silben als bei der Kreuzvalidierung, oder in einigen Anwendungen wird dies auch als „Entwicklungssatz“ bezeichnet. Bedeutet im Grunde dasselbe. Manchmal nennt man dies das Dev-Set, aber alle diese Begriffe bedeuten dasselbe wie Cross-Validation-Set. Ich persönlich verwende den Begriff „Dev-Set“ am häufigsten, weil es die kürzeste und schnellste Art ist, es auszudrücken, aber Kreuzvalidierung wird von Praktikern des maschinellen Lernens etwas häufiger verwendet.

Auf diesen drei Teilmengen des Datentrainingssatzes, des Kreuzvalidierungssatzes und des Testsatzes können Sie dann mithilfe dieser drei Formeln den Trainingsfehler, den Kreuzvalidierungsfehler und den Testfehler berechnen. Während normalerweise keiner dieser Begriffe den im Trainingsziel enthaltenen Regularisierungsbegriff und diesen neuen Begriff in der Mitte enthält, ist der

Kreuzvalidierungsfehler nur der Durchschnitt Ihrer m_cv-Kreuzvalidierungsbeispiele des durchschnittlichen, sagen wir, quadratischen Fehlers.

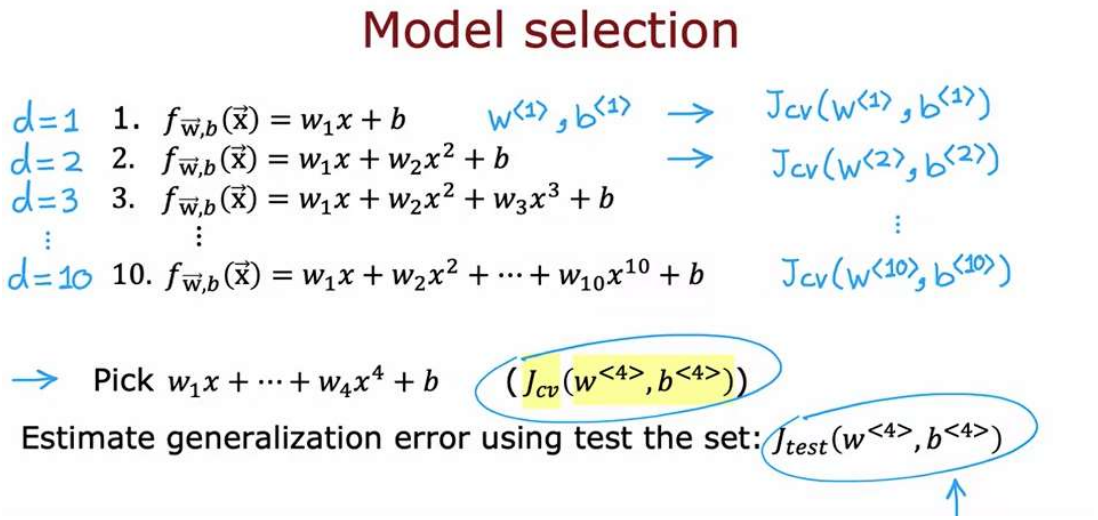
Dieser Begriff wird nicht nur Kreuzvalidierungsfehler, sondern auch kurz Validierungsfehler oder sogar Entwicklungssatzfehler oder Entwicklungsfehler genannt.




Mit diesen drei Maßstäben für die Leistung von Lernalgorithmen können Sie anschließend die Modellauswahl auf diese Weise durchführen. Sie können mit den 10 Modellen, wie zuvor auf dieser Folie, mit d gleich 1, d gleich 2, bis hin zu einem Polynom 10. Grades oder 10. Ordnung die Parameter w_1, b_1 anpassen.

Aber anstatt dies auf Ihrem Testsatz auszuwerten, werden Sie stattdessen diese Parameter auf Ihren Kreuzvalidierungssätzen auswerten und J_cv von w1, b1 berechnen, und in ähnlicher Weise erhalten wir für das zweite Modell J_cv von w2, v2 und so weiter bis hin zu J_cv von w10, b10. Um dann ein Modell auszuwählen, schauen Sie sich an, welches Modell den geringsten Kreuzvalidierungsfehler aufweist. Nehmen wir konkret an, dass J_cv von w4 und b4 so niedrig ist, dass Sie dieses Polynom vierter Ordnung auswählen als Modell, das Sie für diese Anwendung verwenden werden. Schließlich möchten Sie eine Schätzung des Generalisierungsfehlers darüber angeben, wie gut dieses Modell bei neuen Daten abschneidet. Dazu verwenden Sie die dritte Teilmenge Ihrer Daten, den Testsatz, und melden Jtest von w4 ,b4.

Sie bemerken, dass Sie diese Parameter während des gesamten Vorgangs mithilfe des Trainingssatzes angepasst haben. Anschließend haben Sie den Parameter d oder den Grad des Polynoms mithilfe des Kreuzvalidierungssatzes ausgewählt. Bis zu diesem Punkt haben Sie keine Parameter, weder w noch b noch d, an den Testsatz angepasst, weshalb in diesem Beispiel Jtest verwendet wird. Eine faire Schätzung des Generalisierungsfehlers dieses Modells ist somit die Parameter w4,b4.



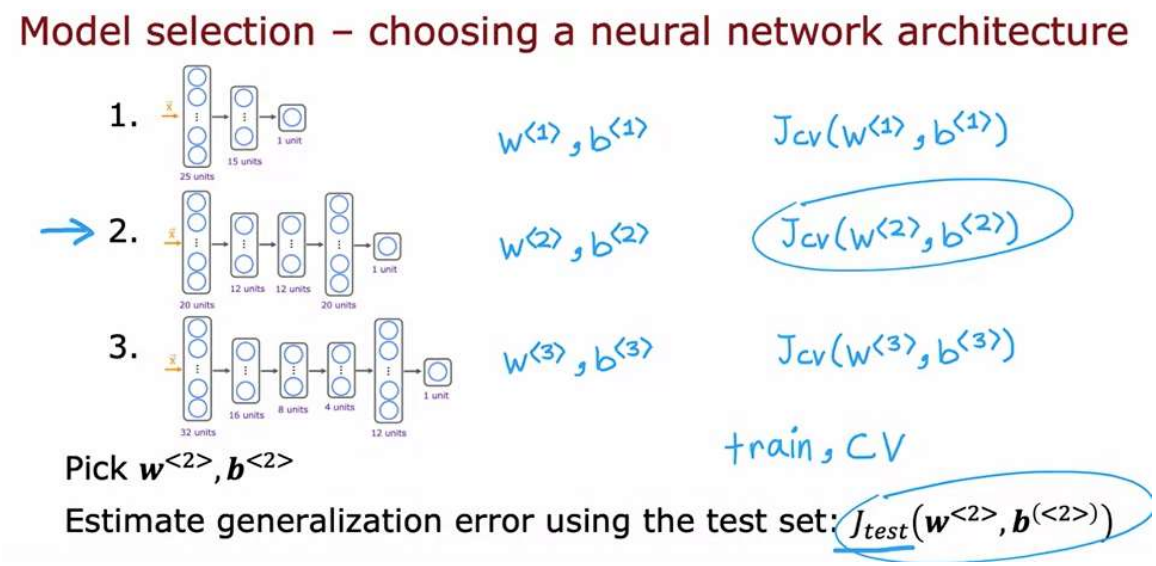

Dies ermöglicht ein besseres Verfahren zur Modellauswahl und ermöglicht es Ihnen, automatisch eine Entscheidung zu treffen, z. B. welches Polynom der Ordnung Sie für Ihr lineares Regressionsmodell wählen möchten.

Dieses Modellauswahlverfahren eignet sich auch für die Auswahl zwischen anderen Modelltypen. Wählen Sie beispielsweise eine neuronale Netzwerkarchitektur. Wenn Sie ein Modell für die handschriftliche Ziffernerkennung anpassen, könnten Sie drei Modelle wie dieses in Betracht ziehen, vielleicht sogar einen größeren Satz von Modellen, aber hier sind ein paar verschiedene neuronale Netze von klein, etwas größer und dann noch größer.

Um Ihnen bei der Entscheidung zu helfen, wie viele Schichten das neuronale Netzwerk hat und wie viele versteckte Einheiten pro Schicht Sie haben sollten, können Sie dann alle drei dieser Modelle trainieren und am Ende die Parameter w1, b1 für das erste Modell und w2, b2 für das erhalten zweites Modell und w3 ,b3 für das dritte Modell. Anschließend können Sie die Leistung neuronaler Netze mithilfe von Jcv anhand Ihres Kreuzvalidierungssatzes bewerten. Da es sich um ein Klassifizierungsproblem handelt, besteht die häufigste Wahl von Jcv darin, dies als den Anteil der Kreuzvalidierungsbeispiele zu berechnen, die der Algorithmus falsch klassifiziert hat. Sie würden dies mit allen drei Modellen berechnen und dann das Modell mit dem niedrigsten Kreuzvalidierungsfehler auswählen.

Wenn dies in diesem Beispiel den niedrigsten Kreuzvalidierungsfehler aufweist, wählen Sie das zweite neuronale Netzwerk aus und verwenden auf diesem Modell trainierte Parameter. Wenn Sie schließlich eine Schätzung des Generalisierungsfehlers melden möchten, verwenden Sie den Testsatz um abzuschätzen, wie gut das gerade ausgewählte neuronale Netzwerk funktionieren wird.

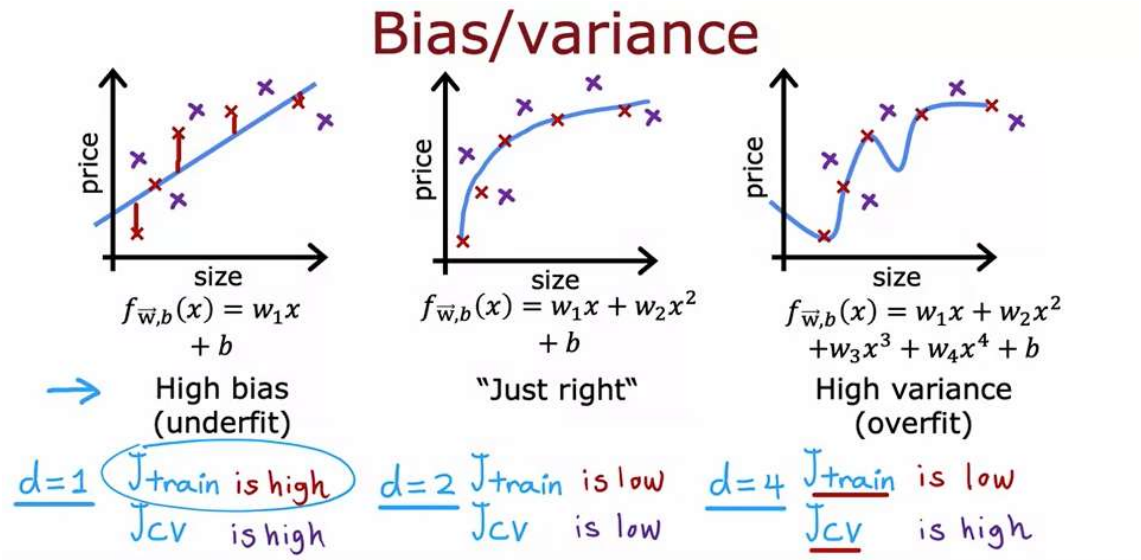
Beim maschinellen Lernen gilt es als Best Practice, dass Sie, wenn Sie Entscheidungen über Ihr Modell treffen müssen, z. B. Parameter anpassen oder die Modellarchitektur auswählen, z. B. die Architektur eines neuronalen Netzwerks (oder den Polynomgrad, wenn Sie eine lineare Regression anpassen), all diese Entscheidungen treffen müssen Entscheidungen nur anhand Ihres Trainingssatzes und Ihres Kreuzvalidierungssatzes treffen und sich den Testsatz überhaupt nicht ansehen, während Sie noch Entscheidungen bezüglich Ihres Lernalgorithmus treffen. Erst nachdem Sie ein Modell als Ihr endgültiges Modell entwickelt haben, können Sie es dann anhand des Testsatzes bewerten.




Da Sie anhand des Testsatzes noch keine Entscheidungen getroffen haben, wird sichergestellt, dass Ihr Testsatz fair und nicht übermäßig ist optimistische Schätzung, wie gut sich Ihr Modell auf neue Daten verallgemeinern lässt. Das ist die Modellauswahl und tatsächlich ein sehr weit verbreitetes Verfahren. Ich verwende dies ständig, um automatisch auszuwählen, welches Modell für eine bestimmte Anwendung des maschinellen Lernens verwendet werden soll. Anfang dieser Woche erwähnte ich die Durchführung von Diagnosen, um zu entscheiden, wie die Leistung eines Lernalgorithmus verbessert werden kann. Nachdem Sie nun die Möglichkeit haben, Lernalgorithmen zu bewerten und sogar automatisch ein Modell auszuwählen, wollen wir uns eingehender mit Beispielen für einige Diagnosen befassen. Die leistungsstärkste Diagnose, die ich kenne und die ich für viele Anwendungen des maschinellen Lernens verwendet habe, heißt Bias und Varianz. Schauen wir uns im nächsten Video an, was das bedeutet.

## Diagnose von Bias und Varianz

das Modell trainieren . Dabei stellen Sie fast immer fest, dass es noch nicht so gut funktioniert, wie Sie es sich wünschen. Wenn ich ein Modell für maschinelles Lernen trainiere, funktioniert es beim ersten Mal so gut wie nie so gut. Der Schlüssel zum Aufbau eines maschinellen Lernsystems liegt darin, zu entscheiden, was als Nächstes zu tun ist, um seine Leistung zu verbessern. Ich habe bei vielen verschiedenen Anwendungen festgestellt, dass die Betrachtung der Voreingenommenheit und Varianz eines Lernalgorithmus sehr gute Hinweise darauf gibt, was man als Nächstes ausprobieren sollte. Werfen wir einen Blick darauf, was das bedeutet. Vielleicht erinnern Sie sich an dieses Beispiel aus dem ersten Kurs zur linearen Regression. Wenn man bei diesem Datensatz eine gerade Linie daran anpasst, funktioniert das nicht so gut. Wir sagten, dass dieser Algorithmus eine hohe Verzerrung aufweist oder dass er nicht zu diesem Datensatz passt. Wenn Sie ein Polynom vierter Ordnung anpassen würden, dann weist es eine hohe Varianz auf oder es liegt eine Überanpassung vor. Wenn Sie in der Mitte ein quadratisches Polynom anpassen, sieht es ziemlich gut aus. Dann sagte ich, das sei genau richtig.



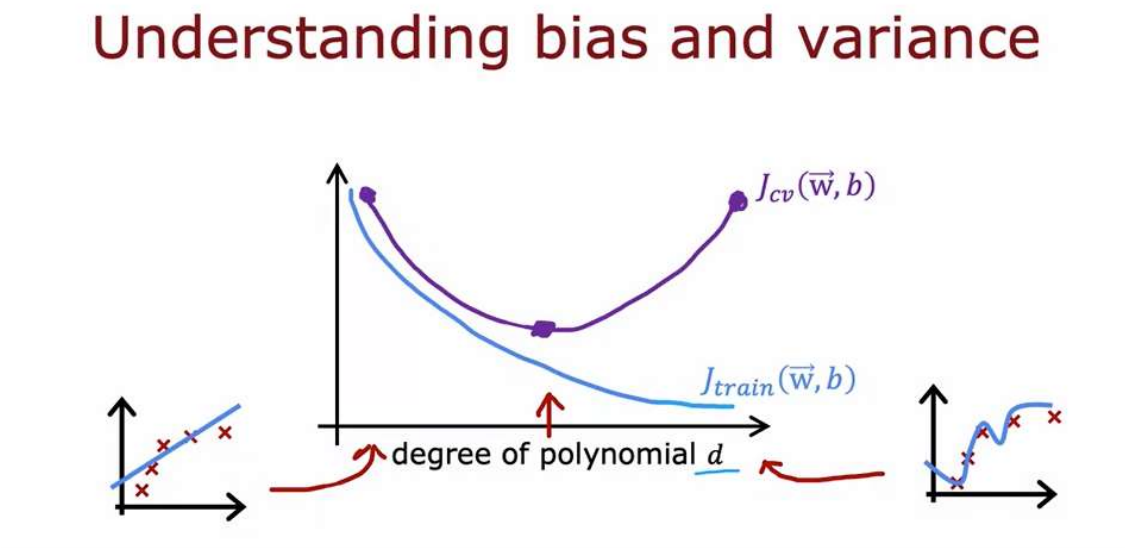

Da dies ein Problem mit nur einem einzelnen Merkmal x ist, könnten wir die Funktion f zeichnen und sie so betrachten. Aber wenn Sie mehr Funktionen hätten, können Sie f nicht so einfach darstellen und visualisieren, ob es gut funktioniert. Anstatt zu versuchen, Diagramme wie diese zu betrachten, besteht eine systematischere Möglichkeit zur Diagnose oder zum Herausfinden, ob Ihr Algorithmus eine hohe Verzerrung oder hohe Varianz aufweist, darin, die Leistung Ihres Algorithmus im Trainingssatz und im Kreuzvalidierungssatz zu betrachten. Schauen wir uns insbesondere das Beispiel links an. Wenn Sie J_train berechnen würden, wie gut schneidet der Algorithmus auf dem Trainingssatz ab?

Nicht so gut. Ich würde sagen, dass J train hier hoch wäre, weil es tatsächlich ziemlich große Fehler zwischen den Beispielen und den tatsächlichen Vorhersagen des Modells gibt. Wie wäre es mit J_cv? J_cv wäre, wenn wir ein paar neue Beispiele hätten, vielleicht solche Beispiele, die der Algorithmus vorher nicht gesehen hat. Auch hier schneidet der Algorithmus bei Beispielen, die er zuvor noch nicht gesehen hat, nicht so gut ab, sodass J_cv ebenfalls hoch sein wird. Ein Merkmal eines Algorithmus mit hoher Verzerrung, der nicht ausreichend angepasst ist, besteht darin, dass er auf dem Trainingssatz nicht einmal so gut abschneidet.

Wenn J_train hoch ist, ist das Ihr starker Indikator dafür, dass dieser Algorithmus eine hohe Verzerrung aufweist. Schauen wir uns nun das Beispiel rechts an. Wenn Sie J_train berechnen würden, wie gut funktioniert das auf dem Trainingssatz? Nun ja, am Trainingsset macht es sich tatsächlich großartig. Passt wirklich gut zu den Trainingsdaten. J_train wird hier niedrig sein. Wenn Sie dieses Modell jedoch an anderen Häusern bewerten würden, die nicht im Trainingssatz enthalten sind, werden Sie feststellen, dass J_cv, der Kreuzvalidierungsfehler, ziemlich hoch ist. Eine charakteristische Signatur oder ein charakteristisches Q, bei dem Ihr Algorithmus eine hohe Varianz aufweist, ist, dass J_cv viel höher ist als J_train.

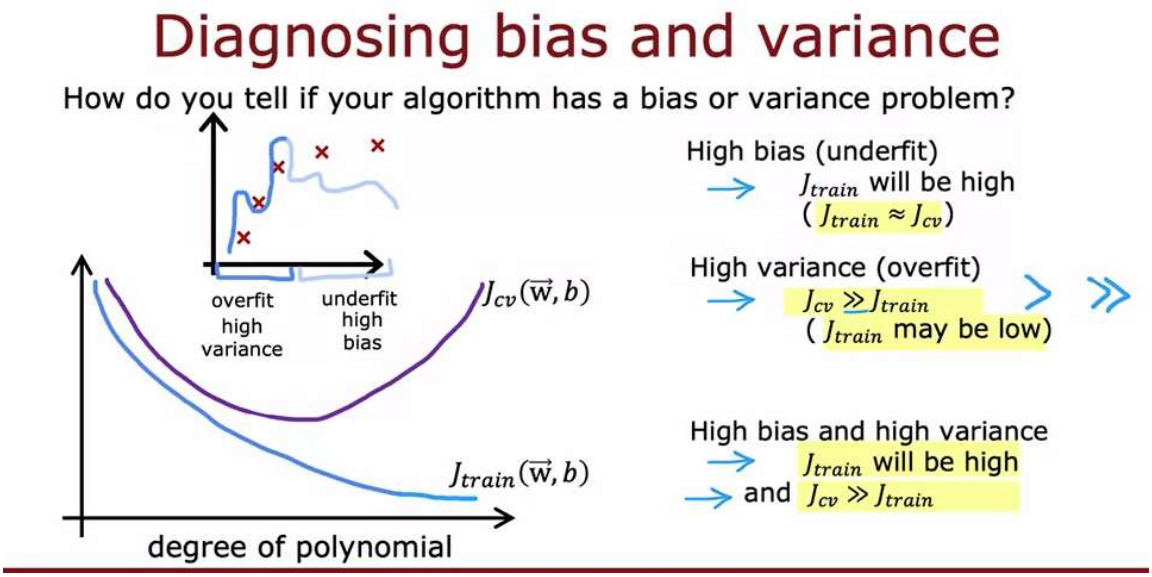
Mit anderen Worten: Es schneidet bei Daten, die es gesehen hat, viel besser ab als bei Daten, die es nicht gesehen hat. Dies erweist sich als starker Indikator dafür, dass Ihr Algorithmus eine hohe Varianz aufweist. Der Sinn unserer Arbeit besteht wiederum darin, dass ich J_train und J_cv berechne und schaue, ob J _train hoch ist oder ob J_cv viel höher als J_train ist. Dies gibt Ihnen einen Eindruck davon, ob Ihr Algorithmus eine hohe Verzerrung oder eine hohe Varianz aufweist, auch wenn Sie die Funktion f nicht grafisch darstellen können. Endlich die Verfolgungsjagd in der Mitte. Wenn Sie sich J_train ansehen, ist es ziemlich niedrig, sodass es auf dem Trainingsset recht gut funktioniert. Wenn Sie sich ein paar neue Beispiele ansehen, beispielsweise die aus Ihrem Kreuzvalidierungssatz, werden Sie feststellen, dass J_cv ebenfalls ziemlich niedrig ist. Wenn J_train nicht zu hoch ist, bedeutet dies,

dass kein Problem mit hoher Verzerrung vorliegt, und wenn J_cv nicht viel schlechter als J_train ist, deutet dies darauf hin, dass es auch kein Problem mit hoher Varianz gibt. Aus diesem Grund scheint das quadratische Modell für diese Anwendung ziemlich gut geeignet zu sein. Zusammenfassend lässt sich sagen, dass J_train hoch und J_cv hoch war, wenn d für ein lineares Polynom gleich 1 ist. Wenn d gleich 4 ist, war J train niedrig, aber J_cv ist hoch. Wenn d gleich 2 ist, waren beide ziemlich niedrig. Lassen Sie uns nun eine andere Sicht auf Voreingenommenheit und Varianz einnehmen. Insbesondere möchte ich Ihnen auf der nächsten Folie zeigen, wie J_train und J_cv die Varianz als Funktion des Grades des Polynoms, das Sie anpassen, beeinflussen. Lassen Sie mich eine Figur zeichnen, bei der die horizontale Achse, dieses d hier, der Grad des Polynoms ist, den wir an die Daten anpassen. Auf der linken Seite entsprechen wir einem kleinen Wert von d, etwa d gleich 1, was einer geraden Linie entspricht. Auf der rechten Seite entsprechen wir beispielsweise d gleich 4 oder sogar höheren Werten von d. Wir passen dieses Polynom höherer Ordnung an. Wenn Sie also J train oder W, B als Funktion des Grades des Polynoms grafisch darstellen würden, würden Sie Folgendes feststellen: Wenn Sie ein Polynom immer höheren Grades anpassen, gehe ich davon aus, dass wir hier nicht die Regularisierung verwenden, sondern as Wenn Sie ein Polynom immer höherer Ordnung anpassen, sinkt der Trainingsfehler tendenziell, denn wenn Sie eine sehr einfache lineare Funktion haben, passt diese nicht so gut zu den Trainingsdaten, wenn Sie eine quadratische Funktion oder ein Polynom dritter oder vierter Ordnung anpassen -Ordnungspolynom passt es immer besser zu den Trainingsdaten. Wenn der Grad des Polynoms zunimmt, sinkt der J-Zug normalerweise. Schauen wir uns als Nächstes J_cv an.




Wie gut funktioniert es bei Daten, an die es nicht angepasst werden konnte? Was wir gesehen haben, war, dass, wenn d gleich eins war und der Grad des Polynoms sehr niedrig war, J_cv ziemlich hoch war, weil es zu wenig passte, sodass es im Kreuzvalidierungssatz nicht gut abschnitt. Auch hier auf der rechten Seite: Wenn der Grad des Polynoms sehr groß ist, beispielsweise vier, schneidet es im Kreuzvalidierungssatz ebenfalls nicht gut ab und ist daher ebenfalls hoch. Aber wenn d dazwischen lag, sagen wir, ein Polynom zweiter Ordnung, dann lief es tatsächlich viel besser. Wenn Sie den Grad des Polynoms variieren würden, würden Sie tatsächlich eine Kurve erhalten, die so aussieht, die nach unten und dann wieder nach oben verläuft. Wenn der Grad des Polynoms zu niedrig ist, passt es zu wenig und führt daher den Kreuzvalidierungssatz nicht aus. Wenn er zu hoch ist, passt es zu stark und schneidet auch im Kreuzvalidierungssatz nicht gut ab. Nur wenn es irgendwo in der Mitte liegt, ist das genau richtig, weshalb das Polynom zweiter Ordnung in unserem Beispiel am Ende einen geringeren Kreuzvalidierungsfehler und weder eine hohe Verzerrung noch eine hohe Varianz aufweist. Zusammenfassend lässt sich sagen: Wie diagnostizieren Sie Voreingenommenheit und Varianz in

Ihrem Lernalgorithmus? Wenn Ihr Lernalgorithmus eine hohe Voreingenommenheit aufweist oder über unbesiegte Daten verfügt, ist der Schlüsselindikator, ob der J-Zug hoch ist. Das entspricht dem äußersten linken Teil der Kurve, wo J den höchsten Wert erreicht. Normalerweise liegen J train und J_cv nahe beieinander. Wie diagnostizieren Sie, wenn Sie eine hohe Varianz haben? Während der Schlüsselindikator für eine hohe Varianz darin besteht, dass J_cv viel größer ist als J, trainiert, ist das Vorzeichen in der Mathematik viel größer als, also ist dies größer, und das bedeutet viel größer. In diesem rechten Teil des Diagramms ist J_cv viel größer als J train. Normalerweise ist J train ziemlich niedrig, aber der Schlüsselindikator ist, ob J_cv viel größer als J train ist. Das passiert, wenn wir ein Polynom sehr hoher Ordnung an diesen kleinen Datensatz angepasst haben. Obwohl wir gerade Käufer in den Bereichen gesehen haben, stellt sich heraus, dass es in einigen Fällen möglich ist, gleichzeitig eine hohe Voreingenommenheit und eine hohe Varianz zu haben.

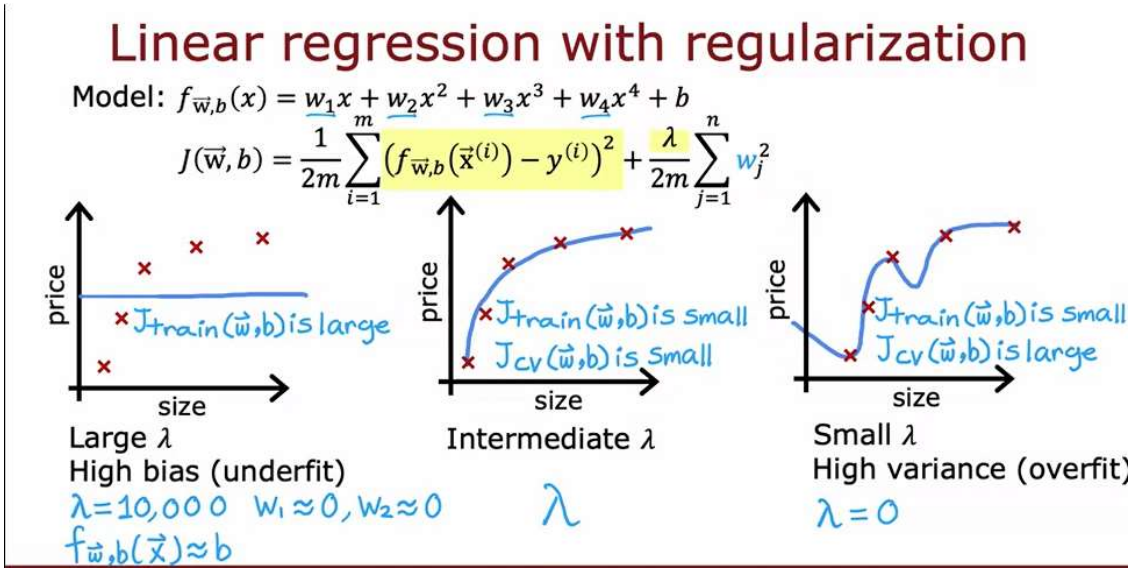



Bei der linearen Regression wird dies nicht so oft passieren, aber es stellt sich heraus, dass es beim Training eines neuronalen Netzwerks einige Anwendungen gibt, bei denen leider eine hohe Verzerrung und eine hohe Varianz auftreten. Eine Möglichkeit, diese Situation zu erkennen, besteht darin, dass der J-Zug hoch ist, sodass Sie auf dem Trainingssatz nicht so gut abschneiden, aber noch schlimmer ist, dass der Kreuzvalidierungsfehler wiederum sogar viel größer ist als der Trainingssatz. Die Vorstellung von hoher Verzerrung und hoher Varianz kommt bei linearen Modellen, die auf eine Tiefe angewendet werden, nicht wirklich vor. Aber um eine Vorstellung davon zu vermitteln, wie es aussieht: Es wäre so, als ob man für einen Teil der Eingabe ein sehr kompliziertes Modell hätte, das überpasst, also überpasst es an einen Teil der Eingaben. Aber aus irgendeinem Grund passt es für andere Teile der Eingabe nicht einmal gut zu den Trainingsdaten und daher für einen Teil der Eingabe nicht ausreichend. In diesem Beispiel, das künstlich aussieht, weil es sich um eine einzelne Feature-Eingabe handelt, passen wir den Trainingssatz wirklich gut an und überpassen einen Teil der Eingabe, und wir passen die Trainingsdaten nicht einmal gut an und wir passen einen Teil der Eingabe nicht gut an . Daher kann es bei manchen Anwendungen unglücklicherweise zu einer hohen Verzerrung und einer hohen Varianz kommen. Der Indikator dafür ist, dass der Algorithmus auf dem Trainingssatz schlecht abschneidet, und wenn er sogar viel schlechter abschneidet als auf dem Trainingssatz. Bei den meisten Lernanwendungen haben Sie wahrscheinlich in erster Linie ein Problem mit hoher Verzerrung oder hoher Varianz und nicht beides gleichzeitig. Aber es ist möglich, dass manchmal beides gleichzeitig der Fall ist. Ich weiß, dass es viele Prozesse und viele Konzepte auf den Folien gibt, aber die wichtigsten Erkenntnisse sind, dass ein hoher Bias bedeutet, dass er auf dem Trainingssatz nicht einmal gut abschneidet, und eine hohe Varianz bedeutet, dass er auf dem Cross viel schlechter abschneidet Validierungssatz und Trainingssatz. Wenn ich einen Algorithmus für

maschinelles Lernen trainiere, versuche ich fast immer herauszufinden, inwieweit der Algorithmus eine hohe Verzerrung oder Unteranpassung im Vergleich zu einem Problem mit hoher Varianz bei Überanpassung aufweist. Wie wir später in dieser Woche sehen werden, wird dies eine gute Anleitung sein, wie Sie die Leistung des Algorithmus verbessern können. Aber werfen wir zunächst einen Blick darauf, wie sich die Regularisierung auf die Voreingenommenheit und Varianz eines Lernalgorithmus auswirkt, denn das wird Ihnen helfen, besser zu verstehen, wann Sie die Regularisierung verwenden sollten. Schauen wir uns das im nächsten Video an.

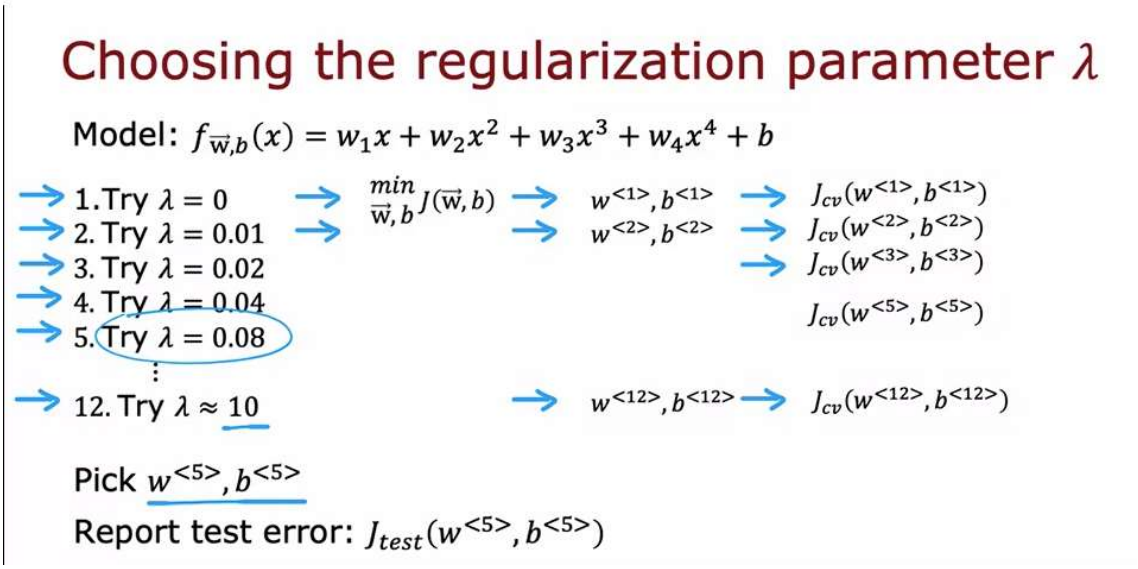
## Regularisierung und Bias/Varianz

Im letzten Video haben Sie gesehen, wie sich unterschiedliche Entscheidungen für den Grad des Polynoms D auf die Varianzabweichung Ihres Lernalgorithmus und damit auf dessen Gesamtleistung auswirken. Schauen wir uns in diesem Video an, wie sich die Regularisierung, insbesondere die Wahl des Regularisierungsparameters Lambda, auf die Bias und Varianz und damit auf die Gesamtleistung des Algorithmus auswirkt. Es stellt sich heraus, dass dies hilfreich ist, wenn Sie einen guten Lambda-Wert des Regularisierungsparameters für Ihren Algorithmus auswählen möchten. Lass uns einen Blick darauf werfen. In diesem Beispiel verwende ich ein Polynom vierter Ordnung, aber wir werden dieses Modell durch Regularisierung anpassen.



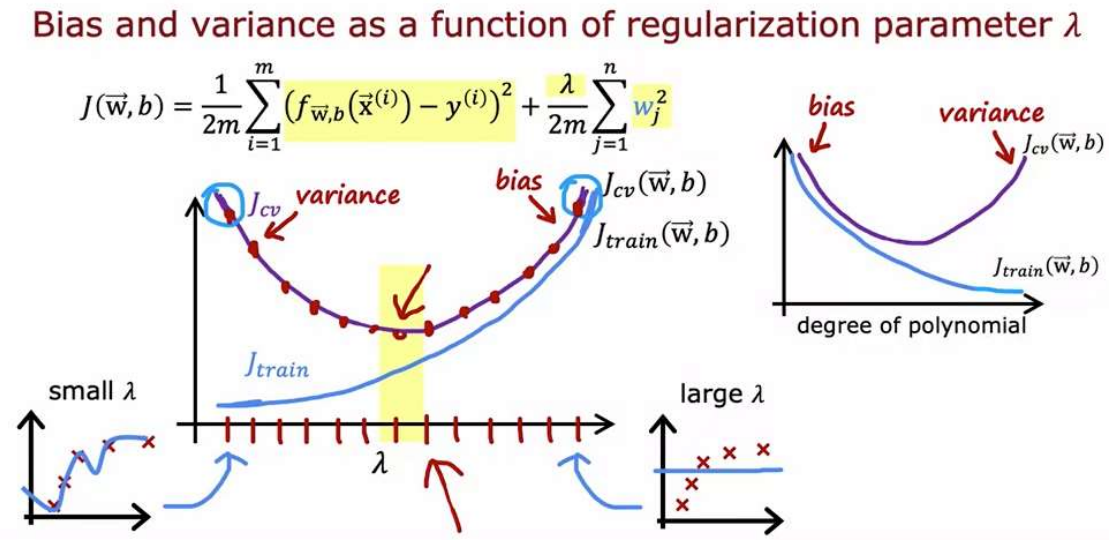

Dabei ist der Wert von Lambda der Regularisierungsparameter, der steuert, wie viel Kompromiss Sie eingehen, indem Sie die Parameter w klein halten und die Trainingsdaten gut anpassen. Beginnen wir mit dem Beispiel, Lambda auf einen sehr großen Wert festzulegen. Angenommen, Lambda ist gleich 10.000. Wenn Sie dies tun würden, würden Sie am Ende ein Modell anbringen, das ungefähr so aussieht. Denn wenn Lambda sehr groß wäre, dann ist der Algorithmus hochmotiviert, diese Parameter w sehr klein zu halten, und so erhält man am Ende w_1, w_2, eigentlich sind alle diese Parameter sehr nahe bei Null. Am Ende ist f von x nur annähernd ein konstanter Wert, weshalb man am Ende ein Modell wie dieses erhält. Dieses Modell weist eindeutig eine hohe Verzerrung auf und passt nicht gut zu den Trainingsdaten, da es auf dem Trainingssatz nicht einmal gut abschneidet und J_train groß ist. Werfen wir einen Blick auf das andere Extrem. Nehmen wir an, Sie stellen Lambda auf einen sehr kleinen Wert ein. Bei einem kleinen Lambda-Wert gehen wir tatsächlich zum Extrem und setzen Lambda gleich Null. Bei dieser Wahl von Lambda gibt es keine Regularisierung, also passen wir einfach ein Polynom vierter Ordnung ohne Regularisierung an und am Ende erhalten Sie

die Kurve, die Sie zuvor gesehen haben und die die Daten überpasst. Was wir zuvor gesehen haben, war, dass bei einem Modell wie diesem J_train klein ist, aber J_cv viel größer als J_train oder J_cv groß ist. Dies weist darauf hin, dass wir eine hohe Varianz haben und diese Daten überpassen. Wenn Sie einen Zwischenwert von Lambda haben, nicht wirklich groß 10.000, aber nicht so klein wie Null, erhalten Sie hoffentlich ein Modell, das so aussieht, das genau richtig ist und gut zu den Daten mit kleinem J_train und kleinem J_cv passt. Wenn Sie entscheiden möchten, welcher Lambda-Wert für den Regularisierungsparameter geeignet ist, bietet Ihnen die Kreuzvalidierung ebenfalls eine Möglichkeit. Werfen wir einen Blick darauf, wie wir dies tun könnten. Zur Erinnerung: Das Problem, mit dem wir uns befassen, besteht darin, dass Sie, wenn Sie ein Polynom vierter Ordnung anpassen, das ist also das Modell, und Sie die Regularisierung verwenden, einen guten Lambda-Wert auswählen können.




Dies wären Verfahren, die denen ähneln, die Sie für die Auswahl des Grades des Polynoms D mithilfe der Kreuzvalidierung gesehen haben. Nehmen wir insbesondere an, wir versuchen, ein Modell mit Lambda gleich 0 anzupassen. Wir würden die Kostenfunktion mit Lambda gleich 0 minimieren und am Ende einige Parameter w1, b1 erhalten, und Sie können dann den Kreuzvalidierungsfehler J_cv von w1, b1 berechnen . Versuchen wir nun einen anderen Wert von Lambda. Nehmen wir an, Sie versuchen, dass Lambda gleich 0,01 ist. Andererseits erhalten Sie durch die Minimierung der Kostenfunktion einen zweiten Parametersatz, w2, b2, und Sie können auch sehen, wie gut das beim Kreuzvalidierungssatz funktioniert, und so weiter. Probieren wir weiterhin andere Lambda-Werte aus und in diesem Beispiel werde ich versuchen, ihn zu verdoppeln, sodass Lambda gleich 0,02 ist. Dadurch erhalten Sie J_cv von w3, b3 usw. Dann lasst uns noch einmal verdoppeln und noch einmal verdoppeln. Nach mehrmaliger Verdoppelung erhalten Sie einen Lambda-Wert von ungefähr 10, und das ergibt die Parameter w12, b12 und J_cv w12 von b12. Indem Sie einen großen Bereich möglicher Werte für Lambda ausprobieren, Parameter mithilfe dieser verschiedenen Regularisierungsparameter anpassen und dann die Leistung des Kreuzvalidierungssatzes bewerten, können Sie dann versuchen, den besten Wert für den Regularisierungsparameter auszuwählen. Schnell. Wenn Sie in diesem Beispiel feststellen, dass J_cv von W5, B5 den niedrigsten Wert aller dieser verschiedenen Kreuzvalidierungsfehler aufweist, können Sie sich entscheiden, diesen Wert für Lambda auszuwählen und daher W5, B5 als ausgewählte Parameter zu verwenden. Wenn Sie schließlich eine Schätzung des Generalisierungsfehlers melden möchten , melden Sie dann den Testsatzfehler, J-Tests von W5, B5. Um die Funktionsweise dieses Algorithmus noch besser zu verstehen, werfen wir einen Blick darauf, wie Trainingsfehler und Kreuzvalidierungsfehler als Funktion des Parameters Lambda variieren. In dieser Abbildung habe ich die x-Achse erneut

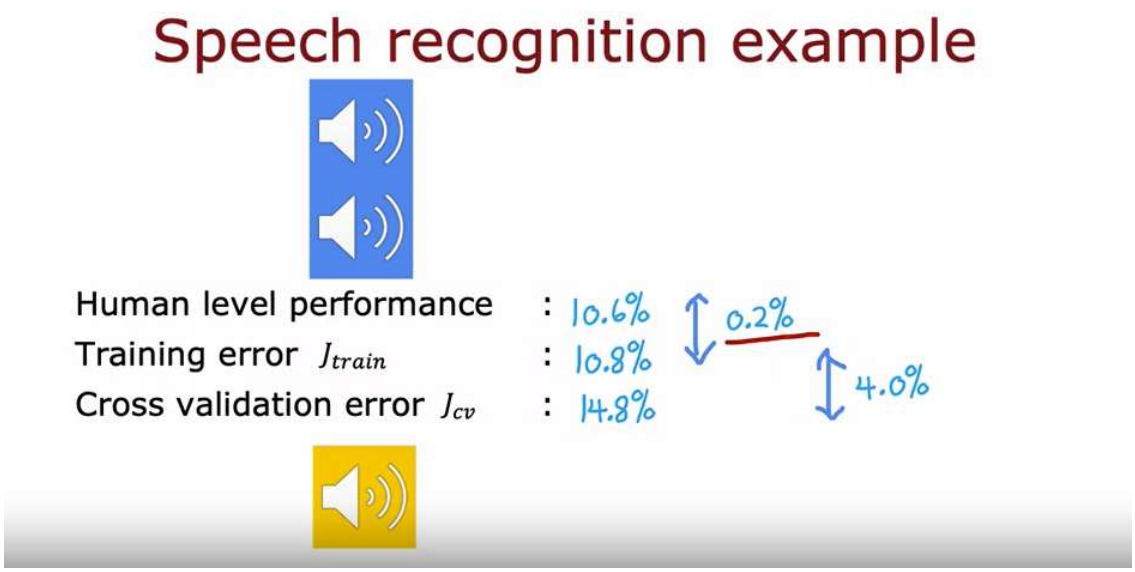
geändert. Beachten Sie, dass die x-Achse hier mit dem Wert des Regularisierungsparameters Lambda versehen ist, und wenn wir uns das Extrem von Lambda gleich Null hier links ansehen, entspricht das der Verwendung keiner Regularisierung, und so sind wir am Ende angelangt diese sehr wackelige Kurve. Wenn Lambda klein oder sogar Null wäre, und in diesem Fall haben wir ein Modell mit hoher Varianz, wird J train klein und J_cv groß sein, weil es bei den Trainingsdaten gut abschneidet, aber viel schlechter auf den Kreuzvalidierungsdaten. Dieses Extrem auf der rechten Seite waren sehr große Lambda-Werte. Angenommen, Lambda ist gleich 10.000, dann ergibt sich ein Modell, das so aussieht. Dies weist eine hohe Verzerrung auf, passt nicht zu den Daten und es stellt sich heraus, dass J train hoch sein wird und J_cv ebenfalls hoch sein wird. Wenn Sie sich tatsächlich ansehen, wie sich der J-Zug als Funktion von Lambda ändert, werden Sie feststellen, dass der J-Zug wie folgt ansteigt, denn in der Optimierungskostenfunktion gilt: Je größer Lambda, desto mehr versucht der Algorithmus, W beizubehalten quadriert klein. Das heißt, je mehr Gewicht diesem Regularisierungsterm beigemessen wird, desto weniger wird darauf geachtet, dass der Trainingssatz tatsächlich gut abschneidet. Dieser Begriff auf der linken Seite ist J-Train. Je mehr man also versucht, die Parameter klein zu halten, desto schlechter gelingt die Minimierung des Trainingsfehlers. Aus diesem Grund nimmt der Trainingsfehler J train mit zunehmendem Lambda tendenziell zu. Wie wäre es nun mit dem Kreuzvalidierungsfehler? Es stellt sich heraus, dass der Kreuzvalidierungsfehler so aussehen wird. Denn wir haben gesehen, dass Lambda, wenn es zu klein oder zu groß ist, im Kreuzvalidierungssatz nicht gut abschneidet. Entweder passt es hier links zu sehr oder hier rechts zu wenig. Es gibt einen Zwischenwert von Lambda, der dafür sorgt, dass der Algorithmus die beste Leistung erbringt. Bei der Kreuzvalidierung werden viele verschiedene Werte von Lambda ausprobiert. Das haben wir auf der letzten Folie gesehen; Versuchs-Lambda ist gleich Null, Lambda ist gleich 0,01, die Logik ist 0,02 . Probieren Sie viele verschiedene Werte von Lambda aus und bewerten Sie den Kreuzvalidierungsfehler an vielen dieser verschiedenen Punkte. Wählen Sie dann hoffentlich einen Wert aus, der einen geringen Kreuzvalidierungsfehler aufweist, und dieser entspricht hoffentlich einem guten Modell für Ihre Anwendung. Wenn Sie dieses Diagramm mit dem im vorherigen Video vergleichen, bei dem die horizontale Achse den Grad des Polynoms darstellt, sehen diese beiden Diagramme ein wenig nicht mathematisch und nicht in irgendeiner formalen Weise aus, aber sie sehen ein wenig wie ein Spiegel aus Bilder voneinander, und das liegt daran, dass bei der Anpassung eines Polynomgrades der linke Teil dieser Kurve einer Unteranpassung und einem hohen Bias entsprach, der rechte Teil einer Überanpassung und einer hohen Varianz. In diesem Fall befand sich die hohe Varianz auf der linken Seite und die hohe Voreingenommenheit auf der rechten Seite. Aber deshalb sind diese beiden Bilder ein wenig wie Spiegelbilder voneinander. Aber in beiden Fällen kann Ihnen die Kreuzvalidierung und die Auswertung verschiedener Werte dabei helfen, einen guten t-Wert oder einen guten Lambda-Wert auszuwählen.




Auf diese Weise wirkt sich die Wahl des Regularisierungsparameters Lambda auf die Bias und Varianz sowie die Gesamtleistung Ihres Algorithmus aus. Außerdem haben Sie gesehen, wie Sie mithilfe der Kreuzvalidierung eine gute Wahl für den Regularisierungsparameter Lambda treffen können. Bisher haben wir darüber gesprochen, dass ein hoher Trainingssatzfehler und ein hoher J-Zug auf eine hohe Verzerrung hinweisen und dass ein hoher Kreuzvalidierungsfehler von J_cv, insbesondere wenn er viel höher als der J-Zug ist, ein Hinweis darauf ist des Varianzproblems. Aber was bedeuten diese Worte „hoch“ oder „viel höher“ eigentlich? Schauen wir uns das im nächsten Video an, in dem wir uns ansehen, wie Sie die Zahlen J train und J_cv betrachten und beurteilen können, ob sie hoch oder niedrig sind. Es stellt sich heraus, dass eine weitere Verfeinerung dieser Ideen erforderlich ist, nämlich Durch die Festlegung eines grundlegenden Leistungsniveaus unseres Lernalgorithmus wird es Ihnen viel leichter fallen, sich diese Zahlen, J train, J_cv, anzusehen und zu beurteilen, ob sie hoch oder niedrig sind. Schauen wir uns im nächsten Video an, was das alles bedeutet.

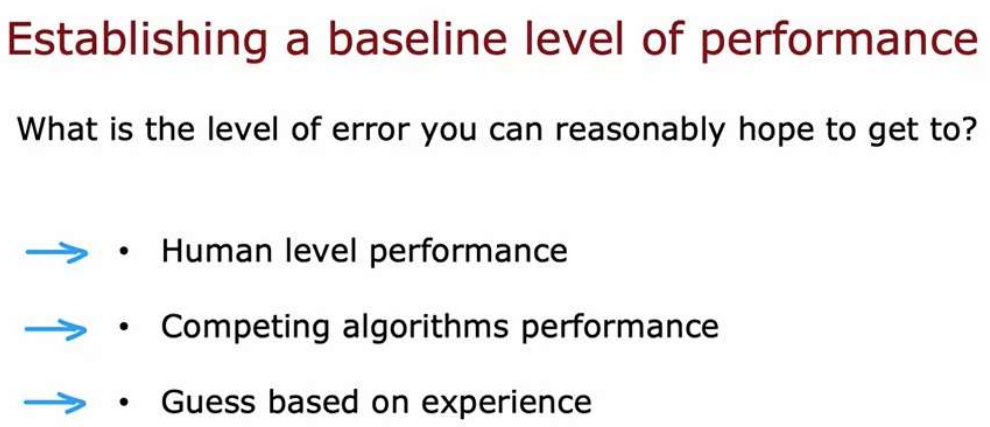
## Festlegung eines grundlegenden Leistungsniveaus

Schauen wir uns einige konkrete Zahlen für J-Train und JCV an und sehen wir uns an, wie Sie beurteilen können, ob ein Lernalgorithmus eine hohe Verzerrung oder hohe Varianz aufweist. Für die Beispiele in diesem Video verwende ich als fortlaufendes Beispiel die Anwendung der Spracherkennung, an der ich im Laufe der Jahre mehrfach gearbeitet habe. Lass uns einen Blick darauf werfen.



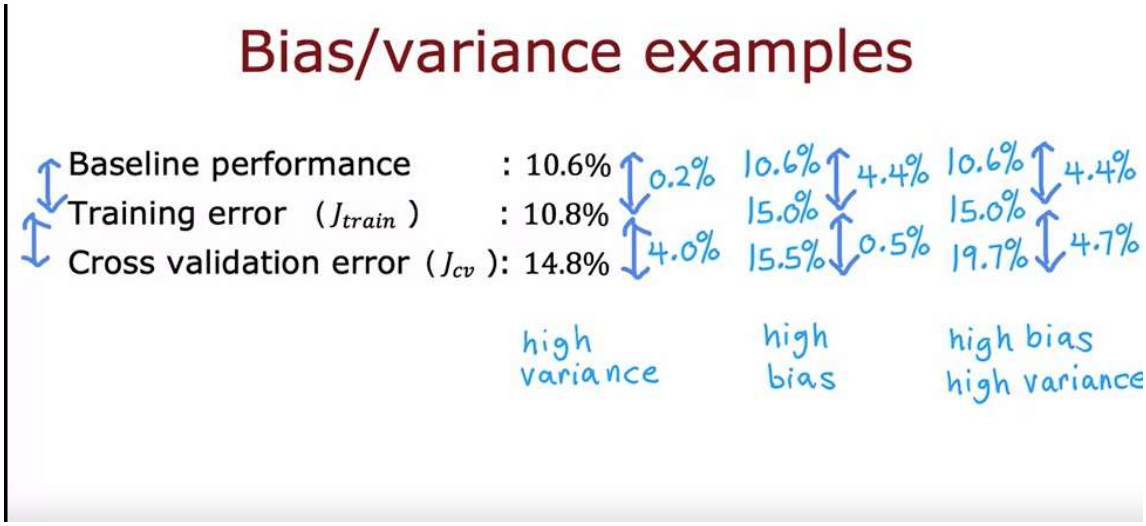

Viele Benutzer, die eine Websuche auf einem Mobiltelefon durchführen, verwenden die Spracherkennung, anstatt auf den winzigen Tastaturen unserer Telefone zu tippen, da das Sprechen mit einem Telefon oft schneller ist als das Tippen. Eine typische Audiodatei, die wir von einer Websuchmaschine erhalten, wäre etwa so: „Wie ist das Wetter heute?“ Oder so: „Cafés in meiner Nähe.“ Es ist die Aufgabe der Spracherkennungsalgorithmen, die Transkripte auszugeben, unabhängig davon, ob es sich um das heutige Wetter oder um Cafés in meiner Nähe handelt. Wenn Sie nun ein Spracherkennungssystem trainieren und den Trainingsfehler messen würden, und der Trainingsfehler bedeutet, wie viel Prozent der Audioclips in Ihrem Trainingssatz der Algorithmus nicht vollständig korrekt transkribiert. Nehmen wir an, der Trainingsfehler für diesen Datensatz beträgt 10,8 Prozent, was bedeutet, dass er ihn für 89,2 Prozent Ihres Trainingssatzes perfekt transkribiert, bei 10,8 Prozent Ihres Trainingssatzes jedoch einen Fehler macht. Wenn Sie die Leistung Ihres Spracherkennungsalgorithmus auch an einem separaten Kreuzvalidierungssatz messen würden, würde er beispielsweise einen Fehler von 14,8 Prozent ergeben. Wenn Sie sich diese Zahlen ansehen, sieht es so aus, als ob der Trainingsfehler wirklich hoch ist, er hat 10 Prozent falsch gemacht, und dann ist der Kreuzvalidierungsfehler höher, aber selbst 10 Prozent Ihres Trainingssatzes falsch zu machen, scheint ziemlich hoch zu sein. Es sieht so aus, als würde ein Fehler von 10 Prozent Sie zu dem Schluss verleiten, dass es sich um eine hohe Verzerrung handelt, weil es in Ihrem Trainingssatz nicht gut abschneidet, aber es stellt sich heraus, dass es bei der Analyse der Spracherkennung nützlich ist, auch eine andere Sache zu messen, nämlich die menschliche Ebene der Leistung? Mit anderen Worten: Wie gut können selbst Menschen Sprache aus diesen Audioclips genau transkribieren? Nehmen wir konkret an, Sie messen, wie gut Sprecher Audioclips transkribieren können, und stellen fest, dass die menschliche Leistung einen Fehler von 10,6 Prozent aufweist. Warum ist der menschliche Fehler so hoch? Es stellt sich heraus, dass es für die Websuche viele Audioclips gibt, die so klingen: „Ich navigiere zu [unverständlich].“ Es gibt viele verrauschte Audioinhalte, bei denen aufgrund des Rauschens im Ton niemand das Gesagte genau wiedergeben kann. Wenn sogar ein Mensch 10,6 Prozent Fehler macht, kann man kaum erwarten, dass ein Lernalgorithmus viel besser abschneidet. Um zu beurteilen, ob der Trainingsfehler hoch ist, erweist es sich als nützlicher zu sehen, ob der Trainingsfehler viel höher ist als das Leistungsniveau eines Menschen, und in diesem Beispiel schneidet er nur 0,2 Prozent schlechter ab als der Mensch. Angesichts der Tatsache, dass Menschen tatsächlich sehr gut darin sind, Sprache zu erkennen, denke ich, dass ich ziemlich zufrieden wäre, wenn ich ein Spracherkennungssystem entwickeln könnte, das eine Fehlerquote von 10,6 Prozent erreicht, die der menschlichen Leistung entspricht. Es schneidet also nur ein bisschen schlechter ab als Menschen. Aber im Gegensatz dazu ist die Lücke bzw. der

Unterschied zwischen JCV und J-Train viel größer. Dort besteht tatsächlich eine Lücke von vier Prozent, während wir zuvor gesagt hatten, dass ein Fehler von vielleicht 10,8 Prozent eine hohe Verzerrung bedeutet. Wenn wir es mit der Leistung auf menschlicher Ebene vergleichen, sehen wir, dass der Algorithmus im Trainingssatz tatsächlich recht gut abschneidet, aber das größere Problem ist, dass der Kreuzvalidierungsfehler viel höher ist als der Trainingsfehler, weshalb ich zu dem Schluss kommen würde, dass dieser Algorithmus hat tatsächlich eher ein Varianzproblem als ein Bias-Problem. Es stellt sich heraus, dass es bei der Beurteilung, ob der Trainingsfehler hoch ist, oft nützlich ist, um ein Basisleistungsniveau zu ermitteln.




Mit Basisleistungsniveau meine ich, welches Fehlerniveau Sie vernünftigerweise hoffen können, dass Ihr Lernalgorithmus irgendwann erreicht. Eine gängige Methode zur Festlegung eines Basisleistungsniveaus besteht darin, zu messen, wie gut Menschen diese Aufgabe bewältigen können, da Menschen wirklich gut darin sind, Sprachdaten zu verstehen, Bilder zu verarbeiten oder Texte zu verstehen. Die Leistung auf menschlicher Ebene ist oft ein guter Maßstab, wenn Sie unstrukturierte Daten wie Audio, Bilder oder Texte verwenden. Eine andere Möglichkeit, ein Basisleistungsniveau abzuschätzen, besteht darin, dass es einen konkurrierenden Algorithmus gibt, vielleicht eine frühere Implementierung, die jemand anderes implementiert hat, oder sogar den Algorithmus eines Mitbewerbers, um ein Basisleistungsniveau festzulegen, wenn Sie das messen können, oder manchmal können Sie es auch basierend darauf erraten Vorerfahrung. Wenn Sie Zugriff auf dieses grundlegende Leistungsniveau haben, wie hoch ist dann die Fehlerquote, auf die Sie vernünftigerweise hoffen können, bzw. welches Leistungsniveau soll Ihr Algorithmus erreichen? Wenn Sie dann beurteilen, ob ein Algorithmus eine hohe Verzerrung oder Varianz aufweist, würden Sie das Basisleistungsniveau, den Trainingsfehler und den Kreuzvalidierungsfehler berücksichtigen. Die beiden wichtigsten zu messenden Größen sind dann: Was ist der Unterschied zwischen dem Trainingsfehler und dem Ausgangsniveau, das Sie erreichen möchten? Dies ist 0,2, und wenn dieser groß ist, dann würden Sie sagen, dass Sie ein Problem mit hoher Verzerrung haben. Anschließend werden Sie sich auch die Lücke zwischen Ihrem Trainingsfehler und Ihrem Kreuzvalidierungsfehler ansehen. Wenn diese groß ist, kommen Sie zu dem Schluss, dass ein Problem mit hoher Varianz vorliegt. Aus diesem Grund sind wir in diesem Beispiel zu dem Schluss gekommen, dass wir ein Problem mit hoher Varianz haben, während wir uns das zweite Beispiel ansehen. Wenn das Basisleistungsniveau; Das heißt, die Leistung auf menschlicher Ebene, Trainingsfehler und Kreuzvalidierungsfehler sehen so aus, dann beträgt diese erste Lücke 4,4 Prozent und es gibt also tatsächlich eine große Lücke. Der Trainingsfehler ist viel höher als das, was Menschen tun können und was wir erreichen wollen, während der Kreuzvalidierungsfehler nur ein wenig größer ist als der Trainingsfehler. Wenn Ihr Trainingsfehler und Ihr Kreuzvalidierungsfehler so aussehen, würde ich

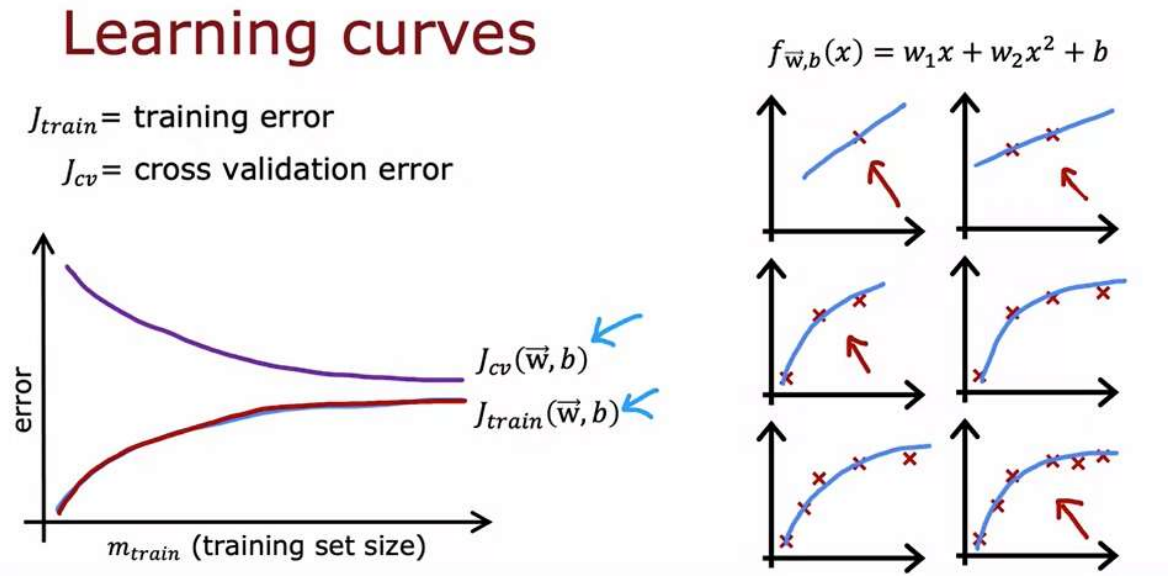
sagen, dass dieser Algorithmus eine hohe Verzerrung aufweist. Indem Sie sich diese Zahlen, Trainingsfehler und Kreuzvalidierungsfehler ansehen, können Sie intuitiv oder informell ein Gefühl dafür bekommen, inwieweit Ihr Algorithmus ein Problem mit hoher Verzerrung oder hoher Varianz aufweist. Um es zusammenzufassen: Diese Lücke zwischen diesen ersten beiden Zahlen gibt Ihnen einen Eindruck davon, ob Sie ein Problem mit hoher Verzerrung haben, und die Lücke zwischen diesen beiden Zahlen gibt Ihnen einen Eindruck davon, ob Sie ein Problem mit hoher Varianz haben. Manchmal kann das Basisleistungsniveau bei null Prozent liegen. Wenn Ihr Ziel darin besteht, eine perfekte Leistung zu erzielen, die über dem Basisleistungsniveau liegt, kann es bei null Prozent liegen, aber bei einigen Anwendungen wie der Spracherkennungsanwendung, bei denen einige Audiosignale nur verrauscht sind, kann das Basisleistungsniveau viel höher als Null sein. Mit der auf dieser Folie beschriebenen Methode können Sie besser erkennen, ob Ihr Algorithmus unter Voreingenommenheit oder Varianz leidet.




Übrigens ist es möglich, dass Ihre Algorithmen eine hohe Verzerrung und hohe Varianz aufweisen. Konkret: Wenn Sie solche Zahlen erhalten, ist die Lücke zwischen der Grundlinie und dem Trainingsfehler groß. Das wären 4,4 Prozent, und auch die Lücke zwischen Trainingsfehler und Kreuzvalidierungsfehler ist groß. Dieser liegt bei 4,7 Prozent. Wenn es so aussieht, werden Sie zu dem Schluss kommen, dass Ihr Algorithmus eine hohe Verzerrung und hohe Varianz aufweist, obwohl dies hoffentlich bei Ihren Lernanwendungen nicht so oft vorkommt. Zusammenfassend haben wir gesehen, dass die Prüfung, ob Ihr Trainingsfehler groß ist, eine Möglichkeit ist, festzustellen, ob Ihr Algorithmus eine hohe Verzerrung aufweist, aber bei Anwendungen, bei denen die Daten manchmal nur verrauscht sind und es unmöglich oder unrealistisch ist, jemals mit einer Null zu rechnen Wenn ein Fehler auftritt, ist es sinnvoll, dieses grundlegende Leistungsniveau festzulegen. Anstatt nur zu fragen, ob mein Trainingsfehler groß ist, können Sie fragen, ob mein Trainingsfehler groß ist im Verhältnis zu dem, was ich hoffentlich irgendwann erreichen kann, wie zum Beispiel: Ist mein Training groß im Verhältnis zu dem, was Menschen bei der Aufgabe leisten können? Dadurch erhalten Sie eine genauere Aussage darüber, wie weit Ihr Trainingsfehler von dem Ziel entfernt ist, das Sie erreichen möchten. Wenn Sie dann prüfen, ob Ihr Kreuzvalidierungsfehler viel größer ist als Ihr Trainingsfehler, erhalten Sie einen Eindruck davon, ob Ihr Algorithmus möglicherweise ebenfalls ein Problem mit hoher Varianz aufweist oder nicht. In der Praxis schaue ich mir diese Zahlen oft so an, um zu beurteilen, ob mein Lernalgorithmus ein Problem mit hoher Verzerrung oder hoher Varianz aufweist. Um unsere Intuition darüber, wie ein Lernalgorithmus funktioniert, weiter zu verfeinern, gibt es noch eine weitere Sache, die ich als nützlich empfand, nämlich die Lernkurve . Schauen wir uns im nächsten Video an, was das bedeutet.

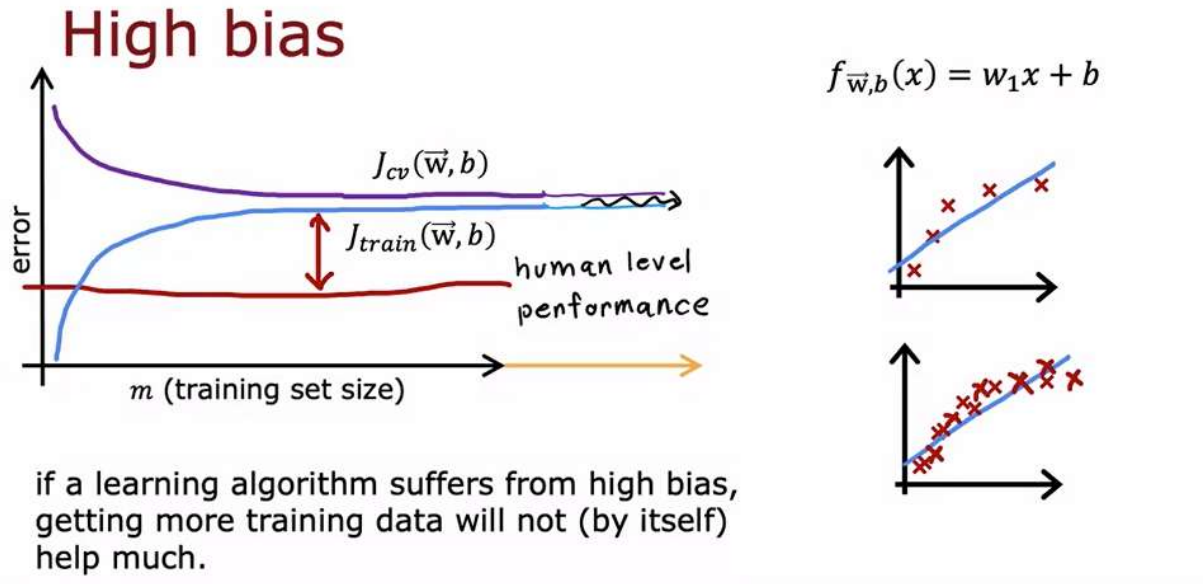
## Lernkurven

Lernkurven sind eine Möglichkeit zu verstehen, wie sich Ihr Lernalgorithmus in Abhängigkeit von der Menge an Erfahrung verhält, über die er verfügt. Mit Erfahrung meine ich beispielsweise die Anzahl der Trainingsbeispiele, über die er verfügt. Lass uns einen Blick darauf werfen. Lassen Sie mich die Lernkurven für ein Modell zeichnen, das wie folgt zu einer polynomialen quadratischen Funktion zweiter Ordnung passt. Ich werde sowohl J_cv, den Kreuzvalidierungsfehler, als auch J_train, den Trainingsfehler, grafisch darstellen. In dieser Abbildung ist die horizontale Achse m_train. Das ist die Größe des Trainingssatzes oder die Anzahl der Beispiele, aus denen der Algorithmus lernen kann. Auf der vertikalen Achse werde ich den Fehler darstellen. Mit Fehler meine ich entweder J_cv oder J_train. Beginnen wir mit der Darstellung des Kreuzvalidierungsfehlers. Es wird ungefähr so aussehen. So wird J_cv von (w, b) aussehen.



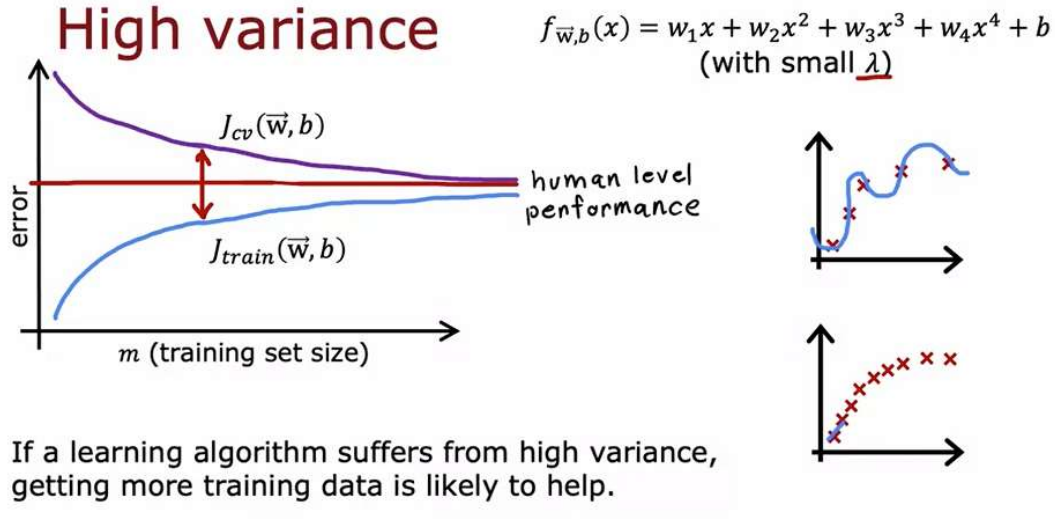

Es ist vielleicht keine Überraschung, dass mit m_train die Größe des Trainingssatzes größer wird, man dann ein besseres Modell lernt und so der Kreuzvalidierungsfehler sinkt . Lassen Sie uns nun J_train von (w, b) darstellen, wie der Trainingsfehler aussieht, wenn die Größe des Trainingssatzes größer wird. Es stellt sich heraus, dass der Trainingsfehler tatsächlich so aussehen wird. Dadurch wird die Größe des Trainingssatzes größer, und der Fehler des Trainingssatzes nimmt tatsächlich zu. Werfen wir einen Blick darauf, warum das so ist. Wir beginnen mit einem Beispiel, wenn Sie nur ein einziges Trainingsbeispiel haben. Nun, wenn Sie ein quadratisches Modell daran anpassen würden, können Sie am einfachsten eine gerade Linie oder eine Kurve anpassen und Ihr Trainingsfehler wäre Null. Wie wäre es, wenn Sie zwei Trainingsbeispiele wie dieses hätten? Nun, Sie können wieder eine gerade Linie anpassen und einen Trainingsfehler von null erreichen. Wenn Sie drei Trainingsbeispiele haben, kann die quadratische Funktion tatsächlich immer noch sehr gut dazu passen und einen Trainingsfehler von nahezu Null erzielen. Wenn Ihr Trainingssatz jedoch etwas größer wird, sagen wir, Sie haben vier Trainingsbeispiele, dann wird er größer etwas schwieriger, alle vier Beispiele perfekt zu passen. Möglicherweise erhalten Sie eine Kurve, die so aussieht, ganz gut, aber an ein paar Stellen liegen Sie hier und da etwas daneben. Wenn Sie Einträge haben, ist die Trainingssatzgröße auf vier gestiegen, der Trainingsfehler ist tatsächlich etwas gestiegen. Wie wäre es, wenn wir fünf Trainingsbeispiele hätten. Nun ja, man kann es ziemlich gut anpassen, aber es wird noch etwas schwieriger, alle perfekt anzupassen. Wir haben nicht einmal größere Schattierungssätze, es wird nur immer schwieriger, jedes einzelne Ihrer Trainingsbeispiele perfekt anzupassen. Um es noch einmal zusammenzufassen: Wenn Sie eine sehr kleine Anzahl von Trainingsbeispielen wie eins, zwei oder

sogar drei haben, ist es relativ leicht, einen Trainingsfehler von Null oder einen sehr kleinen zu erhalten, aber wenn Sie einen größeren Trainingssatz haben, ist es für die quadratische Funktion schwieriger, alle anzupassen Trainingsbeispiele perfekt. Aus diesem Grund steigt der Trainingsfehler mit zunehmender Trainingsmenge, da es schwieriger wird, alle Trainingsbeispiele perfekt anzupassen. Beachten Sie bei diesen Kurven noch etwas anderes: Der Kreuzvalidierungsfehler ist normalerweise höher als der Trainingsfehler, da Sie die Parameter an den Trainingssatz anpassen. Sie erwarten, dass Sie auf dem Trainingssatz zumindest ein bisschen besser abschneiden oder, wenn m klein ist, vielleicht sogar viel besser als auf dem Transvalidierungssatz. Schauen wir uns nun an, wie die Lernkurven für einen Durchschnitt mit hoher Verzerrung im Vergleich zu einem Durchschnitt mit hoher Varianz aussehen werden. Beginnen wir mit dem Fall der hohen Vorspannung oder der Unteranpassung. Denken Sie daran, dass ein Beispiel für einen hohen Bias darin besteht, eine lineare Funktion anzupassen, also eine Kurve, die so aussieht. Wenn Sie den Trainingsfehler grafisch darstellen, steigt der Trainingsfehler wie erwartet an. Tatsächlich könnte diese Kurve des Trainingsfehlers allmählich abflachen. Wir nennen es Plateau, was bedeutet, dass es nach einer Weile abflacht.




Denn je mehr Trainingsbeispiele Sie beim Anpassen der einfachen linearen Funktion erhalten, desto mehr ändert sich Ihr Modell nicht mehr. Es passt sich einer geraden Linie an, und auch wenn Sie immer mehr Beispiele erhalten, gibt es einfach nicht mehr viel zu ändern, weshalb der durchschnittliche Trainingsfehler nach einer Weile abflacht. Ebenso wird Ihr Kreuzvalidierungsfehler nach einer Weile abnehmen und auch fetter, weshalb J_cv wieder höher ist als J_train, aber J_cv wird tendenziell so aussehen. Das liegt daran, dass sich an der Straße, die Sie anpassen, nicht viel ändern wird, auch wenn Sie immer mehr Beispiele erhalten. Es ist einfach ein zu einfaches Modell, um in so viele Daten zu passen. Aus diesem Grund tendieren beide Kurven, J_cv und J_train, nach einer Weile dazu, abzuflachen. Wenn Sie ein Maß für dieses grundlegende Leistungsniveau haben, beispielsweise die Leistung auf menschlicher Ebene, dann liegen diese Werte tendenziell unter Ihrem J_train und Ihrem J_cv. Leistung auf menschlicher Ebene könnte so aussehen. Es besteht eine große Lücke zwischen dem Basisleistungsniveau und J_train, was unser Indikator dafür war, dass dieser Algorithmus eine hohe Verzerrung aufweist. Das heißt, man könnte hoffen, viel besser abzuschneiden, wenn wir nur eine komplexere Funktion als nur eine gerade Linie anpassen könnten. Eine interessante Sache an dieser Handlung ist, dass Sie sich fragen können: Was würde Ihrer Meinung nach passieren, wenn Sie einen viel größeren Trainingssatz hätten? Wie würde es aussehen, wenn wir noch weiter als rechts in diesem Diagramm wachsen könnten, Sie können wie folgt weiter

nach rechts gehen? Nun, Sie können sich vorstellen, dass, wenn Sie diese beiden Kurven nach rechts verlängern würden, sie beide abflachen würden und beide wahrscheinlich einfach weiterhin so flach bleiben würden. Ganz gleich, wie weit man sich von dieser Darstellung, diesen beiden Kurven, nach rechts erstreckt, sie werden nie irgendwie einen Weg finden, auf diese menschliche Leistungsebene abzusinken oder einfach so flach zu bleiben, praktisch für immer, egal wie groß das Training ist Set bekommt. Daraus ergibt sich die vielleicht etwas überraschende Schlussfolgerung, dass bei einem Lernalgorithmus mit hoher Verzerrung das Erhalten von mehr Trainingsdaten allein nicht so viel Hoffnung macht. Ich weiß, dass wir es gewohnt sind zu denken, dass es gut ist, mehr Daten zu haben, aber wenn Ihr Algorithmus eine hohe Verzerrung aufweist und Sie dann nur noch mehr Trainingsdaten hinzufügen, können Sie den Fehler allein dadurch nie verringern bewerte es so sehr. Aus diesem Grund wird die gerade lineare Anpassung einfach nicht viel besser werden, egal wie viele weitere Beispiele Sie zu dieser Abbildung hinzufügen. Aus diesem Grund lohnt es sich zu prüfen, ob Ihr Lernalgorithmus eine hohe Verzerrung aufweist, bevor Sie große Anstrengungen in das Sammeln weiterer Trainingsdaten investieren. Denn wenn dies der Fall ist, müssen Sie wahrscheinlich noch andere Dinge tun, als nur weitere Trainingsdaten hinzuzufügen. Schauen wir uns nun an, wie die Lernkurve für Lernalgorithmen mit hoher Varianz aussieht. Sie erinnern sich vielleicht daran, dass Sie eine Kurve erhalten, die so aussieht, wenn Sie das Vorgängerpolynom beispielsweise mit einem kleinen Lambda oder sogar einem Lambda gleich Null anpassen würden, und obwohl es sehr gut zu den Trainingsdaten passt, lässt es sich nicht verallgemeinern . Schauen wir uns nun an, wie eine Lernkurve in diesem Szenario mit hoher Varianz aussehen könnte. J train steigt mit zunehmender Größe des Trainingssatzes, sodass Sie eine Kurve erhalten, die so aussieht, und J cv wird viel höher sein, sodass Ihr Kreuzvalidierungsfehler viel höher ist als Ihr Trainingsfehler.

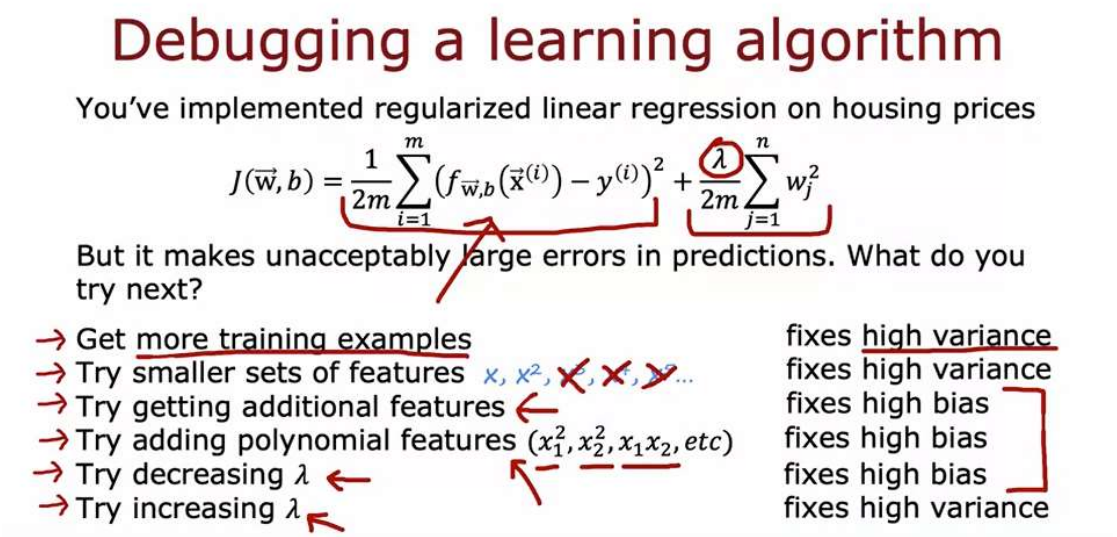



Aufgrund der Tatsache, dass hier eine große Lücke besteht, kann ich Ihnen sagen, dass diese hohe Varianz auf dem Trainingssatz viel besser abschneidet als auf Ihrem Kreuzvalidierungssatz. Wenn Sie ein grundlegendes Leistungsniveau, beispielsweise die Leistung auf menschlicher Ebene, grafisch darstellen, stellen Sie möglicherweise fest, dass hier J train manchmal sogar niedriger sein kann als die Leistung auf menschlicher Ebene, oder dass die Leistung auf menschlicher Ebene möglicherweise etwas geringer ist niedriger als dieser Wert. Aber wenn Sie den Trainingssatz übermäßig anpassen, können Sie den Trainingssatz möglicherweise so gut anpassen, dass ein unrealistisch niedriger Fehler auftritt, wie z. B. ein Fehler von Null in diesem Beispiel hier, was tatsächlich besser ist als die tatsächliche Leistung von Menschen Sie sind in der Lage, Immobilienpreise oder die Anwendung, an der Sie gerade arbeiten, vorherzusagen. Aber auch hier gilt: Ein Signal für eine hohe Varianz ist, ob J cv viel höher ist als J train. Wenn Sie eine hohe Varianz haben, könnte eine Erhöhung der Größe des Trainingssatzes sehr hilfreich sein, und insbesondere, wenn wir diese Kurven nach rechts

extrapolieren und den M-Zug erhöhen könnten, würde der Trainingsfehler weiter steigen, aber dann würde der Cross- Der Validierungsfehler tritt hoffentlich auf und nähert sich dem J-Zug. In diesem Szenario könnte es also möglich sein, einfach durch Erhöhen der Trainingssatzgröße den Kreuzvalidierungsfehler zu verringern und die Leistung Ihres Algorithmus immer besser zu machen, und dies ist anders als im Fall einer hohen Verzerrung, bei der Sie nur Folgendes tun Wenn Sie mehr Trainingsdaten erhalten, können Sie dadurch nicht viel über die Leistung Ihres Algorithmus lernen. Zusammenfassend lässt sich sagen: Wenn ein Lernalgorithmus unter einer hohen Varianz leidet, ist es wahrscheinlich hilfreich, mehr Trainingsdaten zu erhalten. Denn wenn man auf die rechte Seite dieser Kurve extrapoliert, sieht man, dass man damit rechnen kann, dass J cv weiter sinkt. In diesem Beispiel kann der Algorithmus allein durch das Abrufen von mehr Trainingsdaten von einem relativ hohen Kreuzvalidierungsfehler zu einer Leistung auf menschlichem Niveau kommen. Sie sehen, wenn Sie viel mehr Trainingsbeispiele hinzufügen und weiterhin das Polynom vierter Ordnung ausfüllen, können Sie einfach eine bessere Polynomanpassung vierter Ordnung an diese Daten erhalten, als nur eine sehr schwache Kurve oben. Wenn Sie eine Anwendung für maschinelles Lernen erstellen, können Sie bei Bedarf die Lernkurven zeichnen, d. h. Sie können verschiedene Teilmengen Ihrer Trainingssätze verwenden, und selbst wenn Sie beispielsweise 1.000 Trainingsbeispiele haben, können Sie ein Modell trainieren an nur 100 Trainingsbeispielen und schauen Sie sich den Trainingsfehler und den Kreuzvalidierungsfehler an, trainieren Sie sie dann an 200 Beispielen, halten Sie 800 Beispiele durch und verwenden Sie sie vorerst einfach nicht, und zeichnen Sie J train und J cv usw. auf, die Wiederholungen und Zeichnen Sie auf, wie die Lernkurve aussieht. Wenn wir es uns so vorstellen würden, dann könnte das eine weitere Möglichkeit für Sie sein, zu erkennen, ob Ihre Lernkurve eher einer hohen Voreingenommenheit oder einer hohen Varianz ähnelt. Ein Nachteil der Darstellung von Lernkurven wie dieser ist etwas, das ich getan habe, aber ein Nachteil ist, dass es rechenintensiv ist, so viele verschiedene Modelle mit Teilmengen unterschiedlicher Größe Ihres Trainingssatzes zu trainieren, sodass dies in der Praxis nicht durchgeführt wird So oft, aber trotzdem finde ich, dass mir dieses mentale visuelle Bild im Kopf, wie der Trainingssatz aussieht, manchmal hilft, darüber nachzudenken, was mein Lernalgorithmus meiner Meinung nach macht und ob er eine hohe Verzerrung oder eine hohe Varianz aufweist. Ich weiß, wir haben uns viel mit Bias und Varianz befasst. Kehren wir zu unserem früheren Beispiel zurück: Wenn Sie ein Modell für die Vorhersage von Immobilienpreisen trainiert haben, wie helfen Ihnen Bias und Varianz bei der Entscheidung, was als Nächstes zu tun ist? Kehren wir zu dem früheren Beispiel zurück, von dem ich hoffe, dass es für Sie jetzt viel mehr Sinn ergibt. Machen wir das im nächsten Video.

## Die Entscheidung, was ich als nächstes ausprobieren möchte, wurde noch einmal überdacht

Sie haben gesehen, wie das beim Betrachten von J train und Jcv der Trainingsfehler und der Kreuzvalidierungsfehler ist, oder vielleicht sogar beim Zeichnen einer Lernkurve. Sie können versuchen, ein Gefühl dafür zu bekommen, ob Ihr Lernalgorithmus einen hohen Bias oder eine hohe Varianz aufweist. Dies ist das Verfahren, das ich routinemäßig durchführe, wenn ich einen Lernalgorithmus trainiere. Ich schaue mir häufiger den Trainingsfehler und den Kreuzvalidierungsfehler an, um zu entscheiden, ob mein Algorithmus eine hohe Verzerrung oder eine hohe Varianz aufweist.



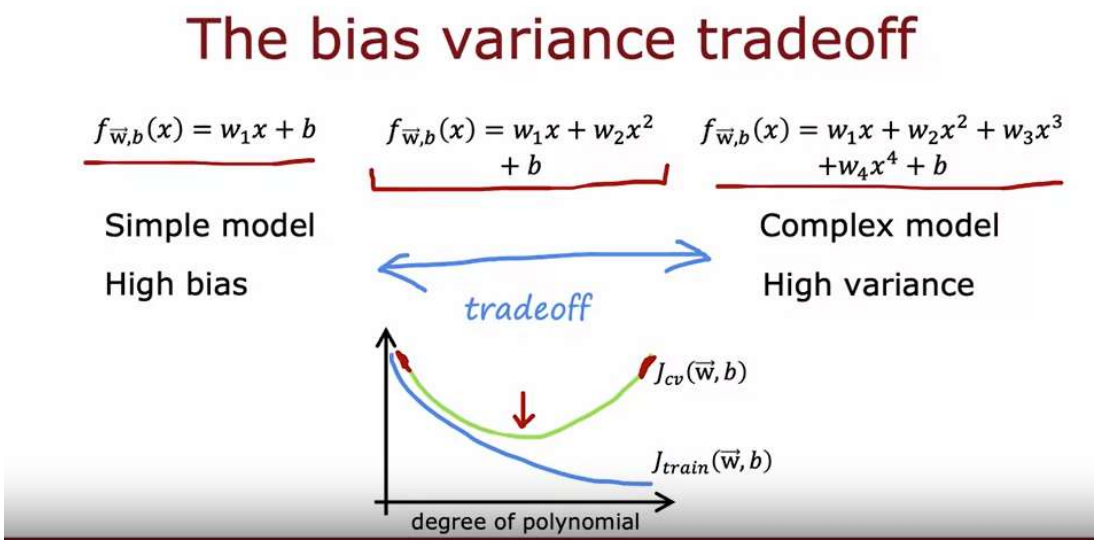

Es stellt sich heraus, dass Sie dadurch bessere Entscheidungen darüber treffen können, was Sie als Nächstes versuchen sollten, um die Leistung Ihres Lernalgorithmus zu verbessern. Schauen wir uns ein Beispiel an. Dies ist tatsächlich das Beispiel, das Sie zuvor gesehen haben. Wenn Sie eine regulierte lineare Regression zur Vorhersage von Immobilienpreisen implementiert haben, Ihr Algorithmus jedoch seit den Vorhersagen drei große Fehler aufweist, was versuchen Sie dann als Nächstes? Dies waren die sechs Ideen, die uns kamen, als wir uns diese Folie vorhin angeschaut hatten. Holen Sie sich weitere Trainingsbeispiele, probieren Sie kleine Funktionen, zusätzliche Funktionen usw. aus. Es stellt sich heraus, dass jedes dieser sechs Elemente entweder dazu beiträgt, ein Problem mit hoher Varianz oder hoher Verzerrung zu beheben. Insbesondere wenn Ihr Lernalgorithmus eine hohe Voreingenommenheit aufweist, sind drei dieser Techniken nützlich. Wenn Ihr Lernalgorithmus eine hohe Varianz aufweist, sind drei dieser Techniken hilfreich. Mal sehen, ob wir herausfinden können, welches welches ist. Zunächst erhalten Sie weitere Trainingsbeispiele. Wir haben im letzten Video gesehen, dass es wahrscheinlich nicht viel hilft, wenn Ihr Algorithmus eine hohe Verzerrung aufweist und wir nur mehr Trainingsdaten erhalten. Wenn Ihr Algorithmus jedoch eine hohe Varianz aufweist, beispielsweise eine übermäßige Anpassung an einen sehr kleinen Trainingssatz, dann wird es sehr hilfreich sein, mehr Trainingsbeispiele zu erhalten. Diese erste Option oder das Erhalten weiterer Trainingsbeispiele hilft, ein Problem mit hoher Varianz zu beheben. Wie wäre es mit den anderen fünf? Glauben Sie, dass Sie herausfinden können, welche der verbleibenden fünf Probleme mit hoher Verzerrung oder hoher Varianz beheben? Ich werde den Rest in diesem Video gleich durchgehen, aber wenn Sie möchten, können Sie das Video anhalten und sehen, ob Sie diese fünf anderen Dinge selbst durchdenken können. Sie können das Video gerne pausieren. Nur ein Scherz, das war meine Pause und nicht die Pause Ihres Videos. Aber im Ernst, wenn Sie es möchten, halten Sie das Video an und überlegen Sie, ob Sie es möchten oder nicht. Wir gehen diese Rezension gleich durch. Wie wäre es, wenn Sie einen kleineren Funktionsumfang ausprobieren würden? Wenn Ihr Lernalgorithmus manchmal über zu viele Funktionen verfügt, ist Ihr Algorithmus dadurch zu flexibel, um ihn an sehr komplizierte Modelle anzupassen. Das ist ein bisschen so, als ob man x, x quadriert, x kubisch, x^4, x^5 usw. hätte. Wenn Sie nur einige davon eliminieren würden, wäre Ihr Modell nicht so komplex und hätte keine so hohe Varianz. Wenn Sie vermuten, dass Ihr Algorithmus über viele Funktionen verfügt, die für die Vorhersage des Immobilienpreises nicht wirklich relevant oder hilfreich sind, oder wenn Sie vermuten, dass Sie sogar über etwas redundante Funktionen verfügten, kann das Eliminieren oder Reduzieren der Anzahl der Funktionen dazu beitragen, die Flexibilität Ihres Algorithmus zu verringern Algorithmus zur Überanpassung der Daten. Dies ist eine Taktik, die Ihnen helfen wird, hohe Varianz zu beheben. Konversation: Das Erhalten zusätzlicher Funktionen, das heißt einfach das Hinzufügen zusätzlicher

Funktionen, ist das Gegenteil davon, auf einen kleineren Satz an Funktionen umzusteigen. Dies wird Ihnen helfen, ein Problem mit hoher Voreingenommenheit zu beheben. Als konkretes Beispiel: Wenn Sie versuchen, den Preis des Hauses nur anhand der Größe vorherzusagen, stellt sich jedoch heraus, dass der Preis des Hauses auch von der Anzahl der Schlafzimmer, der Anzahl der Stockwerke und dem Alter abhängt des Hauses, dann wird der Algorithmus niemals so gut funktionieren, es sei denn, Sie fügen diese zusätzlichen Funktionen hinzu. Das ist ein Problem mit hoher Verzerrung, weil man mit dem Trainingssatz einfach nicht so gut abschneiden kann, wenn nur die Größe, also nur, wenn man dem Algorithmus mitteilt, wie viele Schlafzimmer und wie viele Stockwerke es gibt? Wie alt ist das Haus, dass es endlich genug Informationen hat, um beim Trainingsset noch besser abzuschneiden? Das Hinzufügen zusätzlicher Funktionen ist eine Möglichkeit, ein Problem mit hoher Verzerrung zu beheben. Das Hinzufügen von Polynommerkmalen ähnelt ein wenig dem Hinzufügen zusätzlicher Merkmale. Wenn es sich um lineare Funktionen handelt, kann die Drei-Linien-Funktion so gut zum Trainingssatz passen. Das Hinzufügen zusätzlicher Polynomfunktionen kann dabei helfen, den Trainingssatz besser zu bearbeiten, und eine hohe Verzerrung kann dadurch behoben werden, dass der Trainingssatz besser wird Problem. Dann bedeutet eine Verringerung von Lambda, einen niedrigeren Wert für den Regularisierungsparameter zu verwenden. Das bedeutet, dass wir diesem Begriff weniger Aufmerksamkeit schenken und ihm mehr Aufmerksamkeit schenken werden, um zu versuchen, im Trainingssatz bessere Ergebnisse zu erzielen. Auch dies hilft Ihnen, ein Problem mit hoher Verzerrung zu beheben. Schließlich ist die Erhöhung von Lambda das Gegenteil davon, aber das bedeutet, dass Sie die Daten überanpassen. Eine Erhöhung von Lambda macht Sinn, wenn eine Überanpassung des Trainingssatzes erfolgt und einfach zu viel Aufmerksamkeit auf die Anpassung an den Trainingssatz gelegt wird. Dies geht jedoch auf Kosten der Verallgemeinerung auf neue Beispiele, sodass eine Erhöhung von Lambda den Algorithmus dazu zwingen würde, eine glattere Funktion anzupassen Wiggly-Funktion und verwenden Sie diese, um ein Problem mit hoher Varianz zu beheben. Mir wurde klar, dass es sich bei dieser Folie um eine Menge Dinge handelte. Ich hoffe jedoch, dass Sie Folgendes mitnehmen können: Wenn Sie feststellen, dass Ihr Algorithmus eine hohe Varianz aufweist, gibt es zwei Möglichkeiten, dies zu beheben: Sie erhalten weder mehr Trainingsdaten noch vereinfachen Sie Ihr Modell. Mit der Vereinfachung des Modells meine ich, entweder einen kleineren Satz an Funktionen zu erhalten oder den Regularisierungsparameter Lambda zu erhöhen. Ihr Algorithmus ist weniger flexibel, um sehr komplexe, sehr wackelige Kurven anzupassen. Wenn Ihr Algorithmus hingegen eine hohe Verzerrung aufweist, bedeutet dies, dass er selbst im Trainingssatz nicht gut abschneidet. Wenn dies der Fall ist, bestehen die wichtigsten Korrekturen darin, Ihr Modell leistungsfähiger zu machen oder ihm mehr Flexibilität zu geben, um komplexere oder mit mir verbundene Funktionen zu integrieren. Dies lässt sich beispielsweise erreichen, indem man ihm zusätzliche Merkmale verleiht oder diese Polynommerkmale hinzufügt oder den Regularisierungsparameter Lambda verringert. Wie auch immer, falls Sie sich fragen, ob Sie eine hohe Verzerrung beheben sollten, indem Sie die Größe des Trainingssatzes reduzieren, hilft das nicht wirklich. Wenn Sie die Größe des Trainingssatzes reduzieren, passt der Trainingssatz besser zu Ihnen, aber das führt tendenziell zu einer Verschlechterung Ihres Kreuzvalidierungsfehlers und der Leistung Ihres Lernalgorithmus. Werfen Sie Trainingsbeispiele also nicht einfach weg, nur um zu versuchen, einen hohen Wert zu beheben Bias-Problem. Einer meiner Doktoranden aus Stanford sagte mir viele Jahre nach seinem Abschluss in Stanford einmal, dass er während seines Studiums in Stanford etwas über Voreingenommenheit und Varianz gelernt habe und das Gefühl hatte, dass er es verstanden habe, dass er es verstanden habe. Doch später, nach vielen Jahren Berufserfahrung in verschiedenen Unternehmen, wurde ihm klar, dass Voreingenommenheit und Varianz zu den Konzepten gehören, deren Erlernen nur kurze Zeit in Anspruch nimmt, deren Beherrschung jedoch ein Leben lang dauert. Das waren seine genauen Worte. Voreingenommenheit und Varianz sind eine dieser sehr wirkungsvollen Ideen. Wenn ich Lernalgorithmen trainiere, versuche ich fast immer herauszufinden, ob es sich um eine hohe Verzerrung oder eine hohe Varianz handelt. Aber die Art

und Weise, wie Sie das systematisch angehen, ist etwas, das Sie durch wiederholtes Üben immer besser beherrschen werden. Sie werden jedoch feststellen, dass das Verständnis dieser Ideen Ihnen dabei helfen wird, bei der Entwicklung eines Lernalgorithmus viel effektiver zu entscheiden, was Sie als Nächstes versuchen möchten. Nun, ich weiß, dass wir in diesem Video viel durchgemacht haben, und wenn Sie das Gefühl haben, Junge, das ist hier nicht verloren, ist das in Ordnung, machen Sie sich darüber keine Sorgen. Später in dieser Woche wird es in den Übungslaboren und Übungsquizzen auch zusätzliche Möglichkeiten geben, diese Ideen durchzugehen, damit Sie zusätzliche Übung erhalten. Wir denken über Voreingenommenheit und Varianz verschiedener Lernalgorithmen nach. Wenn es Ihnen im Moment so vorkommt, als ob vieles in Ordnung ist, können Sie diese Ideen später in dieser Woche in die Praxis umsetzen und dabei hoffentlich Ihr Verständnis dafür vertiefen. Bevor wir fortfahren, sind Voreingenommenheit und Varianz auch sehr nützlich, wenn man darüber nachdenkt, wie man ein neuronales Netzwerk trainiert. Im nächsten Video werfen wir einen Blick auf diese Konzepte, die auf das Training neuronaler Netzwerke angewendet werden. Kommen wir zum nächsten Video.

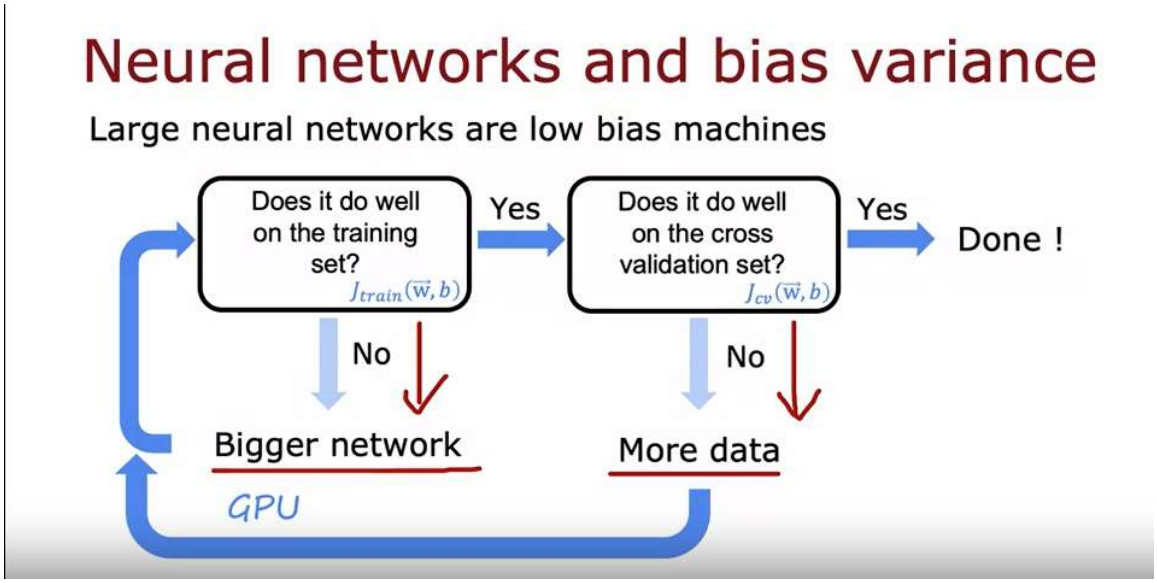
## Bias/Varianz und neuronale Netze

Es wurde festgestellt, dass sowohl eine hohe Verzerrung als auch eine hohe Varianz in dem Sinne schlecht sind, dass sie die Leistung Ihres Algorithmus beeinträchtigen. Einer der Gründe dafür, dass neuronale Netze so erfolgreich sind, sind Ihre Netze, zusammen mit der Idee von Big Data oder hoffentlich großen Datensätzen. Es hat uns neue Möglichkeiten eröffnet, sowohl mit hoher Voreingenommenheit als auch mit hoher Varianz umzugehen. Lass uns einen Blick darauf werfen.



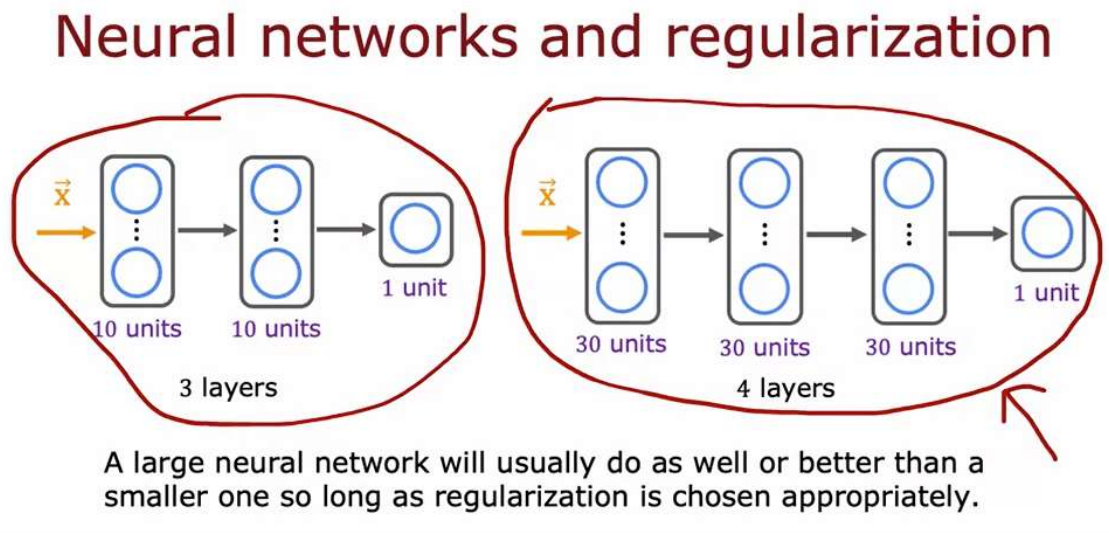

Sie haben gesehen, dass, wenn Sie Polynome unterschiedlicher Ordnung an einen Datensatz anpassen, Sie dann ein lineares Modell wie dieses links anpassen würden. Sie haben ein ziemlich einfaches Modell, das eine hohe Verzerrung aufweisen kann, während Sie ein komplexes Modell anpassen müssten, dann könnten Sie unter einer hohen Varianz leiden. Und es gibt diesen Kompromiss zwischen Bias und Varianz, und in unserem Beispiel war es die Wahl eines Polynoms zweiter Ordnung, das Ihnen hilft, einen Kompromiss einzugehen und ein Modell mit dem geringstmöglichen Kreuzvalidierungsfehler auszuwählen. Und so sprachen die Ingenieure des maschinellen Lernens vor den Tagen der neuronalen Netze viel über diesen Bias-Varianz-Kompromiss, bei dem man die Komplexität, die den Grad des Polynoms darstellt, ausgleichen muss.

Oder der Regularisierungsparameter länger, damit sowohl Käufer als auch Varianten nicht zu hoch sind. Und wenn Sie Ingenieure des maschinellen Lernens über den Kompromiss zwischen Bias und Varianz sprechen hören. Das ist es, worauf sie sich beziehen: Wenn ein Modell zu einfach ist, hat es eine hohe Verzerrung, ein zu komplexes Modell eine hohe Varianz. Und Sie müssen einen Kompromiss zwischen diesen beiden schlechten Dingen finden, um das bestmögliche Ergebnis zu erzielen. Aber es stellt sich heraus, dass Ihre Netzwerke uns das ganze Dilemma, Voreingenommenheit und Varianz mit einigen Vorbehalten abwägen zu müssen, aus der Welt schaffen. Und es stellt sich heraus, dass große neuronale Netze, wenn sie auf kleinen, mittelgroßen Datensätzen trainiert werden, Maschinen mit geringer Voreingenommenheit sind. Damit meine ich Folgendes: Wenn Sie Ihr neuronales Netzwerk groß genug machen, können Sie Ihr Trainingsset fast immer gut anpassen. Solange Ihr Trainingsumfang nicht riesig ist. Und das bedeutet, dass wir damit ein neues Rezept erhalten, mit dem wir versuchen können, je nach Bedarf Voreingenommenheit oder Varianz zu verringern, ohne wirklich einen Kompromiss zwischen beiden eingehen zu müssen. Lassen Sie mich ein einfaches Rezept mit Ihnen teilen, das nicht immer anwendbar ist. Aber wenn es liefert, kann es sehr leistungsfähig sein, ein genaues Modell mithilfe eines neuronalen Netzwerks zu erhalten, das zunächst Ihren Algorithmus auf Ihrem Trainingssatz trainiert und dann fragt, ob er auf dem Trainingssatz gut abschneidet. Messen Sie also Jtrain und prüfen Sie, ob es hoch ist, und zwar im Verhältnis zur Leistung auf menschlicher Ebene oder zu einem Grundleistungsniveau. Wenn es nicht gut läuft, liegt ein Problem mit hoher Verzerrung und einem hohen Zugfehler vor.

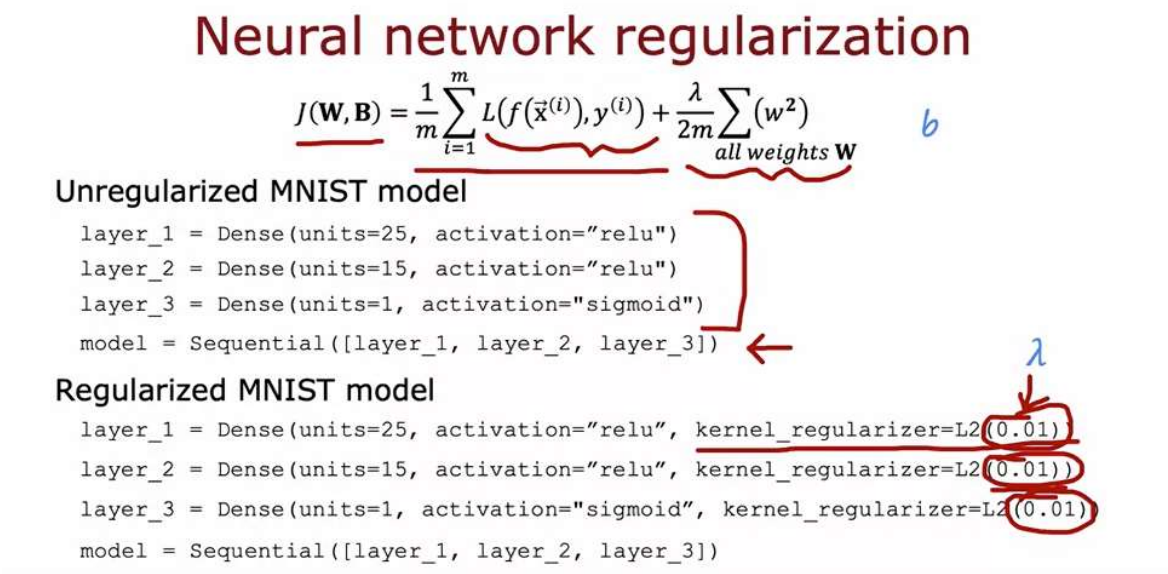



Und eine Möglichkeit, Verzerrungen zu reduzieren, besteht darin, einfach ein größeres neuronales Netzwerk zu verwenden, und mit größerem neuronalen Netzwerk meine ich entweder mehr verborgene Schichten oder mehr verborgene Einheiten pro Schicht. Und Sie können diese Schleife dann weiter durchlaufen und Ihr neuronales Netzwerk immer größer machen, bis es auf dem Trainingssatz gut funktioniert. Dies bedeutet, dass in Ihrem Trainingssatz die Fehlerquote erreicht wird, die in etwa mit der angestrebten Fehlerquote vergleichbar ist, bei der es sich um eine Leistung auf menschlicher Ebene handeln könnte. Nachdem es auf das Trainingsset gefallen ist, lautet die Antwort auf diese Frage „Ja“. Sie fragen dann, ob der Trans- Validierungssatz nicht gut funktioniert . Mit anderen Worten: Weist es eine hohe Varianz auf? Wenn die Antwort „Nein“ lautet, können Sie daraus schließen, dass der Algorithmus eine hohe Varianz aufweist, da er keinen Trainingssatz für den Kreuzvalidierungssatz ausführen möchte. Diese große Lücke in Jcv und Jtrain weist also darauf hin, dass Sie wahrscheinlich ein Problem mit hoher Varianz haben, und wenn Sie ein Problem mit hoher Varianz haben, können Sie versuchen, es zu beheben, indem Sie mehr Daten erhalten. Wollen Sie nur den Trainingssatz, um weitere Daten zu erhalten, das Modell erneut zu trainieren und noch

einmal zu überprüfen? Wenn nicht, verwenden Sie ein größeres Netzwerk, oder prüfen Sie, ob dies der Fall ist, wenn die Cross-Fundament-Einstellung erfolgt, und wenn nicht, erhalten Sie mehr Daten. Und wenn Sie diese Schleife so lange durchgehen können, bis sie schließlich im Kreuzvalidierungssatz gut funktioniert. Dann sind Sie wahrscheinlich fertig, denn jetzt haben Sie ein Modell, das im Kreuzvalidierungssatz gut funktioniert und sich hoffentlich auch auf neue Beispiele verallgemeinern lässt. Nun gibt es natürlich Einschränkungen bei der Anwendung dieses Rezepts. Durch das Training eines größeren neuronalen Netzwerks wird die Verzerrung nicht verringert, aber irgendwann wird es rechenintensiv. Aus diesem Grund wurde der Aufstieg neuronaler Netze durch den Aufstieg sehr schneller Computer, insbesondere GPUs oder Grafikprozessoren, erheblich gefördert. Hardware, die traditionell zur Beschleunigung von Computergrafiken verwendet wird, hat sich jedoch auch für die Beschleunigung neuronaler Netze als sehr nützlich erwiesen. Aber selbst mit Hardwarebeschleunigern ab einem bestimmten Punkt sind die neuronalen Netze so groß, dass das Training so lange dauert, dass es nicht mehr realisierbar ist. Und die andere Einschränkung sind natürlich mehr Daten. Manchmal kann man nur eine begrenzte Datenmenge erhalten, und ab einem bestimmten Punkt ist es schwierig, noch mehr Daten zu erhalten. Aber ich denke, dieses Rezept erklärt einen Großteil des Aufstiegs von Deep Learning in den letzten Jahren, und zwar für Anwendungen, bei denen man Zugriff auf viele Daten hat. Wenn Sie dann in der Lage sind, große neuronale Netze zu trainieren, können Sie schließlich bei vielen Anwendungen eine ziemlich gute Leistung erzielen. Eine Sache, die in dieser Folie implizit enthalten war und möglicherweise nicht offensichtlich war, ist, dass Sie bei der Entwicklung eines Lernalgorithmus manchmal feststellen, dass Sie eine hohe Voreingenommenheit haben. In diesem Fall erweitern Sie beispielsweise Ihr neuronales Netzwerk. Aber nachdem Sie Ihr neuronales Netzwerk erweitert haben, stellen Sie möglicherweise fest, dass die Varianz hoch ist. In diesem Fall können Sie andere Dinge tun, beispielsweise mehr Daten sammeln. Und während der Stunden, Tage oder Wochen, in denen Sie einen Algorithmus für maschinelles Lernen entwickeln, kann es zu unterschiedlichen Zeitpunkten zu hohen Käufen oder einer hohen Varianz kommen. Und es kann sich ändern, aber es hängt davon ab, ob Ihr Algorithmus zu diesem Zeitpunkt eine hohe Verzerrung oder eine hohe Varianz aufweist. Dann kann Ihnen das helfen, Hinweise darauf zu geben, was Sie als Nächstes versuchen sollten. Wenn Sie Ihr neuronales Netzwerk trainieren, haben mich die Leute schon einmal gefragt: „Hey Andrew, was ist, wenn mein neuronales Netzwerk zu groß ist?“ Wird dadurch ein Problem mit hoher Varianz entstehen? Es stellt sich heraus, dass ein großes neuronales Netzwerk mit einer gut gewählten Regularisierung in der Regel genauso gut oder besser abschneidet als ein kleineres.



Wenn Sie beispielsweise über ein kleines neuronales Netzwerk wie dieses verfügen und zu einem viel größeren neuronalen Netzwerk wie diesem wechseln würden, würden Sie annehmen, dass das Risiko einer Überanpassung erheblich steigt. Aber es stellt sich heraus, dass, wenn man dieses größere neuronale Netzwerk entsprechend regulieren würde, dieses größere neuronale Netzwerk normalerweise mindestens genauso gut oder besser abschneiden würde als das kleinere. Solange die Regularisierung angemessen gewählt wurde. Mit anderen Worten: Es schadet fast nie, ein neuronales Netzwerk zu starten, solange man es entsprechend reguliert, mit einer Einschränkung: Wenn man das größere neuronale Netzwerk trainiert, wird es rechenintensiver und teurer. Der Hauptgrund dafür, dass es weh tut, ist, dass es Ihr Training und Ihren Inferenzprozess verlangsamt und ein neuronales Netzwerk ganz kurz reguliert.



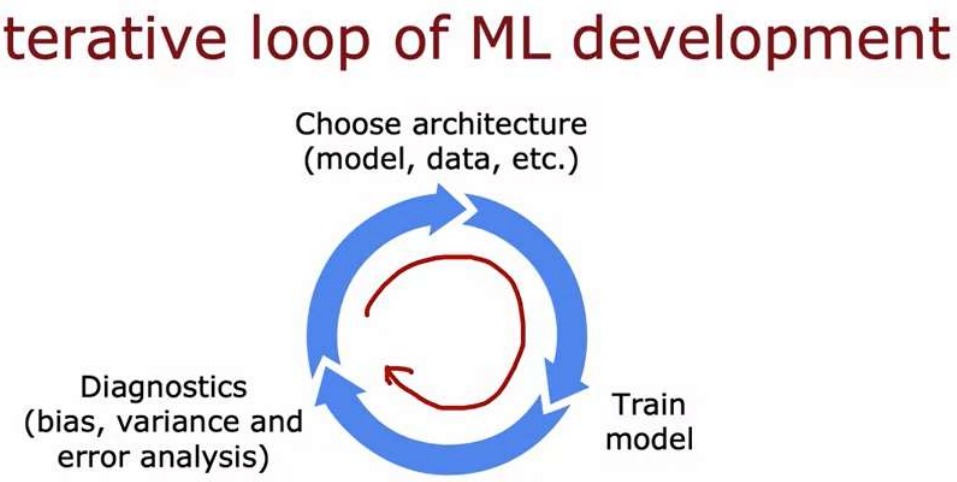

Dies tun Sie, wenn die Kostenfunktion für Ihr neuronales Netzwerk der durchschnittliche Verlust ist und das letzte Jahr daher ein quadratischer Fehler oder ein Logistikverlust sein könnte. Dann sieht der Regularisierungsterm für ein neuronales Netzwerk ungefähr so aus, wie man es erwarten würde, und ist länger als zwei m-mal die Summe von w zum Quadrat, wobei es sich um einen Song über immer W im neuronalen Netzwerk handelt, und ähnelt der Regularisierung für lineare Regression und logistische Regression Normalerweise regulieren wir die Parameter im neuronalen Netzwerk nicht, obwohl es in der Praxis kaum einen Unterschied macht, ob Sie dies tun oder nicht. Und die Art und Weise, wie Sie die Regularisierung in Tensorflow implementieren würden, ist, sich daran zu erinnern, dass dies der Code für die Implementierung eines unregulierten, handgeschriebenen Ziffernklassifizierungsmodells von Rised war. Wir erstellen auf diese Weise drei Schichten mit der Aktivierung mehrerer Anpassungseinheiten und erstellen dann ein sequentielles Modell mit den drei Schichten. Wenn Sie eine Regularisierung hinzufügen möchten, fügen Sie einfach diesen zusätzlichen Term Colonel Regularize A gleich l hinzu. zwei und dann 0,01, wobei das der Wert von „länger“ ist. Sie können jedoch tatsächlich unterschiedliche Lambda-Werte für verschiedene Schichten auswählen. Der Einfachheit halber können Sie jedoch wie folgt denselben Lambda-Wert für alle Gewichtungen und alle verschiedenen Schichten wählen. Und dann können Sie die Regularisierung in Ihrem neuronalen Netzwerk implementieren. Um also zwei Takeaways zusammenzufassen, hoffe ich, dass Sie aus diesem Video eines sind. Es schadet kaum, über ein größeres neuronales Netzwerk zu verfügen, solange Sie es entsprechend regulieren. Eine Einschränkung besteht darin, dass ein größeres neuronales Netzwerk Ihren Algorithmus verlangsamen kann. Vielleicht ist das die eine Art, in der es weh tut, aber es sollte die Leistung Ihres Albums nicht im Wesentlichen beeinträchtigen und könnte es sogar erheblich verbessern. Und zweitens, solange Ihr Trainingssatz nicht zu groß ist. Dann ist ein neues Netzwerk, insbesondere ein großes neuronales Netzwerk, oft eine Maschine mit

geringer Vorspannung. Es passt einfach sehr gut zu sehr komplizierten Funktionen, weshalb ich beim Training neuronaler Netze oft mit verschiedenen Problemen und nicht mit Bias-Problemen zu kämpfen habe, zumindest wenn das neuronale Netz groß genug ist. Der Aufstieg des Deep Learning hat also die Art und Weise, wie Praktiker des maschinellen Lernens über Voreingenommenheit und Varianz denken, wirklich verändert. Allerdings ist das, was Sie als Nächstes tun, oft sehr hilfreich, selbst wenn Sie ein neuronales Netzwerk trainieren, das Bias und Varianz misst und dies für Gott nutzt. Das war's also mit Voreingenommenheit und Varianz. Kommen wir zum nächsten Video.

Wir werden alle Ideen, die wir gelernt haben, aufgreifen und prüfen, wie sie in den Entwicklungsprozess maschineller Lernsysteme passen. Und ich hoffe, dass wir viele dieser Teile zusammenfügen, um Ihnen praktische Ratschläge zu geben, wie Sie bei der Entwicklung Ihrer maschinellen Lernsysteme schnell vorankommen können

## Iterative Schleife der ML-Entwicklung

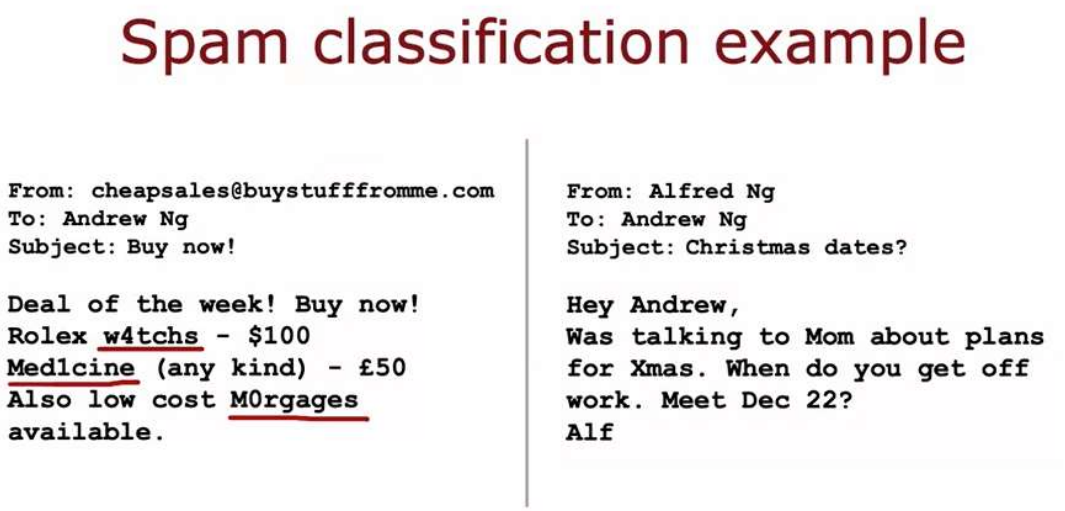
In den nächsten Videos möchte ich mit Ihnen teilen, wie es ist, den Prozess der Entwicklung eines maschinellen Lernsystems zu durchlaufen, damit Sie, wenn Sie dies selbst tun, hoffentlich in der Lage sind, gute Entscheidungen zu treffen in vielen Phasen des maschinellen Lernentwicklungsprozesses. Werfen wir zunächst einen Blick auf die iterative Schleife der maschinellen Lernentwicklung.



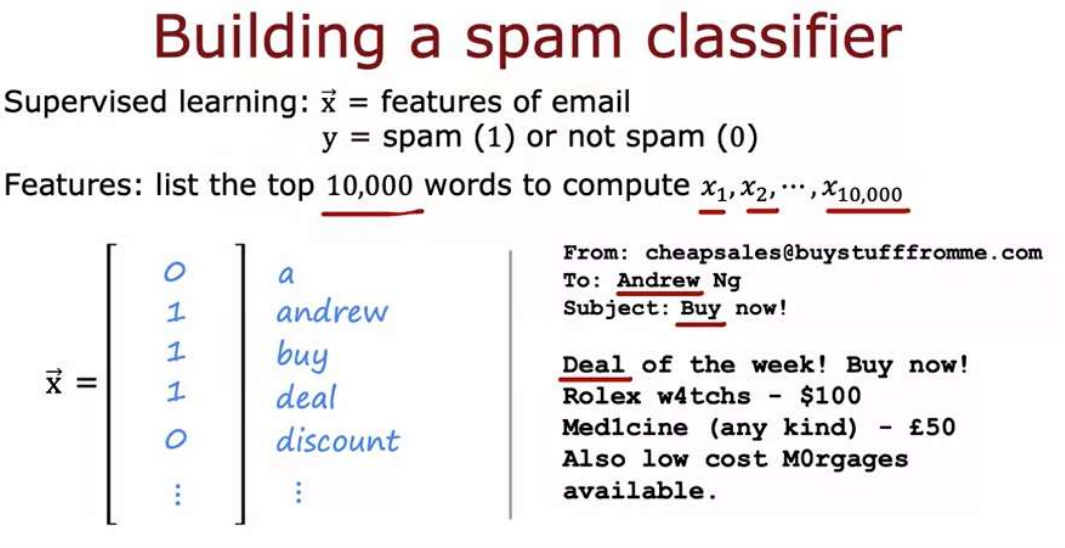

So wird sich die Entwicklung eines maschinellen Lernmodells oft anfühlen. Zunächst entscheiden Sie über die Gesamtarchitektur Ihres Systems. Das bedeutet, dass Sie Ihr Modell für maschinelles Lernen auswählen und entscheiden müssen, welche Daten verwendet werden sollen, möglicherweise die Hyperparameter auswählen und so weiter. Basierend auf diesen Entscheidungen würden Sie dann ein Modell implementieren und trainieren. Wie ich bereits erwähnt habe: Wenn Sie ein Modell zum ersten Mal trainieren, wird es fast nie so gut funktionieren, wie Sie es möchten. Der nächste Schritt, den ich dann empfehle, ist die Implementierung oder Betrachtung einiger Diagnosen, z. B. der Betrachtung der Verzerrung und Varianz Ihres Algorithmus sowie etwas, das wir im nächsten Video namens Fehleranalyse sehen werden.

Basierend auf den Erkenntnissen aus der Diagnose können Sie dann Entscheidungen treffen, z. B. ob Sie Ihr neuronales Netzwerk vergrößern oder den Lambda-Regularisierungsparameter ändern möchten oder vielleicht mehr Daten hinzufügen oder weitere Features hinzufügen oder Features

entfernen möchten. Dann durchlaufen Sie diese Schleife noch einmal mit Ihrer neuen Architekturwahl, und oft sind mehrere Iterationen durch diese Schleife erforderlich, bis Sie die gewünschte Leistung erreicht haben. Schauen wir uns ein Beispiel für den Aufbau eines E-Mail-Spam-Klassifikators an.

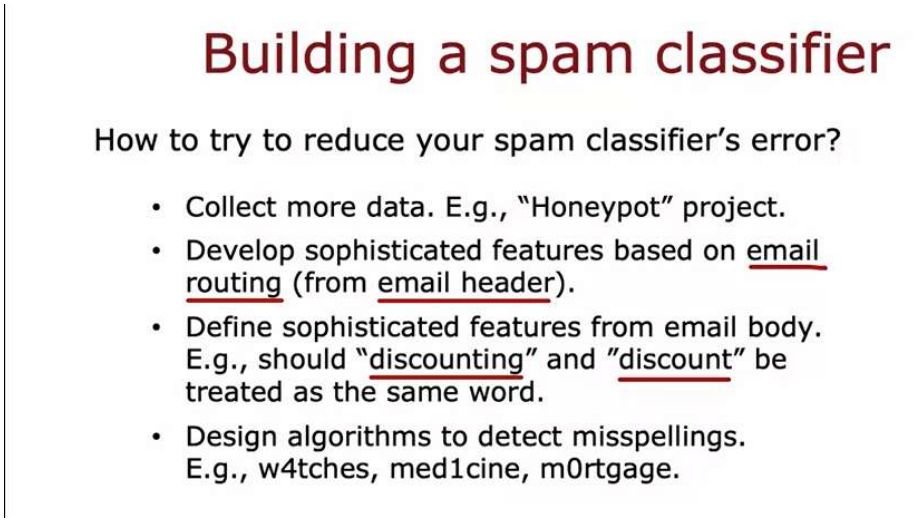



Ich denke, dass viele von uns E-Mail-Spam leidenschaftlich hassen, und das ist ein Problem, an dem ich vor Jahren gearbeitet habe und an dem ich vor Jahren auch beteiligt war, als ich eine Anti-Spam-Konferenz ins Leben rief. Das Beispiel links zeigt, wie eine stark spammige E-Mail aussehen könnte. Mittlerweile der Deal der Woche, Rolex-Uhren. Manchmal schreiben Spammer absichtlich Wörter wie „Uhren“, „Medizin“ und „Hypotheken“ falsch, um eine Spam-Erkennung auszuschalten. Im Gegensatz dazu handelt es sich bei dieser E-Mail auf der rechten Seite um eine tatsächliche E-Mail, die ich einmal von meinem jüngeren Bruder Alfred über ein Treffen zu Weihnachten erhalten habe. Wie erstellt man einen Klassifikator, um Spam-E-Mails von Nicht-Spam-E-Mails zu unterscheiden? Eine Möglichkeit wäre, einen überwachten Lernalgorithmus zu trainieren, bei dem die Eingabemerkmale x die Merkmale einer E-Mail sind und die Ausgabebezeichnung y eins oder null ist, je nachdem, ob es sich um Spam oder Nicht-Spam handelt.



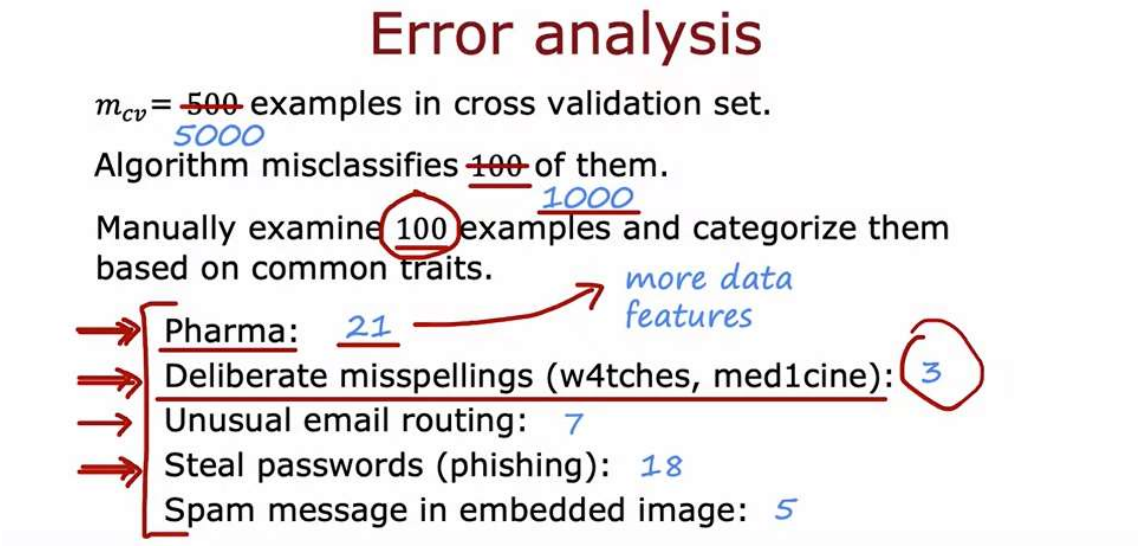

Diese Anwendung ist ein Beispiel für die Textklassifizierung, da Sie ein Textdokument, das eine E-Mail ist, nehmen und versuchen, es entweder als Spam oder Nicht-Spam zu klassifizieren. Eine

Möglichkeit, die Merkmale der E-Mail zu konstruieren, wäre beispielsweise, die wichtigsten 10.000 Wörter in der englischen Sprache oder in einem anderen Wörterbuch zu nehmen und sie zur Definition der Merkmale x_1, x_2 bis x_10.000 zu verwenden. Wenn wir beispielsweise diese E-Mail auf der rechten Seite sehen, lautet die Liste der Wörter, die wir haben, „Andrew, Kaufangebot, Rabatt“ und so weiter. Anhand der E-Mail auf der rechten Seite würden wir diese Funktionen dann beispielsweise auf 0 oder 1 setzen, je nachdem, ob dieses Wort vorkommt oder nicht. Das Wort a kommt nicht vor. Das Wort Andrew erscheint. Das Wort „Kaufen“ kommt vor, „Deal“ kommt vor, „Rabatt“ nicht usw. Sie können also 10.000 Funktionen dieser E-Mail erstellen. Es gibt viele Möglichkeiten, einen Merkmalsvektor zu erstellen. Eine andere Möglichkeit wäre, diese Zahlen nicht nur 1 oder 0 sein zu lassen, sondern tatsächlich zu zählen, wie oft ein bestimmtes Wort in der E-Mail vorkommt. Wenn „buy“ zweimal angezeigt wird, möchten Sie dies vielleicht auf 2 setzen, die Einstellung aber nur auf 1 oder 0. Das funktioniert eigentlich recht gut. Angesichts dieser Merkmale können Sie dann einen Klassifizierungsalgorithmus wie ein logistisches Regressionsmodell oder ein neuronales Netzwerk trainieren, um y anhand dieser Merkmale x vorherzusagen. Wenn Ihr erstes Modell nach dem Training nicht so gut funktioniert, wie Sie es sich wünschen, haben Sie höchstwahrscheinlich mehrere Ideen zur Verbesserung der Leistung des Lernalgorithmus. Beispielsweise ist es immer verlockend, mehr Daten zu sammeln. Tatsächlich habe ich Freunde, die an sehr großen Honeypot-Projekten gearbeitet haben.

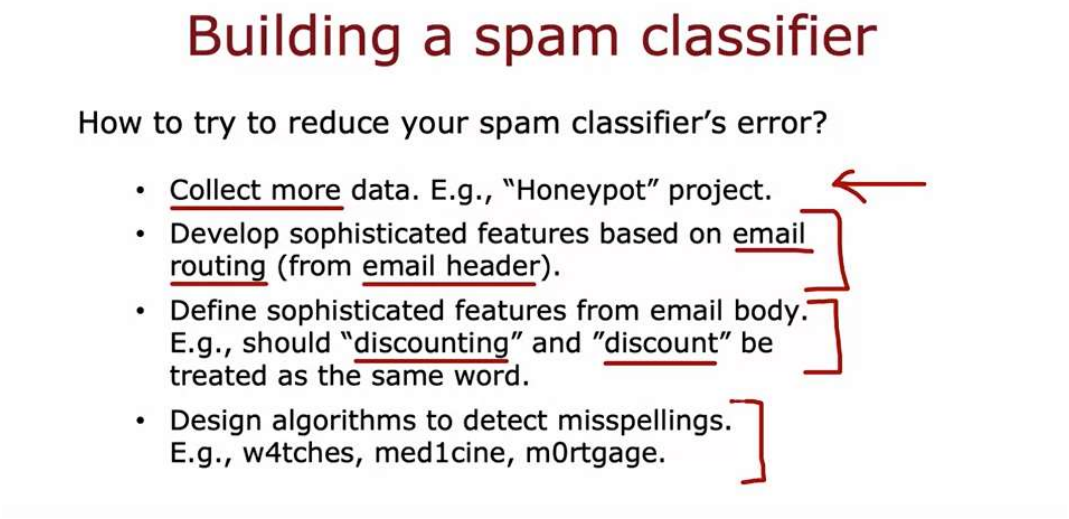



Hierbei handelt es sich um Projekte, die eine große Anzahl gefälschter E-Mail-Adressen erstellen und versuchen, diese gefälschten E-Mail-Adressen absichtlich in die Hände von Spammern zu bringen, sodass wir, wenn sie Spam-E-Mails an diese gefälschten E-Mails senden, wissen, dass es sich um Spam-E-Mail-Nachrichten handelt, und das ist auch der Fall eine Möglichkeit, viele Spam-Daten zu erhalten. Oder Sie möchten an der Entwicklung ausgefeilterer Funktionen basierend auf dem E-Mail-Routing arbeiten. E-Mail-Routing bezieht sich auf die Reihenfolge der Rechendienste. Manchmal hat die E-Mail auf der ganzen Welt den ganzen Weg zurückgelegt, um Sie zu erreichen, und E-Mails enthalten tatsächlich sogenannte E-Mail-Header-Informationen. Dabei handelt es sich um Informationen, die verfolgen, wie die E-Mail über verschiedene Server und Netzwerke gelaufen ist, um ihren Weg zu Ihnen zu finden. Manchmal kann der Weg, den eine E-Mail zurückgelegt hat, Aufschluss darüber geben, ob sie von einem Spammer gesendet wurde oder nicht. Oder Sie arbeiten daran, aus dem E-Mail-Text, also dem E-Mail-Text, ausgefeiltere Funktionen zu entwickeln. In den Funktionen, über die ich letztes Mal gesprochen habe, werden Rabatt und Rabatt möglicherweise als unterschiedliche Wörter behandelt, und vielleicht sollten sie als dieselben Wörter behandelt werden. Oder Sie entscheiden sich möglicherweise für die Entwicklung von Algorithmen zur Erkennung von Rechtschreibfehlern oder absichtlichen Rechtschreibfehlern wie „Uhren“, „Medizin“ und „Hypothek“.

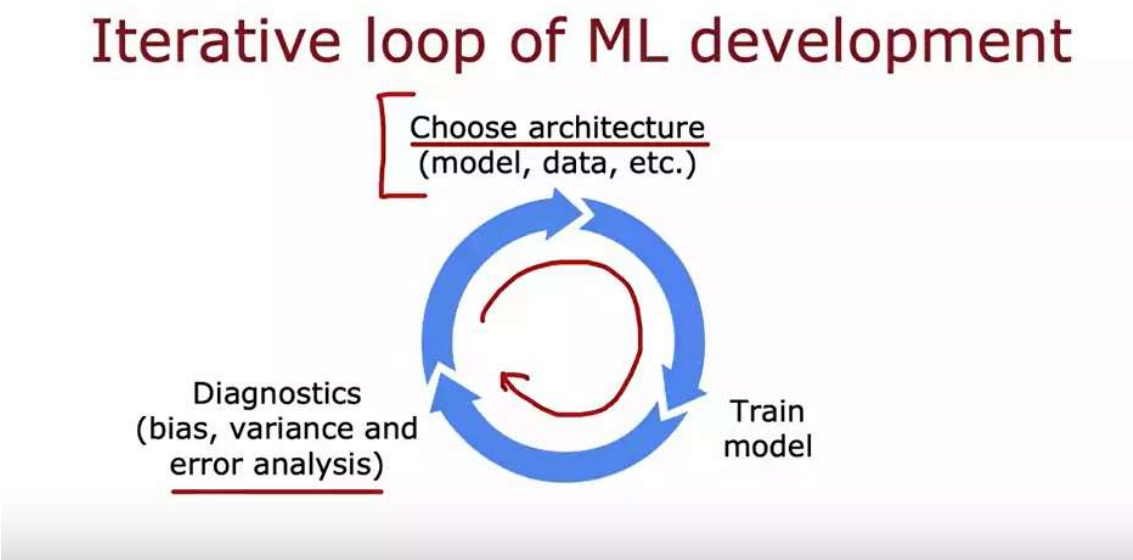
Auch dies könnte Ihnen bei der Entscheidung helfen, ob eine E-Mail Spam ist. Wie können Sie angesichts all dieser und möglicherweise noch weiterer Ideen entscheiden, welche dieser Ideen vielversprechender für die Arbeit sind? Denn die Wahl des vielversprechenderen Wegs nach vorn kann Ihr Projekt leicht um das Zehnfache beschleunigen, verglichen mit der Wahl einiger der weniger vielversprechenden Richtungen.




Wir haben beispielsweise bereits gesehen, dass, wenn Ihr Algorithmus eher eine hohe Verzerrung als eine hohe Varianz aufweist, es möglicherweise nicht die fruchtbarste Richtung ist, monatelang an einem Honeypot-Projekt zu arbeiten. Wenn Ihr Algorithmus jedoch eine hohe Varianz aufweist, könnte das Sammeln weiterer Daten sehr hilfreich sein.



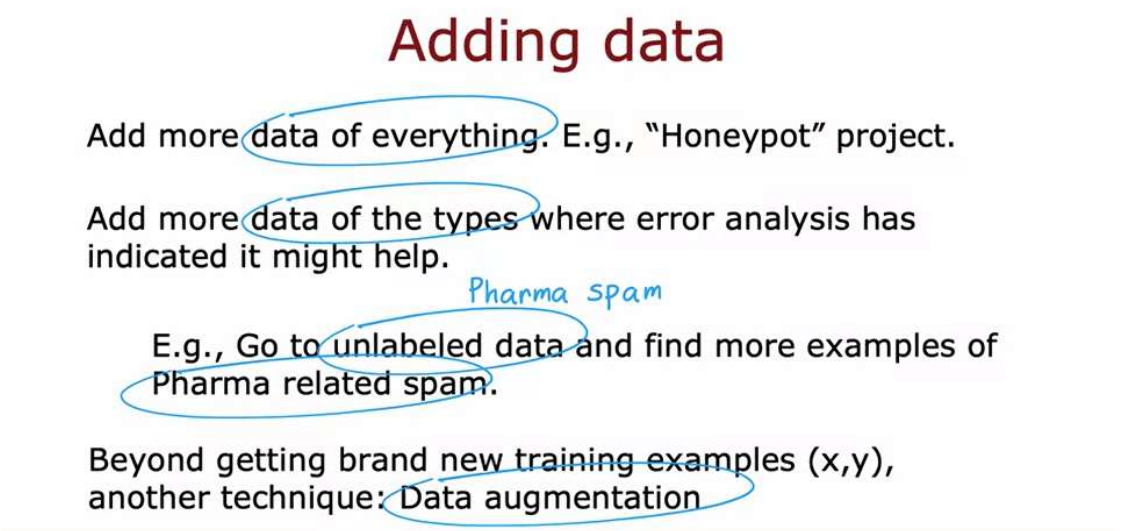

Wenn Sie die iterative Schleife von Maschinerie und Entwicklung durchlaufen, haben Sie möglicherweise viele Ideen, wie Sie das Modell oder die Daten ändern können, und es werden verschiedene Diagnosen erstellt, die Ihnen viele Hinweise zu den Auswahlmöglichkeiten für das Modell oder die Daten geben können. oder andere Teile der Architektur könnten am vielversprechendsten zum Ausprobieren sein. In den letzten Videos haben wir bereits über Voreingenommenheit und Varianz gesprochen.




Im nächsten Video möchte ich damit beginnen, Ihnen den Fehleranalyseprozess zu beschreiben, der eine zweite Reihe wichtiger Ideen enthält, um Erkenntnisse darüber zu gewinnen, welche Architekturentscheidungen fruchtbar sein könnten. Das ist die iterative Schleife der maschinellen Lernentwicklung und am Beispiel der Erstellung eines Spam-Klassifikators. Werfen wir einen Blick darauf, wie eine Fehleranalyse aussieht. Machen wir das im nächsten Video.

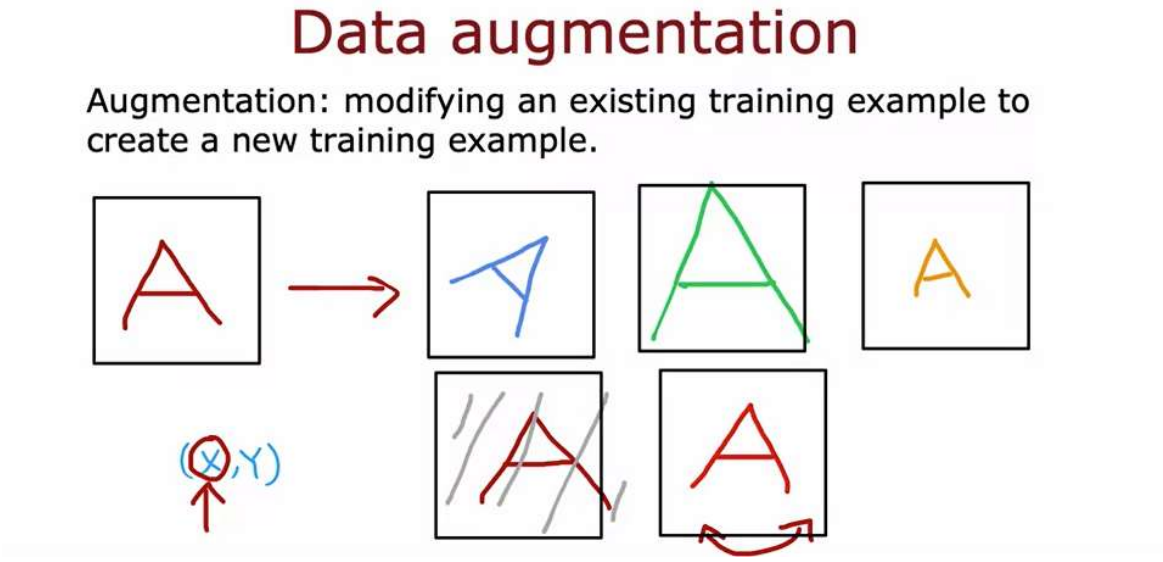
## Daten hinzufügen

In diesem Video möchte ich Ihnen einige Tipps zum Hinzufügen von Daten oder zum Sammeln weiterer Daten oder manchmal sogar zum Erstellen weiterer Daten für Ihre maschinelle Lernanwendung geben. Nur eine Warnung, dass dies in den nächsten Videos ein wenig wie eine Wundertüte verschiedener Techniken wirken wird. Und ich entschuldige mich, wenn es etwas überladen wirkt, denn Anwendungen für maschinelles Lernen sind anders.



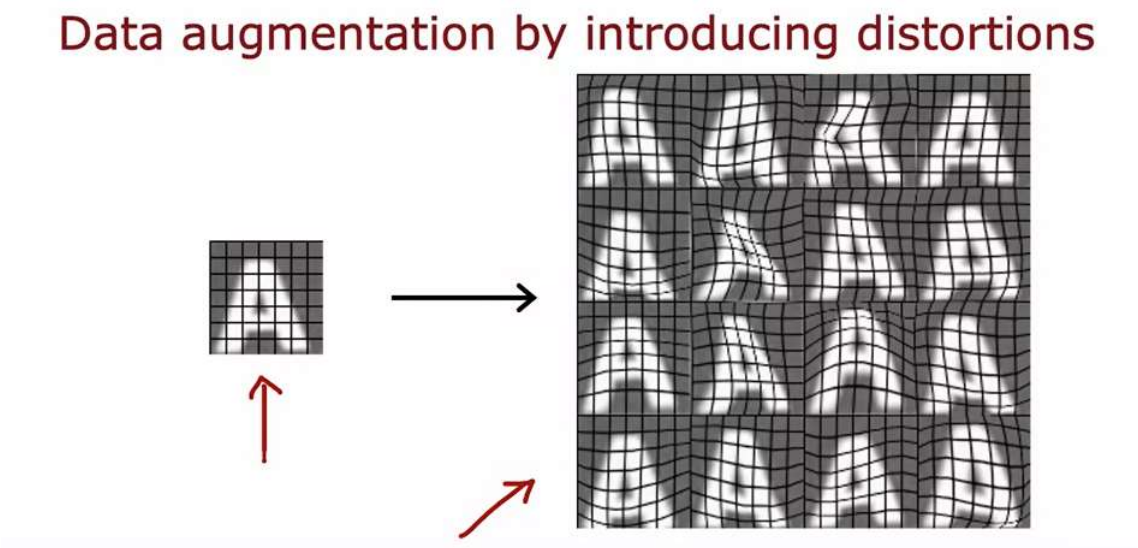

Maschinelles Lernen wird auf so viele verschiedene Probleme angewendet und manche Menschen sind hervorragend darin, Etiketten zu erstellen. Und für einige können Sie mehr Daten erhalten, für andere nicht. Und deshalb erfordern unterschiedliche Anwendungen tatsächlich manchmal leicht unterschiedliche Techniken. Ich hoffe jedoch, dass ich Ihnen in den nächsten Videos einige der Techniken vorstellen kann, die sich für verschiedene Anwendungen als am nützlichsten erwiesen

haben, auch wenn nicht jede davon für jede einzelne Anwendung anwendbar ist. Aber ich hoffe, dass viele davon auch für viele der Anwendungen nützlich sein werden, an denen Sie arbeiten werden. Werfen wir jedoch einen Blick auf einige Tipps zum Hinzufügen von Daten für Ihre Anwendung. Wenn wir Algorithmen für maschinelles Lernen trainieren, kommt es uns so vor, als würden wir uns immer wünschen, wir hätten fast immer noch mehr Daten. Und deshalb ist es manchmal verlockend, einfach mehr Daten von allem zu sammeln. Der Versuch, mehr Daten aller Art zu erhalten, kann jedoch langsam und teuer sein. Stattdessen könnte eine alternative Möglichkeit zum Hinzufügen von Daten darin bestehen, sich auf das Hinzufügen weiterer Daten der Typen zu konzentrieren, bei denen die Analyse gezeigt hat, dass dies hilfreich sein könnte. In der vorherigen Folie haben wir gesehen, dass, wenn die Fehleranalyse ergab, dass Pharma-Spam ein großes Problem darstellt, Sie sich möglicherweise für eine gezieltere Anstrengung entscheiden, nicht um mehr Daten zu erhalten, sondern sich darauf zu konzentrieren, mehr Beispiele für Pharma-Spam zu ermitteln Bei geringeren Kosten könnten Sie damit genau die E-Mails hinzufügen, die Sie benötigen, um zu lernen und Pharma-Spam besser zu erkennen. Ein Beispiel dafür, wie Sie dies tun könnten, wäre also, wenn Sie viele unbeschriftete E-Mail-Daten haben, beispielsweise E-Mails, die herumliegen und sich noch niemand die Mühe gemacht hat, sie als Spam oder Nicht-Spam zu kennzeichnen. Möglicherweise können Sie Ihre Mitarbeiter bitten, sie schnell zu überfliegen Durchsuchen Sie die unbeschrifteten Daten und finden Sie weitere Beispiele, insbesondere Spam im Pharmabereich. Und dies könnte die Leistung Ihres Lernalgorithmus weitaus steigern, als wenn Sie nur versuchen, mehr Daten aller Arten von E-Mails hinzuzufügen. Ich hoffe jedoch, dass Sie das allgemeinere Muster daraus mitnehmen können: Wenn Sie Möglichkeiten haben, mehr Daten von allem hinzuzufügen, ist das in Ordnung. Daran ist nichts auszusetzen. Aber wenn die Fehleranalyse ergeben hat, dass es bestimmte Teilmengen der Daten gibt, bei denen das Album besonders schlecht abschneidet. Und Sie möchten die Leistung verbessern und dann mehr Daten genau der Typen erhalten, bei denen Sie eine Verbesserung erzielen wollten. Seien es weitere Beispiele für Pharma-Spam oder weitere Beispiele für Phishing-Spam oder etwas anderes. Das könnte eine effizientere Möglichkeit sein, nur ein paar Daten hinzuzufügen, aber die Leistung Ihrer Algorithmen um einiges zu steigern. Abgesehen davon, dass Sie brandneue Trainingsbeispiele in die Hände bekommen xy. Es gibt noch eine weitere Technik, die sie insbesondere für Bild- und Audiodaten verwenden und die die Größe Ihres Trainingssatzes erheblich erhöhen kann.



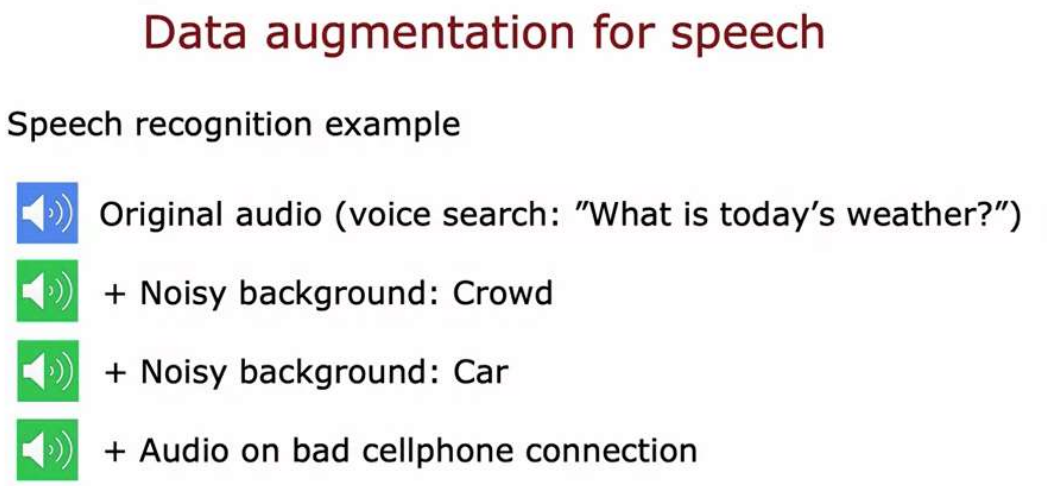

Diese Technik wird als Datenerweiterung bezeichnet. Und wir werden ein vorhandenes Zugbeispiel verwenden, um ein neues Trainingsbeispiel zu erstellen. Wenn Sie zum Beispiel versuchen, die Buchstaben von A bis Z aufgrund eines [UNVERSTÄNDLICHEN] Problems bei der optischen

Zeichenerkennung zu erkennen. Also nicht nur die Ziffern 0-9, sondern auch die Buchstaben von A bis Z. Wenn Sie ein Bild wie dieses haben, möchten Sie vielleicht ein neues Trainingsbeispiel erstellen, indem Sie das Bild ein wenig drehen. Oder indem Sie das Bild ein wenig vergrößern oder ein wenig verkleinern oder den Kontrast des Bildes ändern. Und das sind Beispiele für Verzerrungen des Bildes, die nichts an der Tatsache ändern, dass es sich immer noch um den Buchstaben A handelt. Und für einige Buchstaben, für andere jedoch nicht, kann man auch das Spiegelbild des Buchstabens nehmen und es sieht immer noch wie der Buchstabe A aus. Aber Dies gilt nur für einige Buchstaben, aber dies wären Möglichkeiten, ein Trainingsbeispiel X, Y zu nehmen und eine Verzerrung oder Transformation auf die Eingabe X anzuwenden, um ein anderes Beispiel mit derselben Bezeichnung zu erhalten.

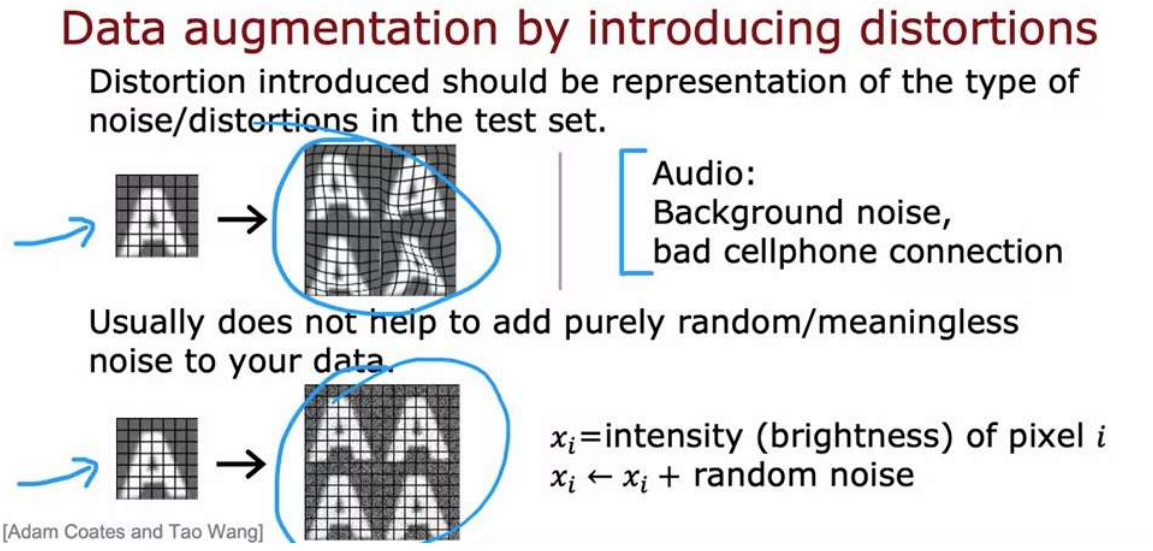



Und indem Sie dies tun, teilen Sie dem Algorithmus mit, dass sich der Buchstabe A etwas gedreht, etwas vergrößert oder ein wenig verkleinert hat, es immer noch der Buchstabe A ist. Wenn Sie zusätzliche Beispiele wie dieses erstellen, bleibt der Lernalgorithmus erhalten. Lernen Sie besser, wie das geht um den Buchstaben A zu erkennen. Für ein fortgeschritteneres Beispiel der Datenerweiterung. Sie können auch den Buchstaben A nehmen und ein Gitter darauf legen. Und indem Sie zufällige Verzerrungen dieses Gitters einführen, können Sie den Buchstaben A nehmen. Und Verzerrungen des Leders A einführen, um eine viel umfangreichere Bibliothek von Beispielen des Buchstabens A zu erstellen. Und dieser Prozess der Verzerrung dieser Beispiele hat dann ein Bild von verwandelt Hier finden Sie ein Beispiel für Trainingsbeispiele, die Sie dem Lernalgorithmus zuführen können, um zu hoffen, dass er robuster lernt. Was ist der Buchstabe A? Diese Idee der Datenerweiterung funktioniert auch für die Spracherkennung. Nehmen wir an, Sie haben für eine Sprachsuchanwendung einen Original-Audioclip, der so klingt. >> Wie ist das Wetter heute ? >> Eine Möglichkeit, die Datendokumentation auf Sprachdaten anzuwenden, besteht darin, verrauschtes Hintergrundaudio wie dieses aufzunehmen. So klingt zum Beispiel der Lärm einer Menschenmenge. Und es stellt sich heraus, dass, wenn man diese beiden Audioclips, den ersten und den Massenlärm, nimmt und sie zusammenfügt, man am Ende einen Audioclip erhält, der so klingt. >> Wie ist das Wetter heute ? >> Und Sie haben gerade einen Audioclip erstellt, der so klingt, als würde jemand sagen, wie das Wetter heute ist. Aber sie sagen es in der lauten Menge im Hintergrund. Oder wenn Sie ein anderes Hintergrundgeräusch aufnehmen würden, beispielsweise jemanden im Auto, dann klingen die Hintergrundgeräusche eines Autos so. Und Sie möchten den Original-Audioclip zum Autogeräusch hinzufügen, dann bekommen Sie das hier. >> Wie ist das Wetter heute ? >> Und es klingt wie der Original-Audioclip, aber als würde der Sprecher es aus einem Auto sagen. Und der fortgeschrittenere Schritt zur Datenerweiterung wäre, wenn Sie den Originalton so klingen lassen, als würden Sie ihn über eine schlechte Mobilfunkverbindung wie diese aufnehmen. Und so haben wir

hier gesehen, wie Sie einen Audioclip in drei Trainingsbeispiele umwandeln können, eines mit Hintergrundgeräuschen von Menschenmengen, eines mit Hintergrundgeräuschen von Autos und eines so, als ob es über eine schlechte Mobilfunkverbindung aufgenommen worden wäre.

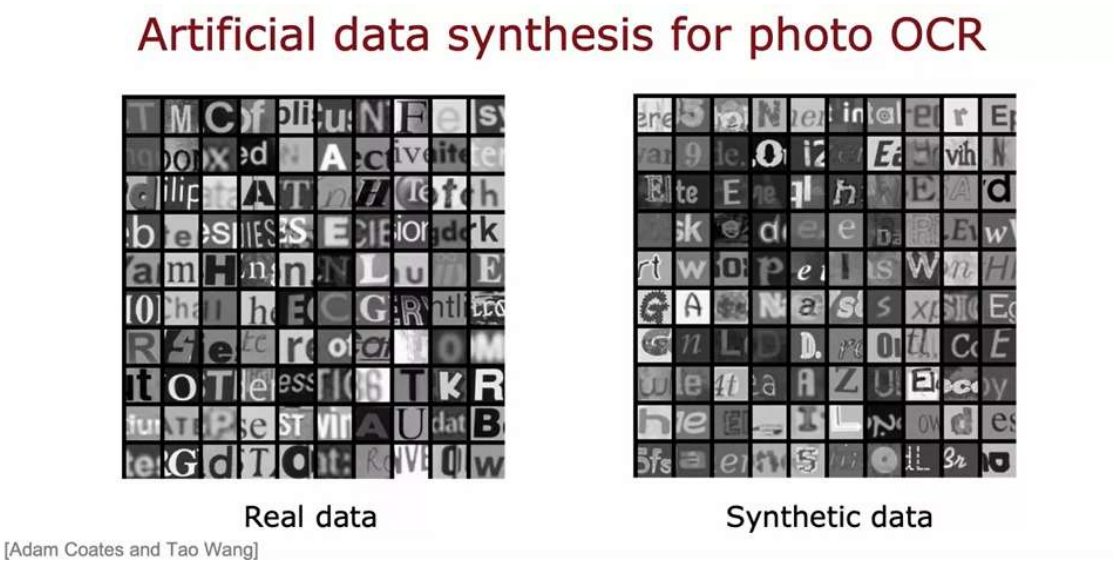



Und als ich an Spracherkennungssystemen gearbeitet habe, war dies tatsächlich eine wirklich entscheidende Technik, um die Größe der Trainingsdaten, die ich hatte, künstlich zu vergrößern, um einen genaueren Spracherkenner zu bauen. Ein Tipp für die Datenerweiterung ist, dass die Änderungen oder Verzerrungen, die Sie an den Daten vornehmen, repräsentativ für die Arten von Rauschen oder Verzerrungen im Testsatz sein sollten. Wenn Sie beispielsweise den Buchstaben „a" nehmen und ihn auf diese Weise verzerren, sieht dies immer noch wie Beispiele für Buchstaben aus, die Sie vielleicht da draußen sehen und die Sie gerne erkennen würden. Oder für Audio, das Hintergrundgeräusche oder eine schlechte Mobiltelefonverbindung hinzufügt, wenn dies repräsentativ für das ist, was Sie hier im Testsatz erwarten, dann ist dies eine hilfreiche Möglichkeit, eine Datenerweiterung an Ihren Audiodaten durchzuführen. Im Gegensatz dazu ist rein zufälliges, bedeutungsloses Rauschen der Daten meist nicht so hilfreich. Sie haben zum Beispiel den Buchstaben A genommen und ich habe das Rauschen pro Pixel hinzugefügt.



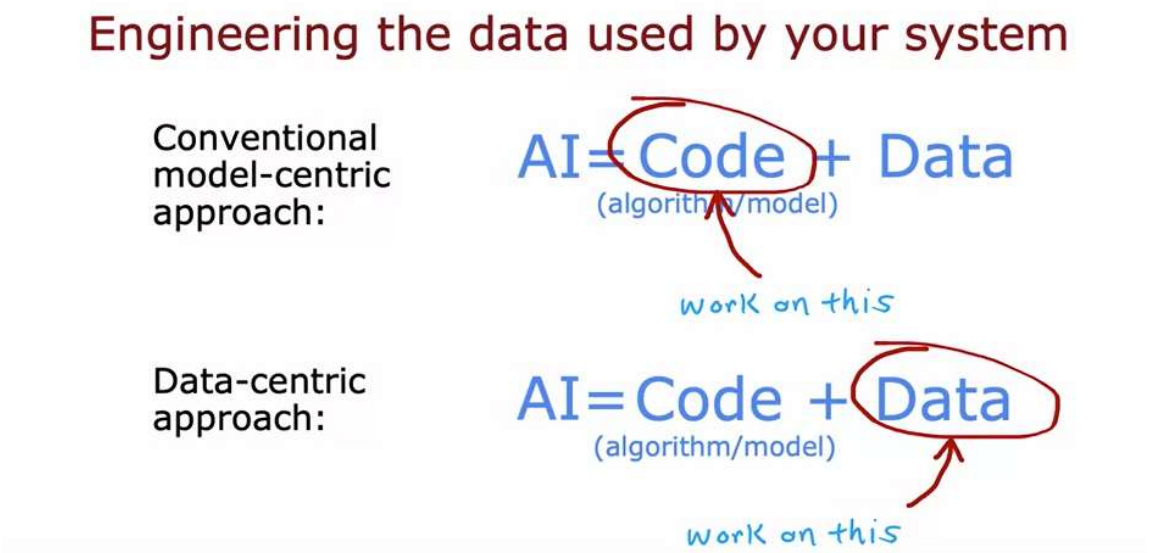

Wenn Xi die Intensität oder Helligkeit von Pixel i ist und ich nur jedem Pixel Rauschen hinzufügen würde, erhalten Sie am Ende Bilder, die so aussehen Das. Aber wenn dies nicht so repräsentativ für das ist, was Sie im Testsatz sehen, weil Sie im Testsatz nicht oft solche Bilder erhalten, ist das

tatsächlich weniger hilfreich. Eine Möglichkeit, über Datenerweiterung nachzudenken, ist also, wie Sie Ihre Daten modifizieren, verzerren oder verzerren oder mehr Rauschen in ihnen erzeugen können. Aber in gewisser Weise ist das, was Sie erhalten, immer noch ziemlich ähnlich zu dem, was Sie in Ihrem Testsatz haben, denn darin wird der Lernalgorithmus letztendlich gut abschneiden. Im Gegensatz dazu nimmt die Datenerweiterung ein vorhandenes Trainingsbeispiel und modifiziert es, um ein weiteres Trainingsbeispiel zu erstellen. Eine dieser Techniken ist die Datensynthese, bei der Sie völlig neue Beispiele von Grund auf erstellen. Nicht durch die Modifizierung eines vorhandenen Beispiels, sondern durch die Erstellung völlig neuer Beispiele. Nehmen Sie das Beispiel der Foto- OCR. Unter Foto-OCR oder fotooptischer Zeichenerkennung versteht man das Problem, ein Bild wie dieses zu betrachten und den Text, der in diesem Bild erscheint, automatisch von einem Computer lesen zu lassen. Es gibt also viel Text in diesem Bild. Wie kann man den OCR-Algorithmus trainieren, um Text aus einem Bild wie diesem zu lesen? Nun, wenn man sich die Buchstaben in diesem Bild genau ansieht, sehen sie tatsächlich so aus. Das sind also echte Daten aus einer Foto-OCR-Aufgabe. Und ein wichtiger Schritt bei der Foto-OCR-Aufgabe besteht darin, das kleine Bild wie dieses betrachten zu können und den Buchstaben in der Mitte zu erkennen.




Das hat also ein T in der Mitte, das hat den Buchstaben L in der Mitte, das hat den Buchstaben C in der Mitte und so weiter. Eine Möglichkeit, künstliche Daten für diese Aufgabe zu erstellen, besteht darin, den Texteditor Ihres Computers aufzurufen und festzustellen, dass er viele verschiedene Schriftarten enthält. Sie können diese Schriftarten übernehmen und im Grunde genommen zufälligen Text in Ihren Texteditor eingeben. Und Screenshots mit unterschiedlichen Farben, unterschiedlichen Kontrasten und sehr unterschiedlichen Schriftarten, und auf der rechten Seite erhalten Sie solche synthetischen Daten. Die Bilder auf der linken Seite waren reale Daten von realen Bildern, die in der Welt aufgenommen wurden. Die Bilder rechts sind synthetisch mit Mitteln am Computer erstellt und sehen tatsächlich ziemlich realistisch aus. Mit einer solchen synthetischen Datengenerierung können Sie also eine sehr große Anzahl von Bildern oder Beispielen für Ihre Foto-OCR-Aufgabe generieren. Es kann eine Menge Arbeit sein, den Code zu schreiben, um realistisch aussehende synthetische Daten für eine bestimmte Anwendung zu generieren. Aber wenn Sie sich die Zeit dafür nehmen, kann es Ihnen manchmal dabei helfen, eine sehr große Datenmenge für Ihre Anwendung zu generieren und die Leistung Ihres Albums enorm zu steigern. Die Generierung synthetischer Daten wurde höchstwahrscheinlich für die Computer-Vision-Entwicklung und weniger für andere Anwendungen verwendet. Auch für Audiogespräche nicht so viel. Alle Techniken, die Sie in diesem Video gesehen haben, beziehen sich auf die Suche nach Möglichkeiten, die von Ihrem System verwendeten Daten zu bearbeiten. In der Art und Weise, wie sich maschinelles Lernen in den letzten Jahrzehnten, vielen

Jahrzehnten, entwickelt hat. Die meisten Forscher im Bereich maschinelles Lernen konzentrierten sich auf den herkömmlichen modellzentrierten Ansatz und hier ist, was ich meine. Ein maschinelles Lernsystem oder ein KI-System umfasst sowohl Code zur Implementierung Ihres Albums oder Ihres Modells als auch die Daten, mit denen Sie das Algorithmusmodell trainieren. Und in den letzten Jahrzehnten haben die meisten Forscher, die maschinelles Lernen erforschten, den Datensatz heruntergeladen und die Daten gespeichert, während sie sich auf die Verbesserung des Codes, des Algorithmus oder des Modells konzentrierten.

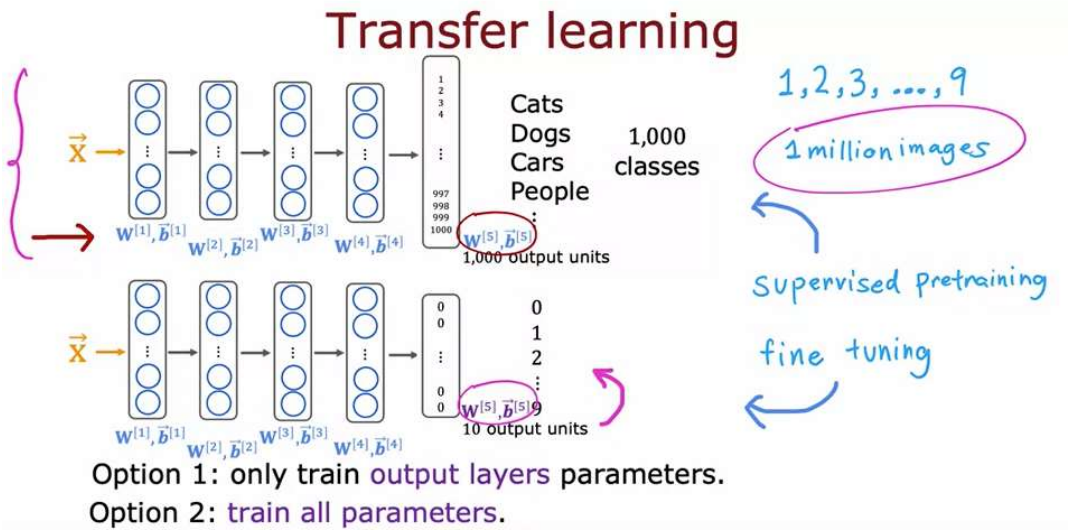



Dank dieses Paradigmas der maschinellen Lernforschung. Ich finde, dass wir die Algorithmen, auf die wir heute Zugriff haben, wie lineare Regression, logistische Regression, neuronale Netze und auch Entscheidungsbäume, nächste Woche sehen sollten. Es gibt Algorithmen, die bereits sehr gut sind und für viele Anwendungen gut funktionieren werden. Daher kann es manchmal sinnvoller sein, mehr Zeit mit einem datenzentrierten Ansatz zu verbringen, bei dem Sie sich auf die Entwicklung der von Ihrem Algorithmus verwendeten Daten konzentrieren. Dabei kann es sich um alles Mögliche handeln, von der Sammlung weiterer Daten nur über Arzneimittel-Spam. Wenn es das ist, was Ihnen die Fehleranalyse befohlen hat. Um die Datenerweiterung zu nutzen, um mehr Bilder oder mehr Audio zu erzeugen, oder um die Datensynthese zu nutzen, um einfach mehr Trainingsbeispiele zu erstellen. Und manchmal kann die Fokussierung auf die Daten eine effiziente Möglichkeit sein, die Leistung Ihres Lernalgorithmus zu verbessern. Daher hoffe ich, dass Ihnen dieses Video eine Reihe von Tools an die Hand gibt, mit denen Sie effizient und effektiv mehr Daten hinzufügen können, damit Ihr Lernalgorithmus besser funktioniert. Nun gibt es auch einige Anwendungen, bei denen man einfach nicht so viele Daten hat und es wirklich schwierig ist, mehr Daten zu bekommen. Es stellt sich heraus, dass es eine Technik namens Transferlernen gibt, die in diesem Umfeld eingesetzt werden könnte, um die Leistung Ihres Lernalgorithmus enorm zu steigern. Und die Schlüsselidee besteht darin, Daten aus völlig anderen, kaum zusammenhängenden Aufgaben zu übernehmen. Aber mit einem neuronalen Netzwerk gibt es manchmal Möglichkeiten, diese Daten aus ganz anderen Aufgaben zu nutzen, um Ihren Algorithmus dazu zu bringen, Ihre Anwendung besser zu bearbeiten. Gilt nicht für alles, aber wenn doch, kann es sehr wirkungsvoll sein. Werfen wir im nächsten Video einen Blick darauf, wie Transferlernen funktioniert.

## Lernen übertragen: Daten aus einer anderen Aufgabe verwenden

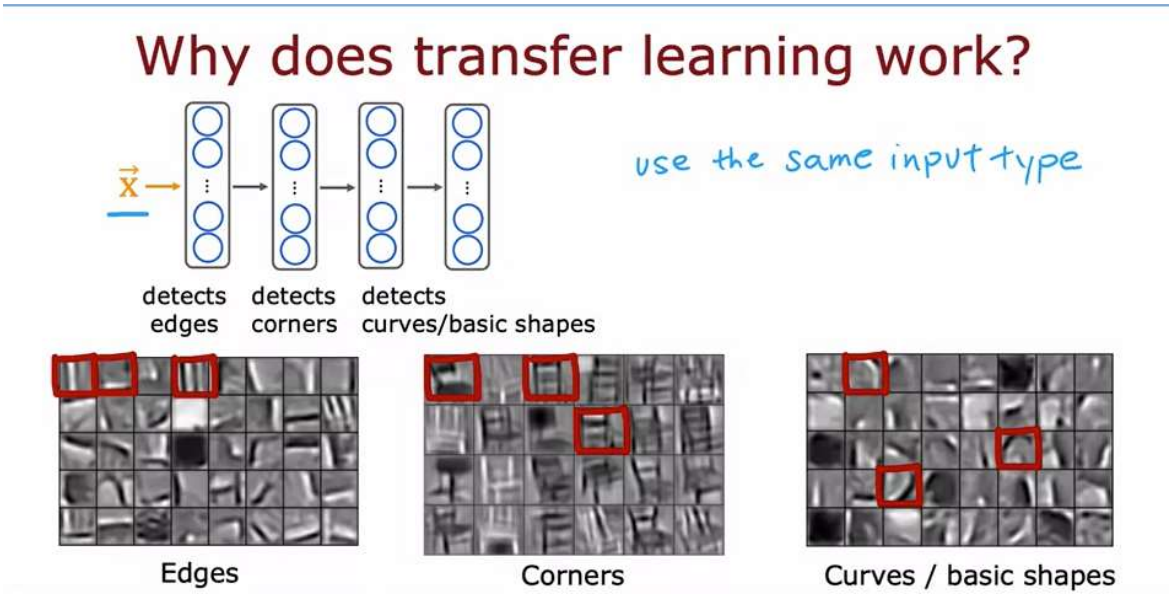
Für eine Anwendung, bei der Sie nicht so viele Daten haben, ist Transferlernen eine wunderbare Technik, mit der Sie Daten aus einer anderen Aufgabe zur Unterstützung Ihrer Anwendung verwenden können. Dies ist eine dieser Techniken, die ich sehr häufig verwende. Werfen wir einen

Blick darauf, wie Transferlernen funktioniert. So funktioniert Transferlernen. Nehmen wir an, Sie möchten die handgeschriebenen Ziffern von Null bis Neun erkennen, verfügen aber nicht über so viele beschriftete Daten dieser handschriftlichen Ziffern. Hier erfahren Sie, was Sie tun können. Angenommen, Sie finden einen sehr großen Datensatz mit einer Million Bildern von Katzen, Hunden, Autos, Menschen usw., tausend Klassen. Anschließend können Sie damit beginnen, ein neuronales Netzwerk anhand dieses großen Datensatzes aus einer Million Bildern mit tausend verschiedenen Klassen zu trainieren und den Algorithmus so zu trainieren, dass er ein Bild X als Eingabe verwendet, und lernen, jede dieser 1.000 verschiedenen Klassen zu erkennen. In diesem Prozess lernen Sie am Ende die Parameter für die erste Schicht des neuronalen Netzwerks W^1, b^1, für die zweite Schicht W^2, b^2 usw., W^3, b^3, W ^4, b^4 und W^5, b^5 für die Ausgabeebene. Um Transferlernen anzuwenden, erstellen Sie dann eine Kopie dieses neuronalen Netzwerks, in das Sie die Parameter W^1, b^1, W^2, b^2, W^3, b^3 und W^ einfügen 4, b^4. Aber für die letzte Schicht würden Sie die Ausgabeschicht eliminieren und sie durch eine viel kleinere Ausgabeschicht mit nur 10 statt 1.000 Ausgabeeinheiten ersetzen.



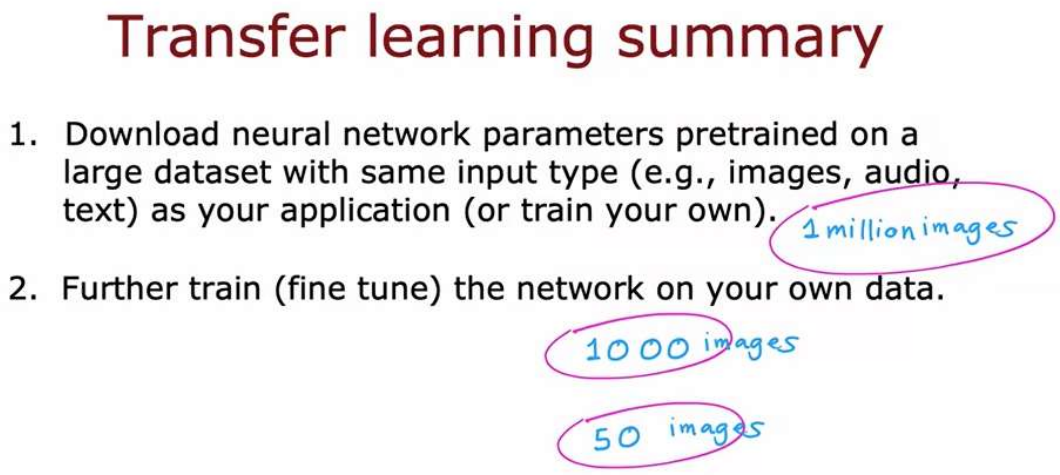

Diese 10 Ausgabeeinheiten entsprechen den Klassen Null, Eins und Neun, die Ihr neuronales Netzwerk erkennen soll. Beachten Sie, dass die Parameter W^5, b^5 nicht kopiert werden können, da sich die Dimension dieser Ebene geändert hat. Sie müssen sich also neue Parameter W^5, b^5 ausdenken, die Sie von Grund auf trainieren müssen anstatt es einfach aus dem vorherigen neuronalen Netzwerk zu kopieren. Beim Transferlernen können Sie die Parameter der ersten vier Schichten, also aller Schichten außer der letzten Ausgabeschicht, als Ausgangspunkt für die Parameter verwenden und dann einen Optimierungsalgorithmus wie den Gradientenabstieg oder den Adam-Optimierungsalgorithmus ausführen Die Parameter werden mit den Werten aus diesem neuronalen Netzwerk oben initialisiert. Im Detail gibt es zwei Möglichkeiten, wie man diese Parameter neuronaler Netze trainieren kann. Option 1 besteht darin, dass Sie nur die Parameter der Ausgabeebenen trainieren. Sie würden die Parameter W^1, b^1, W^2, b^2 bis W^4, b^4 als Werte von oben nehmen und sie einfach festhalten und sich nicht einmal die Mühe machen, sie zu ändern, und Verwenden Sie einen Algorithmus wie den stochastischen Gradientenabstieg oder den Adam-Optimierungsalgorithmus, um nur W^5, b^5 zu aktualisieren, um die übliche Kostenfunktion zu senken, die Sie zum Erlernen der Erkennung dieser Ziffern Null bis Neun aus einem kleinen Trainingssatz dieser Ziffern Null bis verwenden Neun, das ist also Option 1. Option 2 wäre, alle Parameter im Netzwerk zu trainieren, einschließlich W^1, b^1, W^2, b^2 bis hin zu W^5, b^5, aber dem ersten Die Parameter der vier Ebenen würden mit den Werten initialisiert, die Sie oben trainiert haben. Wenn Sie einen sehr kleinen Trainingssatz haben, funktioniert Option 1 möglicherweise etwas besser. Wenn Sie jedoch einen etwas größeren Trainingssatz haben, funktioniert Option 2

möglicherweise etwas besser. Dieser Algorithmus wird Transferlernen genannt, weil die Intuition darin besteht, zu lernen, Katzen, Hunde, Kühe, Menschen usw. zu erkennen. Es wird hoffentlich einige plausible Parametersätze für die früheren Ebenen zur Verarbeitung von Bildeingaben gelernt haben. Durch die Übertragung dieser Parameter auf das neue neuronale Netzwerk beginnt das neue neuronale Netzwerk mit den Parametern an einer viel besseren Stelle, sodass wir nur noch ein wenig weiter lernen müssen. Hoffentlich entsteht am Ende ein ziemlich gutes Modell. Diese beiden Schritte des ersten Trainings an einem großen Datensatz und der anschließenden weiteren Optimierung der Parameter an einem kleineren Datensatz werden als überwachtes Vortraining für diesen Schritt oben bezeichnet. Dabei trainieren Sie das neuronale Netzwerk anhand eines sehr großen Datensatzes von beispielsweise einer Million Bildern, die nicht ganz der entsprechenden Aufgabe entsprechen. Dann wird der zweite Schritt als Feinabstimmung bezeichnet. Dabei nehmen Sie die Parameter, die Sie initialisiert oder aus einem überwachten Vortraining erhalten haben, und führen dann einen weiteren Gradientenabstieg durch, um die Gewichte so abzustimmen, dass sie zu Ihrer spezifischen Anwendung der handschriftlichen Ziffernerkennung passen. Wenn Sie über einen kleinen Datensatz verfügen, sogar über Zehntausende oder Hunderte oder Tausende oder nur Zehntausende Bilder der handgeschriebenen Ziffern, kann die Möglichkeit, aus diesen Millionen Bildern einer nicht ganz verwandten Aufgabe zu lernen, die Leistung Ihres Lernalgorithmus tatsächlich erheblich verbessern.




Eine schöne Sache beim Transferlernen ist auch, dass Sie möglicherweise nicht derjenige sein müssen, der eine beaufsichtigte Vorschulung durchführt. Für viele neuronale Netze wird es bereits Forscher geben, die bereits ein neuronales Netz auf einem großen Bild trainiert haben und ein trainiertes neuronales Netz im Internet veröffentlicht haben, das jeder kostenlos herunterladen und nutzen kann. Das bedeutet, dass Sie, anstatt den ersten Schritt selbst durchzuführen, einfach das neuronale Netzwerk herunterladen können, das jemand anderes möglicherweise wochenlang trainiert hat, und dann die Ausgabeschicht durch Ihre eigene Ausgabeschicht ersetzen und entweder Option 1 oder Option 2 ausführen können Optimieren Sie ein neuronales Netzwerk, an dem jemand anderes bereits ein überwachtes Vortraining durchgeführt hat, und nehmen Sie einfach ein wenig Feinabstimmung vor, um schnell ein neuronales Netzwerk zu erhalten, das Ihre Aufgabe gut erfüllt. Das Herunterladen eines vorab trainierten Modells, das jemand anderes trainiert und kostenlos zur Verfügung gestellt hat, ist eine dieser Techniken, mit denen wir alle viel bessere Ergebnisse erzielen können, wenn wir auf der Arbeit des anderen in der Community für maschinelles Lernen aufbauen. Dank der Großzügigkeit anderer Forscher, die ihre neuronalen Netze vorab trainiert und online gestellt haben. Aber warum funktioniert Transferlernen überhaupt? Wie können Sie Parameter, die

Sie durch die Erkennung von Katzen, Hunden, Autos und Menschen erhalten, nutzen und diese nutzen, um etwas so Unterschiedliches wie handgeschriebene Ziffern zu erkennen? Hier steckt eine gewisse Intuition dahinter. Wenn Sie ein neuronales Netzwerk trainieren, um beispielsweise verschiedene Objekte in Bildern zu erkennen, lernt die erste Schicht eines neuronalen Netzwerks möglicherweise, Kanten im Bild zu erkennen. Wir stellen uns dies als etwas untergeordnete Merkmale im Bild vor, die dazu dienen, Kanten zu erkennen. Jedes dieser Quadrate ist eine Visualisierung dessen, was ein einzelnes Neuron zu erkennen gelernt hat, indem es lernt, Pixel zu gruppieren, um Kanten in einem Bild zu finden. Die nächste Schicht des neuronalen Netzwerks lernt dann, Kanten zu gruppieren, um Ecken zu erkennen. Jedes davon ist eine Visualisierung dessen, was ein Neuron möglicherweise gelernt hat, technische, einfache Formen wie eckige Formen wie diese zu erkennen. Die nächste Schicht des neuronalen Netzwerks hat möglicherweise gelernt, einige komplexere Formen zu erkennen. Sie speichern generische Formen wie Grundkurven oder kleinere Formen wie diese. Deshalb bringen Sie dem neuronalen Netzwerk bei, Kanten, Ecken und Grundformen zu erkennen, indem Sie lernen, viele verschiedene Bilder zu erkennen. Indem Sie ein neuronales Netzwerk darauf trainieren, so unterschiedliche Dinge wie Katzen, Hunde, Autos und Menschen zu erkennen, helfen Sie ihm daher dabei, diese ziemlich allgemeinen Merkmale von Bildern zu erkennen und Kanten, Ecken, Kurven und Grundformen zu finden.

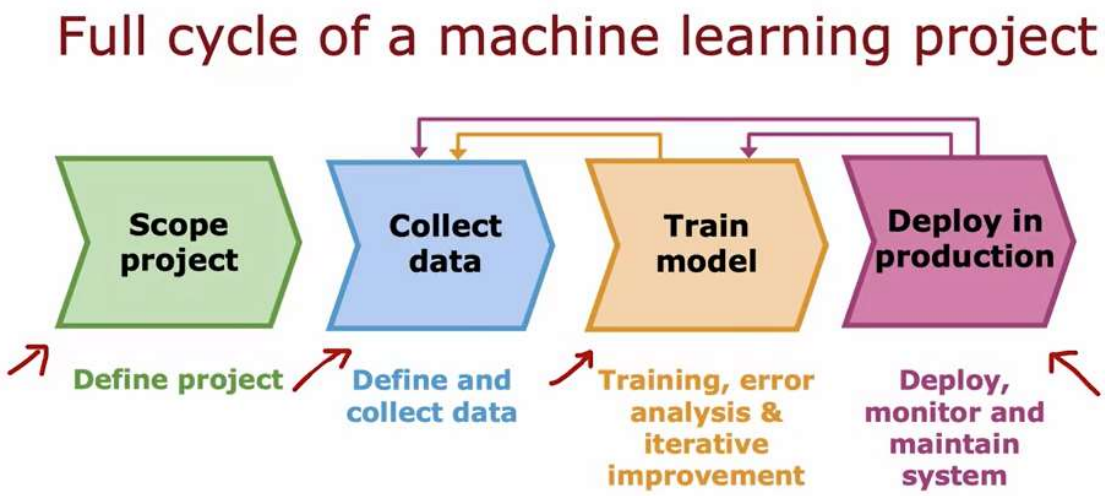



Dies ist für viele andere Computer-Vision-Aufgaben nützlich, beispielsweise für das Erkennen handgeschriebener Ziffern. Eine Einschränkung des Vortrainings besteht jedoch darin, dass der Bildtyp x für die Schritte Vortraining und Feinabstimmung derselbe sein muss. Wenn die letzte Aufgabe, die Sie lösen möchten, eine Computer-Vision-Aufgabe ist, dann wurde im Vortrainingsschritt auch ein neuronales Netzwerk auf die gleiche Art von Eingabe trainiert, nämlich ein Bild mit den gewünschten Abmessungen. Wenn Ihr Ziel hingegen darin besteht, ein Spracherkennungssystem zur Verarbeitung von Audio zu entwickeln, wird ein auf Bildern vorab trainiertes neuronales Netzwerk bei Audio wahrscheinlich nicht viel nützen. Stattdessen benötigen Sie ein vorab auf Audiodaten trainiertes neuronales Netzwerk, in dem Sie dann Ihren eigenen Audiodatensatz verfeinern und das Gleiche auch für andere Arten von Anwendungen tun. Sie können ein neuronales Netzwerk vorab anhand von Textdaten trainieren. Wenn Ihre Anwendung über eine Speicherfunktion für die Eingabe von x Textdaten verfügt, können Sie dieses neuronale Netzwerk anhand Ihrer eigenen Daten feinabstimmen. Zusammenfassend sind dies die beiden Schritte des Transferlernens. Schritt 1 besteht darin, ein neuronales Netzwerk mit Parametern herunterzuladen, die anhand eines großen Datensatzes mit demselben Eingabetyp wie Ihre Anwendung vorab trainiert wurden. Bei diesem Eingabetyp kann es sich um Bilder, Audio, Texte oder etwas anderes handeln.

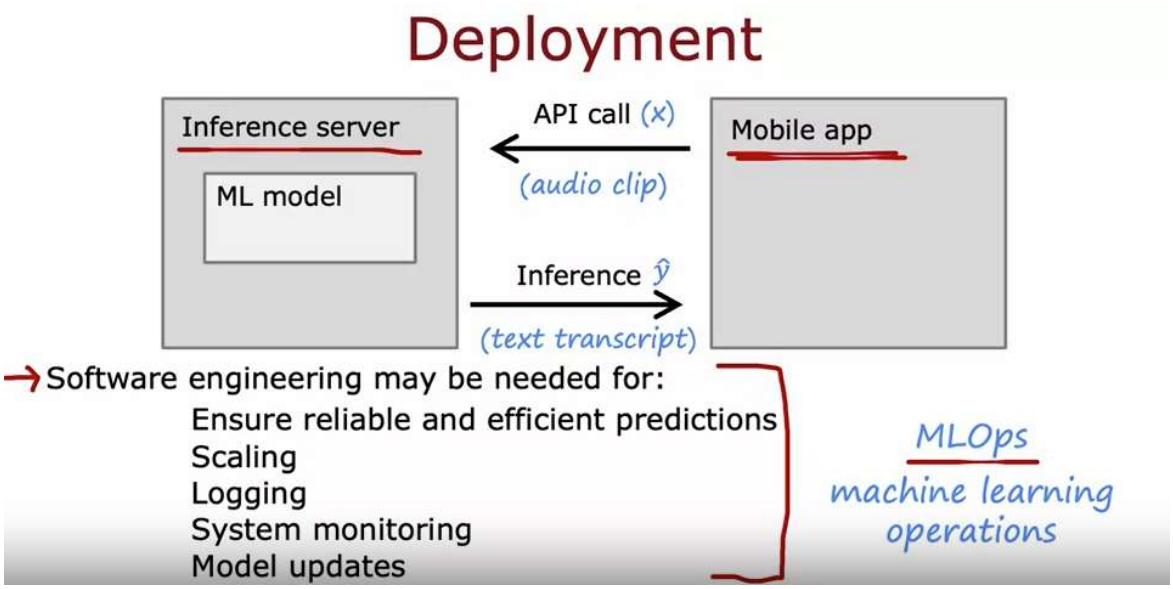
Wenn Sie das neuronale Netzwerk nicht herunterladen möchten, können Sie vielleicht Ihr eigenes Netzwerk trainieren. Aber in der Praxis, wenn Sie beispielsweise Bilder verwenden, ist es weitaus üblicher, das vorab trainierte neuronale Netzwerk einer anderen Person herunterzuladen. Anschließend können Sie das Netzwerk anhand Ihrer eigenen Daten weiter trainieren oder optimieren. Ich habe herausgefunden, dass, wenn man ein neuronales Netzwerk vorab auf einen großen Datensatz, beispielsweise eine Million Bilder, trainieren kann, man manchmal einen viel kleineren Datensatz, vielleicht tausend Bilder, vielleicht sogar noch kleiner, verwenden kann, um das neuronale Netzwerk selbst zu verfeinern Daten und erhalten ziemlich gute Ergebnisse. Manchmal trainierte ich neuronale Netze anhand von nur 50 Bildern, die mit dieser Technik recht gut funktionierten, obwohl sie bereits anhand eines viel größeren Datensatzes vorab trainiert worden waren. Diese Technik ist kein Allheilmittel. Sie können nicht jede Anwendung mit nur 50 Bildern zum Laufen bringen, aber sie hilft sehr, wenn der Datensatz, den Sie für Ihre Anwendung haben, nicht so groß ist. Übrigens, wenn Sie in den Nachrichten von fortgeschrittenen Techniken wie GPT-3 oder BERTs oder auf ImageNet vortrainierten neuronalen Netzen gehört haben, handelt es sich tatsächlich um Beispiele für neuronale Netze, die von jemand anderem auf einem sehr großen Bild vortrainiert wurden B. Datensätze oder Textdatensätze, können sie dann in anderen Anwendungen feinabgestimmt werden. Wenn Sie noch nie von GPT-3, BERTs oder ImageNet gehört haben, machen Sie sich darüber keine Sorgen. Dies sind erfolgreiche Anwendungen des Vortrainings in der Literatur zum maschinellen Lernen. Eines der Dinge, die mir am Transferlernen gefallen, ist einfach die Möglichkeit, wie die Community für maschinelles Lernen Ideen, Code und sogar Parameter untereinander ausgetauscht hat, dank der Forscher, die große neuronale Netze vorab trainiert und gepostet haben Die Parameter stehen im Internet kostenlos zur Verfügung, damit jeder sie herunterladen und verwenden kann. Dies ermöglicht es jedem, vorab trainierte Modelle für die Feinabstimmung potenziell viel kleinerer Datensätze zu verwenden. Beim maschinellen Lernen bauen wir alle häufig auf der Arbeit des anderen auf, und der offene Austausch von Ideen, Codes und trainierten Parametern ist eine der Möglichkeiten, mit denen die Gemeinschaft des maschinellen Lernens und wir alle gemeinsam viel besser abschneiden können arbeiten, als es jeder einzelne Mensch alleine kann. Ich hoffe, dass Sie, wenn Sie der Community für maschinelles Lernen beitreten, eines Tages vielleicht auch einen Weg finden, einen Beitrag zu dieser Community zu leisten. Das war’s für das Vortraining. Ich hoffe, dass Sie diese Technik nützlich finden. Im nächsten Video möchte ich Ihnen einige Gedanken zum gesamten Zyklus eines maschinellen Lernprojekts mitteilen. Wenn Sie ein maschinelles Lernsystem aufbauen, sollten Sie alle Schritte berücksichtigen, über die Sie nachdenken sollten. Schauen wir uns das im nächsten Video an.

## Vollständiger Zyklus eines maschinellen Lernprojekts

Bisher haben wir viel darüber gesprochen, wie man ein Modell trainiert, und auch ein wenig darüber, wie man Daten für Ihre Anwendung für maschinelles Lernen erhält. Aber wenn ich ein maschinelles Lernsystem aufbaue, ist das Trainieren eines Modells für mich nur ein Teil des Puzzles. In diesem Video möchte ich mit Ihnen teilen, was meiner Meinung nach den gesamten Zyklus eines maschinellen Lernprojekts darstellt. Das heißt, welche Schritte sollten beim Aufbau eines wertvollen Systems für maschinelles Lernen beachtet und geplant werden? Werfen wir einen Blick darauf und lassen Sie mich den gesamten Zyklus eines maschinellen Lernprojekts am Beispiel der Spracherkennung veranschaulichen.




Der erste Schritt eines maschinellen Lernprojekts besteht darin, den Projektumfang festzulegen. Mit anderen Worten: Entscheiden Sie, um welches Projekt es sich handelt und woran Sie arbeiten möchten. Ich habe zum Beispiel einmal beschlossen, an der Spracherkennung für die Sprachsuche zu arbeiten. Das heißt, Sie führen eine Websuche durch, indem Sie mit Ihrem Mobiltelefon sprechen, anstatt etwas in Ihr Mobiltelefon einzugeben. Dieser Projektumfang. Nachdem Sie entschieden haben, woran Sie arbeiten möchten, müssen Sie Daten sammeln. Entscheiden Sie, welche Daten Sie zum Trainieren Ihres maschinellen Lernsystems benötigen, und erledigen Sie die Arbeit, um die Audiodaten und die Transkripte der Beschriftungen für Ihren Datensatz abzurufen. Das ist Datenerfassung. Nachdem Sie Ihre erste Datenerfassung durchgeführt haben, können Sie mit dem Training des Modells beginnen. Hier würden Sie ein Spracherkennungssystem und eine Karavellen-Fehleranalyse trainieren und Ihr Modell iterativ verbessern. Ist überhaupt nicht ungewöhnlich. Nachdem Sie mit dem Training des Modells für die Fehleranalyse oder für eine Bias-Varianz-Analyse begonnen haben, wird Ihnen mitgeteilt, dass Sie vielleicht noch einmal zurückgehen möchten, um weitere Daten zu sammeln. Sammeln Sie möglicherweise mehr Daten von allem oder einfach mehr Daten eines bestimmten Typs, wenn Ihre Fehleranalyse Ihnen sagt, dass Sie die Leistung Ihres Lernalgorithmus verbessern möchten. Als ich beispielsweise einmal an der Sprache arbeitete, stellte ich fest, dass es meinem Modell besonders schlecht ging, wenn im Hintergrund Autogeräusche zu hören waren. Das klang, als würde jemand in einem Auto sprechen. Die Leistung meines Sprachsystems war schlecht. Ich habe beschlossen, mehr Daten abzurufen. Tatsächlich nutzte ich die Datenerweiterung, um mehr Sprachdaten zu erhalten, die wie ein Auto klingen, um die Leistung meines Lernalgorithmus zu verbessern. Sie durchlaufen diese Schleife ein paar Mal, trainieren das Modell, führen eine Fehleranalyse durch, gehen zurück, um weitere Daten zu sammeln, und tun dies möglicherweise eine Weile, bis Sie schließlich sagen, dass das Modell gut genug ist, um es dann in einer Produktionsumgebung bereitzustellen.



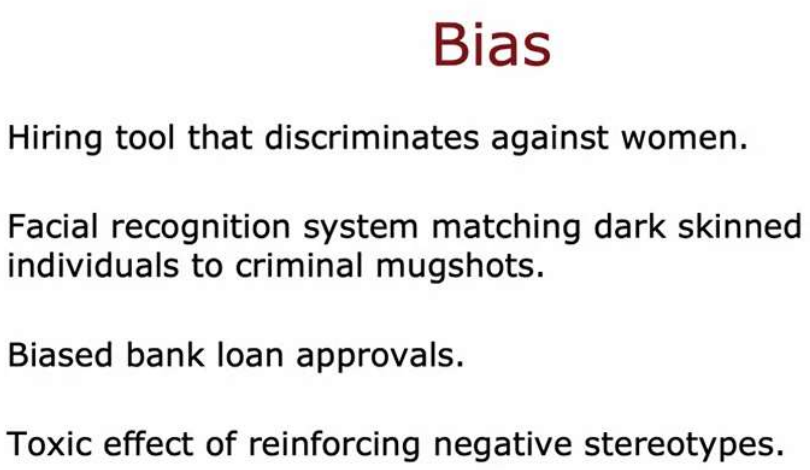

Das bedeutet, dass Sie es den Benutzern zur Nutzung zur Verfügung stellen. Wenn Sie ein System bereitstellen, müssen Sie auch sicherstellen, dass Sie die Leistung des Systems weiterhin überwachen und das System warten, falls sich die Leistung verschlechtert, um die Leistung wiederherzustellen, anstatt Ihr maschinelles Lernmodell nur auf einem Server zu hosten. Auf der nächsten Folie werde ich etwas mehr darüber sagen, warum Sie diese maschinellen Lernsysteme warten müssen. Aber nach dieser Bereitstellung stellen Sie manchmal fest, dass dies nicht so gut funktioniert, wie Sie es sich erhofft haben, und Sie kehren zurück, um das Modell zu trainieren, um es erneut zu verbessern, oder gehen sogar noch einmal zurück, um weitere Daten zu erhalten. Wenn Benutzer und Sie die Berechtigung haben, Daten aus Ihrer Produktionsbereitstellung zu verwenden, können Ihnen diese Daten aus Ihrem funktionierenden Sprachsystem manchmal sogar Zugriff auf noch mehr Daten verschaffen, mit denen Sie die Leistung Ihres Systems weiter verbessern können. Ich denke, Sie haben ein Gefühl dafür, was die Festlegung des Projektumfangs bedeutet, und wir haben in diesem Kurs viel über das Sammeln von Daten und das Trainieren von Modellen gesprochen. Aber lassen Sie mich etwas detaillierter mit Ihnen teilen, wie die Bereitstellung in der Produktion aussehen könnte. Nachdem Sie ein leistungsstarkes Modell für maschinelles Lernen trainiert haben, beispielsweise ein Spracherkennungsmodell, besteht eine übliche Methode zur Bereitstellung des Modells darin, Ihr Modell für maschinelles Lernen zu nehmen und es auf einem Server zu implementieren, den ich als Inferenzserver bezeichnen werde , dessen Aufgabe es ist, Ihr maschinelles Lernmodell, Ihr trainiertes Modell, aufzurufen , um Vorhersagen zu treffen. Wenn Ihr Team dann eine mobile App implementiert hat, beispielsweise eine soziale Anwendung, kann die mobile App, wenn ein Benutzer mit der mobilen App spricht, einen API-Aufruf durchführen, um den aufgezeichneten Audioclip und den des Inferenzservers an Ihren Inferenzserver weiterzuleiten Die Aufgabe besteht darin, ihm das Modell für maschinelles Lernen bereitzustellen und ihm dann die Vorhersage Ihres Modells zurückzugeben, in diesem Fall wären es die Texttranskripte dessen, was gesagt wurde. Dies wäre eine gängige Methode zum Implementieren einer Anwendung, die über die API und den Inferenzserver aufruft und Ihr Modell wiederholt Vorhersagen basierend auf der Eingabe x treffen lässt. Dabei handelte es sich um ein häufiges Muster, bei dem es auf die implementierte Anwendung ankam. Sie haben einen API-Aufruf, um Ihrem Lernalgorithmus D die Eingabe x und Ihrem maschinellen Lernmodell eine Ausgabe zur Vorhersage zu geben, beispielsweise y hat. Um dies zu implementieren, ist möglicherweise etwas Softwareentwicklung erforderlich, um den gesamten Code zu schreiben, der all diese Dinge erledigt. Je nachdem, ob Ihre Anwendung nur ein paar wenige Benutzer oder Millionen von Benutzern bedienen muss, kann der Umfang des erforderlichen Softwareentwicklers sehr unterschiedlich sein. Ich habe Software entwickelt, die nur einer Handvoll

Benutzern auf meinem Laptop dient, und ich habe auch Software entwickelt, die Hunderte Millionen Benutzer bedient, die erhebliche Rechenzentrumsressourcen benötigen. Abhängig von der benötigten Waagenanwendung ist möglicherweise Softwareentwicklung erforderlich, um sicherzustellen, dass Ihr Inferenzserver zuverlässige und effiziente Vorhersagen über den nicht zu hohen Rechenaufwand treffen kann. Um die Skalierung auf eine große Anzahl von Benutzern zu verwalten, ist möglicherweise Softwareentwicklung erforderlich. Sie möchten häufig die Daten protokollieren, die Sie erhalten, sowohl die Eingaben (x) als auch die Vorhersagen (y), vorausgesetzt, dass die Privatsphäre und Einwilligung des Benutzers die Speicherung dieser Daten zulässt. Diese Daten sind, wenn Sie darauf zugreifen können, auch für die Systemüberwachung sehr nützlich. Ich habe zum Beispiel einmal ein Spracherkennungssystem auf der Grundlage eines bestimmten Datensatzes erstellt, den ich hatte, aber als plötzlich neue Berühmtheiten bekannt wurden oder Wahlen dazu führten, dass neue Politiker gewählt wurden, suchten die Leute nach diesen neuen Namen, die nicht im Datensatz enthalten waren Trainingsset und dann lief mein System schlecht. Da wir das System überwachten, konnten wir herausfinden, wann sich die Daten veränderten und der Algorithmus ungenauer wurde. Dadurch konnten wir das Modell neu trainieren und anschließend ein Modellupdate durchführen, um das alte Modell durch ein neues zu ersetzen. Der Bereitstellungsprozess kann ein gewisses Maß an Software-Engineering erfordern. Für einige Anwendungen ist möglicherweise nicht so viel Softwareentwicklung erforderlich, wenn Sie sie nur auf einem Laptop oder auf einem oder zwei Servern ausführen. Abhängig von dem Team, in dem Sie arbeiten, ist es möglich, dass Sie das Modell für maschinelles Lernen erstellt haben, für die Bereitstellung ist jedoch möglicherweise ein anderes Team verantwortlich. Aber es gibt ein wachsendes Feld im maschinellen Lernen namens MLOps. Dies steht für Machine Learning Operations. Dies bezieht sich auf die Praxis, maschinelle Lernsysteme systematisch aufzubauen, bereitzustellen und zu warten. Um sicherzustellen, dass Ihr Modell für maschinelles Lernen zuverlässig ist, gut skaliert, über gute Gesetze verfügt und überwacht wird, müssen Sie all diese Dinge tun. Anschließend haben Sie die Möglichkeit, das Modell entsprechend zu aktualisieren, damit es weiterhin einwandfrei läuft. Wenn Sie Ihr System beispielsweise für Millionen von Menschen bereitstellen, möchten Sie möglicherweise sicherstellen, dass Sie über hochoptimierte Implementierungen verfügen, damit die Rechenkosten für die Versorgung von Millionen von Menschen nicht zu hoch werden. In diesem und den letzten Parsern habe ich viel Zeit damit verbracht, darüber zu sprechen, wie man ein Modell für maschinelles Lernen trainiert, und habe hier den absolut entscheidenden Teil gefunden, um sicherzustellen, dass Sie über ein Hochleistungssystem verfügen. Wenn Sie das System jemals für Millionen von Menschen bereitstellen müssen, sind dies wahrscheinlich einige zusätzliche Schritte, die Sie berücksichtigen müssen.

## Fairness, Voreingenommenheit und Ethik

Heutzutage wirken sich maschinelle Lernalben auf Milliarden von Menschen aus. Du hast schon gehört, dass ich in anderen Videos Ethik erwähnt habe. Und ich hoffe, dass Sie, wenn Sie ein maschinelles Lernsystem aufbauen, das Menschen betrifft, darüber nachdenken, sicherzustellen, dass Ihr System einigermaßen fair und einigermaßen frei von Vorurteilen ist. Und dass Sie bei Ihrer Bewerbung einen ethischen Ansatz verfolgen. Werfen wir einen Blick auf einige Themen im Zusammenhang mit Fairness, Voreingenommenheit und Ethik. Leider gab es in der Geschichte des maschinellen Lernens einige Systeme, von denen einige weithin bekannt gemacht wurden und die ein völlig inakzeptables Maß an Voreingenommenheit aufwiesen. Es gab zum Beispiel eine Einstellung von zwei Personen, bei der sich einmal herausstellte, dass sie Frauen diskriminierte. Das Unternehmen, das das System gebaut hat, hat es nicht mehr verwendet, aber man wünscht sich, dass das System überhaupt nicht eingeführt worden wäre. Oder es gab auch ein gut dokumentiertes

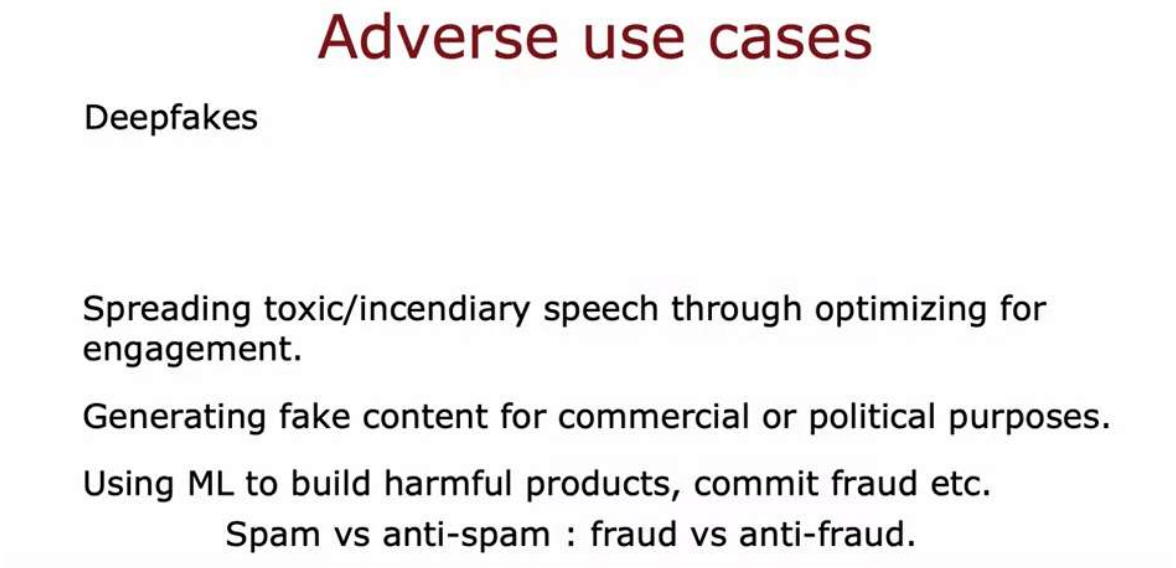
Beispiel für Gesichtserkennungssysteme, die dunkelhäutige Personen viel häufiger mit kriminellen Fahndungsfotos zuordnen als hellhäutige Personen.




Und das ist eindeutig nicht akzeptabel, und das sollten wir tun. Ja, das ist der Punkt, an dem die Community bei einem solchen Problem einfach keine Systeme baut und bereitstellt. Erstens gibt es Systeme, die Bankkreditgenehmigungen auf eine Weise erteilten, die voreingenommen und gegenüber Untergruppen diskriminierend war. Und wir möchten auch, dass Lernalgorithmen nicht die toxische Wirkung haben, negative Stereotypen zu verstärken. Ich habe zum Beispiel eine Tochter und wenn sie online nach bestimmten Berufen sucht und niemanden sieht, der wie sie aussieht, würde ich es hassen, wenn sie dadurch davon abgehalten würde, bestimmte Berufe zu ergreifen. Zusätzlich zu den Problemen der Voreingenommenheit und der fairen Behandlung von Einzelpersonen gab es auch nachteilige oder negative Anwendungsfälle von Algorithmen für maschinelles Lernen. Da war zum Beispiel diese viel zitierte und weithin angesehene Videoveröffentlichung mit vollständiger Offenlegung und vollständiger Transparenz.

Durch das Unternehmen Buzzfeed eines Deepfakes des ehemaligen US-Präsidenten Barack Obama und Sie können das gesamte Video tatsächlich online finden und ansehen, wenn Sie möchten. Aber das Unternehmen, das dieses Video erstellt hat, hat volle Transparenz und vollständige Offenlegung geleistet. Aber es wäre eindeutig unethisch, diese Technologie zu nutzen, um gefälschte Videos ohne Zustimmung und ohne Offenlegung zu erstellen. Wir haben leider auch gesehen, dass soziale Medien manchmal giftige oder aufrührerische Äußerungen verbreiten, weil die Optimierung für die Benutzereinbindung dazu geführt hat, dass Algorithmen dies tun.

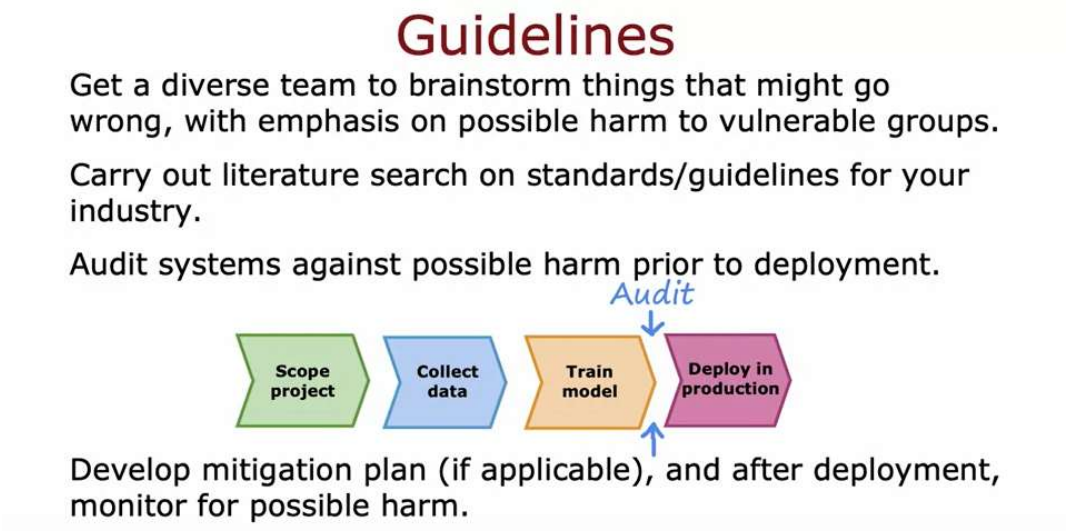
Es gab Bots, die zur Generierung gefälschter Inhalte entweder für kommerzielle Zwecke wie das Posten gefälschter Kommentare zu Produkten oder für politische Zwecke verwendet wurden. Und es gibt Nutzer des maschinellen Lernens, um schädliche Produkte zu entwickeln, Betrug zu begehen und so weiter. Und in Teilen der Welt des maschinellen Lernens, genau wie bei E-Mails, gab es einen Kampf zwischen den Spammern und der Anti-Spam-Community.




Ich erlebe heute beispielsweise in der Finanzbranche einen Kampf zwischen Leuten, die versuchen, Betrug zu begehen, und Leuten, die Betrug bekämpfen. Und leider wird maschinelles Lernen von einigen Betrügern und einigen Standards genutzt. Bauen Sie also um Himmels willen kein maschinelles Lernsystem auf, das sich negativ auf die Gesellschaft auswirkt. Und wenn Sie gebeten werden, an einer Bewerbung zu arbeiten, die Sie für unethisch halten, fordere ich Sie dringend auf, das zu tun, was es wert ist. Ich habe mir das Projekt mehrmals angeschaut und schien finanziell solide zu sein. Sie werden Geld für ein Unternehmen verdienen.

Aber ich habe das Projekt nur aus ethischen Gründen aufgegeben, weil ich denke, dass ich trotz der finanziellen Argumente das Gefühl habe, dass es die Welt noch schlimmer macht, und ich einfach nie an einem solchen Projekt beteiligt sein möchte. Ethik ist ein sehr kompliziertes und sehr reichhaltiges Thema, mit dem sich die Menschheit seit mindestens einigen 1000 Jahren beschäftigt. Als sich aI weiter verbreitete, habe ich tatsächlich mehrere Bücher über Philosophie und mehrere Bücher über Ethik gelesen, weil ich naiver Weise gehofft hatte, dass es dabei herauskommen würde, wenn es nur eine Checkliste mit fünf Dingen gäbe, die wir tun könnten, und zwar während wir diese tun Fünf Dinge, dann können wir ethisch sein, aber ich habe versagt.

Und ich glaube nicht, dass es jemals irgendjemandem gelungen ist, eine einfache Checkliste mit Dingen zu erstellen, die man tun muss, um so konkrete Leitlinien für ethisches Verhalten zu geben. Was ich Ihnen stattdessen mitteilen möchte, ist keine Checkliste, denn ich hatte noch nicht einmal eine Checkliste mit nur einigen allgemeinen Leitlinien und einigen Vorschlägen, wie Sie sicherstellen können, dass die Arbeit weniger voreingenommen, fairer und ethischer ist. Und ich hoffe, dass einige dieser Anleitungen, von denen einige relativ allgemein gehalten sind, Ihnen auch bei Ihrer Arbeit helfen werden. Hier sind einige Vorschläge, wie Sie Ihre Arbeit fairer, weniger voreingenommen und ethischer gestalten können, bevor Sie ein System einsetzen, das Schaden anrichten könnte.




Normalerweise versuche ich, ein vielfältiges Team zusammenzustellen, um ein Brainstorming über mögliche Dinge durchzuführen, die schiefgehen könnten, wobei der Schwerpunkt auf möglichen Schäden liegt. Zwei gefährdete Gruppen, die ich in meinem Leben oft erlebt habe, waren ein vielfältigeres Team, und mit vielfältig meine ich Diversität in mehreren Dimensionen, vom Geschlecht über die ethnische Zugehörigkeit bis hin zur Kultur und vielen anderen Merkmalen. Ich habe herausgefunden, dass vielfältigere Teams tatsächlich dazu führen, dass ein Team insgesamt besser in der Lage ist, Ideen für Dinge zu entwickeln, die schief gehen könnten, und es erhöht die Wahrscheinlichkeit, dass das Problem erkannt und behoben wird, bevor das System eingeführt wird und dadurch Schaden entsteht eine bestimmte Gruppe. Zusätzlich zu Diversität und Brainstorming. Ich habe es auch als nützlich empfunden, eine Literaturrecherche zu Standards oder Richtlinien für Ihre Branche oder Ihren bestimmten Anwendungsbereich durchzuführen. Beispielsweise gibt es in der Finanzbranche zunehmend etablierte Standards dafür, was es bedeutet, ein System zu sein. Sie wollen also entscheiden, wer diese beiden genehmigt, was es für ein solches System bedeutet, einigermaßen fair und frei von Voreingenommenheit zu sein, und welche Standards, die in verschiedenen Sektoren immer noch entstehen, Ihre Arbeit beeinflussen können, je nachdem, woran Sie arbeiten. Nachdem mögliche Probleme identifiziert wurden. Ich fand es nützlich, das System dann anhand dieser identifizierten Dimensionen eines möglichen Zuhauses zu prüfen. Vor der Bereitstellung haben Sie im letzten Video den gesamten Zyklus des maschinellen Lernprojekts gesehen. Und ein wichtiger Schritt, der oft eine entscheidende Verteidigungslinie gegen die Bereitstellung problematischer Elemente darstellt, ist, nachdem Sie das Modell trainiert haben. Wenn das Team jedoch vor dem Einsatz in der Produktion ein Brainstorming durchgeführt hat, kann es sein, dass es gegenüber bestimmten Untergruppen wie bestimmten Geschlechtern oder bestimmten ethnischen Gruppen voreingenommen ist. Anschließend können Sie das System anweisen, die Leistung zu messen, um festzustellen, ob es sich wirklich um eine Voreingenommenheit gegenüber bestimmten Geschlechtern, Ethnien oder anderen Untergruppen handelt, und um sicherzustellen, dass etwaige Probleme erkannt und behoben werden. Vor dem Einsatz. Schließlich fand ich es nützlich, gegebenenfalls einen Abhilfeplan zu entwickeln. Und ein einfacher Plan zur Schadensbegrenzung wäre ein Rollback auf das frühere System, von dem wir wussten, dass es einigermaßen fair war. Und auch nach der Bereitstellung können Sie den Schaden weiterhin überwachen, sodass Sie dann einen Schadensbegrenzungsplan auslösen und schnell handeln können, falls ein Problem behoben werden muss. Beispielsweise hatten alle Teams für selbstfahrende Autos vor der Einführung selbstfahrender Autos auf der Straße Schadensbegrenzungspläne entwickelt, um zu zeigen, was zu tun ist, falls das Auto jemals in einen Unfall verwickelt wird, sodass, falls das Auto jemals in einen Unfall verwickelt wurde, dies der Fall ist

Es gab bereits einen Schadensbegrenzungsplan, den sie sofort umsetzen konnten, anstatt ein Auto in einen Unfall zu verwickeln und erst im Nachhinein zu überlegen, was zu tun ist. Ich habe an vielen maschinellen Lernsystemen gearbeitet und möchte Ihnen die Themen Ethik, Fairness und Voreingenommenheit erläutern, die wir ernst nehmen sollten. Es ist nichts, was man abtun kann. Es ist nichts, was man wahrscheinlich nehmen sollte. Natürlich gibt es einige Projekte, die schwerwiegendere ethische Auswirkungen haben als andere. Wenn ich zum Beispiel ein neuronales Netzwerk aufbaue, um zu entscheiden, wie lange ich meine Kaffeebohnen rösten soll, sind die ethischen Implikationen deutlich geringer, als wenn ich beispielsweise ein System aufbaue, um zu entscheiden, welche Kredite ich leihe. Es werden Bankdarlehen gewährt, die für den Käufer erheblichen Schaden anrichten können. Aber ich hoffe, dass wir alle, die gemeinsam im Bereich des maschinellen Lernens arbeiten, diese Themen weiterhin besser diskutieren können. Erkennen Sie Probleme und beheben Sie sie, bevor sie Schaden anrichten, damit wir gemeinsam einige der Fehler vermeiden können, die die Welt des maschinellen Lernens zuvor gemacht hat, denn diese Dinge sind wichtig und die Systeme, die wir entwickelt haben, können viele Menschen betreffen. Das war's also mit dem Prozess der Entwicklung eines maschinellen Lernsystems und herzlichen Glückwunsch, dass Sie das Ende der erforderlichen Videos dieser Woche erreicht haben. Ich habe diese Woche nur zwei weitere optionale Videos zum Umgang mit verzerrten Datensätzen für Sie, und das bedeutet, Steve sagt, dass das Verhältnis von positiven zu negativen Beispielen sehr weit von 50, 50 entfernt ist. Und es stellt sich heraus, dass einige spezielle Techniken erforderlich sind, um maschinell zu reagieren Lernanwendungen wie diese. Ich hoffe, Sie im nächsten optionalen Video zum Umgang mit verzerrten Datensätzen zu sehen.

## Fehlermetriken für verzerrte Datensätze

Wenn Sie an einer Anwendung für maschinelles Lernen arbeiten, bei der das Verhältnis von positiven zu negativen Beispielen sehr verzerrt ist, sehr weit von 50:50 entfernt, dann stellt sich heraus, dass die üblichen Fehlermetriken wie Genauigkeit nicht so gut funktionieren. Beginnen wir mit einem Beispiel. Nehmen wir an, Sie trainieren einen binären Klassifikator, um eine seltene Krankheit bei Patienten anhand von Labortests oder anderen Daten der Patienten zu erkennen. Y ist gleich 1, wenn die Krankheit vorliegt, andernfalls ist y gleich 0. Angenommen, Sie stellen fest, dass Sie im Testsatz einen Fehler von einem Prozent erreicht haben, sodass Sie eine zu 99 Prozent korrekte Diagnose haben. Das scheint ein großartiges Ergebnis zu sein. Es stellt sich jedoch heraus, dass dies möglicherweise nicht so beeindruckend ist, wie es sich anhört, wenn es sich um eine seltene Krankheit handelt, also y gleich 1 ist, was sehr selten vorkommt. Wenn es sich insbesondere um eine seltene Krankheit handelt und nur 0,5 Prozent der Patienten in Ihrer Bevölkerung an dieser Krankheit leiden, dann drucken Sie, wenn Sie stattdessen das gerade genannte Programm geschrieben haben, y gleich 0 aus. Es sagt voraus, dass y immer gleich 0 ist . Dieser sehr einfache, sogar nicht lernende Algorithmus hat tatsächlich eine Genauigkeit von 99,5 Prozent oder einen Fehler von 0,5 Prozent, da er immer nur sagt, dass y gleich 0 ist. Dieser wirklich dumme Algorithmus übertrifft Ihren Lernalgorithmus, der einen Fehler von einem Prozent hatte, viel schlimmer als 0,5 Prozent Fehler. Aber ich denke, dass eine Software, die nur „y gleich 0“ ausgibt, kein sehr nützliches Diagnosetool ist. Was das wirklich bedeutet, ist, dass man nicht sagen kann, ob ein Fehler von einem Prozent tatsächlich ein gutes oder ein schlechtes Ergebnis ist. Insbesondere wenn Sie einen Algorithmus haben, der eine Genauigkeit von 99,5 Prozent erreicht, einen anderen, der eine Genauigkeit von 99,2 Prozent erreicht, und einen anderen, der eine Genauigkeit von 99,6 Prozent erreicht. Es ist schwierig zu wissen, welcher dieser Algorithmen tatsächlich der beste ist . Denn wenn Sie einen Algorithmus haben, der einen Fehler von 0,5 Prozent erreicht, einen anderen, der einen Fehler von einem Prozent erreicht, und einen anderen, der einen Fehler von 1,2 Prozent erreicht, ist es schwierig zu wissen, welcher dieser Algorithmen der beste ist. Da diejenige mit dem geringsten Fehler möglicherweise nicht besonders nützlich ist, ist eine Vorhersage wie diese, die immer vorhersagt, dass y gleich 0 ist,

und bei keinem Patienten die Diagnose gestellt wird, dass er an dieser Krankheit leidet. Es könnte durchaus sein, dass ein Algorithmus, der einen Fehler von einem Prozent hat, der aber zumindest bei einigen Patienten die Krankheit diagnostiziert, nützlicher sein könnte, als immer nur „y gleich 0“ auszugeben. Wenn wir an Problemen mit verzerrten Datensätzen arbeiten, verwenden wir normalerweise eine andere Fehlermetrik als nur den Klassifizierungsfehler, um herauszufinden, wie gut Ihr Lernalgorithmus funktioniert. Ein häufiges Paar von Fehlermetriken sind insbesondere Präzision und Erinnerung, die wir auf der Folie definieren. In diesem Beispiel ist y gleich eins die seltene Klasse, beispielsweise die seltene Krankheit, die wir möglicherweise erkennen möchten. Um insbesondere die Leistung eines Lernalgorithmus mit einer seltenen Klasse von nützlichen Elementen zu bewerten, ist es sinnvoll, eine sogenannte Verwirrungsmatrix zu erstellen, bei der es sich um eine Zwei-mal-Zwei-Matrix oder eine Zwei-mal-Zwei-Tabelle handelt, die wie folgt aussieht. Auf der Achse oben schreibe ich die eigentliche Klasse, die eins oder null sein kann. Auf der vertikalen Achse schreibe ich die vorhergesagte Klasse. Was hat Ihr Lernalgorithmus für ein bestimmtes Beispiel vorhergesagt, eins oder null? Um die Leistung Ihres Algorithmus beispielsweise im Kreuzvalidierungssatz oder Testsatz zu bewerten, zählen wir dann wie viele Beispiele? War die tatsächliche Klasse 1 und die vorhergesagte Klasse 1? Vielleicht haben Sie 100 Kreuzvalidierungsbeispiele und bei 15 davon hatte der Lernalgorithmus eins vorhergesagt und die tatsächliche Bezeichnung war auch eins. Hier würden Sie die Anzahl der Beispiele in C oder im Kreuzvalidierungssatz zählen, bei denen die tatsächliche Klasse Null war und Ihr Algorithmus Eins vorhergesagt hat. Vielleicht haben Sie dort und hier fünf Beispiele mit vorhergesagter Klasse 0 und tatsächlicher Klasse 1. Sie haben 10 Beispiele und sagen wir 70 Beispiele mit vorhergesagter Klasse 0 und tatsächlicher Klasse 0. In diesem Beispiel ist die Abweichung nicht so extrem wie bei mir auf der vorherigen Folie. Denn in diesen 100 Beispielen in Ihrem Kreuzvalidierungssatz haben wir durch vertikale Addition dieser Zahlen insgesamt 25 Beispiele, bei denen die tatsächliche Klasse eins war, und 75, bei denen die tatsächliche Klasse Null war. Sie werden auch feststellen, dass ich zur Kennzeichnung dieser vier Zellen in der Tabelle unterschiedliche Farben verwende. Ich werde diesen vier Zellen tatsächlich Namen geben. Wenn die tatsächliche Klasse eins und die vorhergesagte Klasse eins ist, nennen wir das ein echtes Positiv, weil Sie positiv vorhergesagt haben und es stimmte, dass es ein positives Beispiel gibt. In dieser Zelle unten rechts, in der die tatsächliche Klasse Null und die vorhergesagte Klasse Null ist, nennen wir das ein wahres Negativ, weil Sie ein Negativ vorhergesagt haben und es wahr war. Es war wirklich ein Negativbeispiel. Diese Zelle oben rechts wird als falsch positiv bezeichnet, da der Algorithmus ein positives Ergebnis vorhergesagt hat, dieses jedoch falsch war. Es ist nicht wirklich positiv, daher wird dies als falsch positiv bezeichnet. Diese Zelle wird als Anzahl falsch negativer Ergebnisse bezeichnet, da der Algorithmus Null vorhergesagt hat, diese aber falsch war. Es war nicht wirklich negativ. Die eigentliche Klasse war eine. Nachdem Sie die Klassifizierungen in diese vier Zellen unterteilt haben, können Sie zwei gängige Metriken berechnen: Präzision und Rückruf. Hier ist, was sie bedeuten. Die Präzision des Lernalgorithmus berechnet, welcher Anteil aller Patienten, bei denen wir vorhergesagt haben, dass y gleich 1 ist, tatsächlich an der seltenen Krankheit leidet. Mit anderen Worten ist Präzision definiert als die Anzahl der echten Positiven dividiert durch die als positiv eingestufte Anzahl. Mit anderen Worten: Welchen Bruchteil aller Beispiele, die Sie als positiv vorhergesagt haben, haben wir tatsächlich richtig gemacht? Eine andere Möglichkeit, diese Formel zu schreiben, wäre „wahre positive Ergebnisse“ dividiert durch „wahre positive Ergebnisse“ plus „falsche positive Ergebnisse“, denn durch die Summierung dieser Zelle und dieser Zelle erhält man am Ende die Gesamtzahl, die als positiv vorhergesagt wurde. In diesem Beispiel wäre der Zähler (echte Positive) 15 und würde durch 15 plus 5 geteilt, also wäre das 15 über 20 oder drei Viertel, also 0,75. Wir sagen also, dass dieser Algorithmus eine Präzision von 75 Prozent hat, weil er alles als positiv vorhergesagt hat. Von allen Patienten, von denen er glaubte, dass sie an dieser seltenen Krankheit leiden, hatte er in 75 Prozent der Fälle Recht. Die zweite Metrik, deren Berechnung nützlich ist, ist die Erinnerung. Und Recall fragt: Bei welchem Anteil aller

Patienten, die tatsächlich an der seltenen Krankheit leiden, haben wir korrekt festgestellt, dass sie daran leiden? Der Recall ist definiert als die Anzahl der echten Positiven dividiert durch die Anzahl der tatsächlichen Positiven. Alternativ können wir dies als Anzahl der echten Positiven dividiert durch die Anzahl der tatsächlichen Positiven schreiben. Nun, es ist diese Zelle plus diese Zelle. Es handelt sich also tatsächlich um die Anzahl der richtig positiven Ergebnisse plus die Anzahl der falsch negativen Ergebnisse, denn durch die Summierung dieser oberen linken Zelle und dieser unteren linken Zelle erhält man die Anzahl der tatsächlich positiven Beispiele. In unserem Beispiel wäre das 15 geteilt durch 15 plus 10, also 15 über 25, also 0,6 oder 60 Prozent. Dieser Lernalgorithmus hätte eine Genauigkeit von 0,75 und einen Rückruf von 0,60. Sie bemerken, dass Sie dadurch erkennen können, ob der Lernalgorithmus immer nur y gleich 0 ausgibt. Denn wenn es immer Null vorhersagt, wäre der Zähler dieser beiden Größen Null. Es hat keine wirklich positiven Aspekte. Insbesondere die Rückrufmetrik hilft Ihnen zu erkennen, ob der Lernalgorithmus ständig Null vorhersagt. Denn wenn Ihr Lernalgorithmus einfach y gleich 0 ausgibt, ist die Anzahl der echten Positiven Null, da er niemals positive Ergebnisse vorhersagt, und daher ist der Rückruf gleich Null geteilt durch die Anzahl der tatsächlichen Positiven, was gleich Null ist. Im Allgemeinen ist ein Lernalgorithmus mit Nullgenauigkeit oder Nullrückruf kein nützlicher Algorithmus. Aber nur als Randbemerkung: Wenn ein Algorithmus tatsächlich ständig Null vorhersagt, wird die Präzision tatsächlich undefiniert, weil sie tatsächlich über Null liegt. Null . Wenn ein Algorithmus jedoch in der Praxis nicht einmal ein einziges positives Ergebnis vorhersagt, sagen wir einfach, dass die Präzision ebenfalls gleich Null ist. Aber wir werden feststellen, dass die Berechnung von Präzision und Erinnerung es einfacher macht, zu erkennen, ob ein Algorithmus beide einigermaßen genau ist, denn wenn er sagt, dass ein Patient eine Krankheit hat, besteht eine gute Wahrscheinlichkeit, dass der Patient eine Krankheit hat, z. B. 0,75 In diesem Beispiel hilft es, sicherzustellen, dass von allen Patienten, die an dieser Krankheit leiden, eine angemessene Diagnose gestellt werden kann, so wie hier die Entdeckung von 60 Prozent. Wenn Sie eine seltene Klasse haben, achten Sie auf Präzision und Erinnerung und stellen Sie sicher, dass beide Zahlen angemessen hoch sind. Das hilft Ihnen hoffentlich dabei, sich zu vergewissern, dass Ihr Lernalgorithmus tatsächlich nützlich ist. Der Begriff „Recall“ wurde durch die Beobachtung motiviert, dass, wenn man eine Gruppe von Patienten oder eine Population von Patienten hat, Recall-Messwerte für alle Patienten sind, die an der Krankheit leiden, wie viele davon genau diagnostiziert worden wären. Wenn Sie also verzerrte Klassen oder eine seltene Klasse erkennen möchten, können Sie anhand von Präzision und Rückruf erkennen, ob Ihr Lernalgorithmus gute oder nützliche Vorhersagen macht. Nachdem wir nun über diese Metriken verfügen, mit denen wir feststellen können, wie gut Ihr Lernalgorithmus abschneidet, zeigen wir Ihnen im nächsten Video:

## Kompromiss zwischen Präzision und Erinnerung

Werfen wir einen Blick darauf, wie Sie einen Kompromiss zwischen Präzision und Rückruf finden, um die Leistung Ihres Lernalgorithmus zu optimieren. Im Idealfall bevorzugen wir Lernalgorithmen mit hoher Präzision und hohem Rückruf. Hohe Präzision würde bedeuten, dass, wenn eine Diagnose bei Patienten mit dieser seltenen Krankheit gestellt wird, der Patient wahrscheinlich auch darunter leidet und es sich um eine genaue Diagnose handelt. Eine hohe Erinnerung bedeutet, dass der Algorithmus bei einem Patienten mit dieser seltenen Krankheit wahrscheinlich korrekt erkennt, dass er tatsächlich an dieser Krankheit leidet. Es stellt sich jedoch heraus, dass es in der Praxis oft einen Kompromiss zwischen Präzision und Erinnerung gibt. In diesem Video werfen wir einen Blick auf diesen Kompromiss und wie Sie bei diesem Kompromiss einen guten Punkt auswählen können. Hier sind die Definitionen aus dem letzten Video zu Präzision und Rückruf, ich schreibe sie einfach hier auf. Nun, Sie erinnern sich, Präzision ist die Anzahl der wirklich positiven Ergebnisse geteilt durch die Gesamtzahl der vorhergesagten positiven Ergebnisse, und Erinnerung ist die Anzahl der wirklich

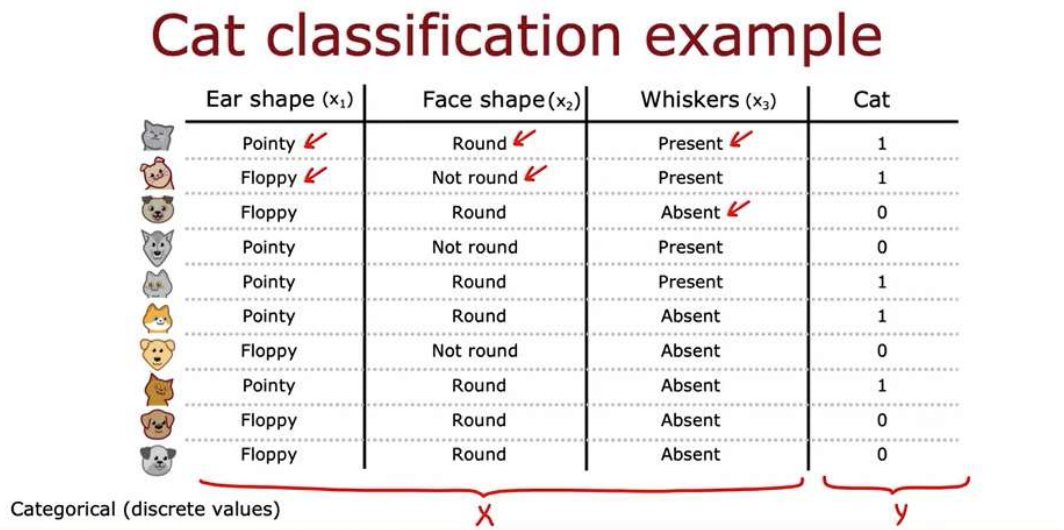
positiven Ergebnisse geteilt durch die tatsächliche Gesamtzahl der positiven Ergebnisse. Wenn Sie die logistische Regression verwenden, um Vorhersagen zu treffen, gibt das logistische Regressionsmodell Zahlen zwischen 0 und 1 aus. Normalerweise würden wir die Ausgabe der logistischen Regression auf einen Schwellenwert von 0,5 beschränken und 1 vorhersagen, wenn f von x größer als gleich 0,5 ist, und vorhersagen 0, wenn es kleiner als 0,5 ist. Aber nehmen wir an, wir wollen vorhersagen, dass y gleich 1 ist. Das heißt, die seltene Krankheit liegt nur dann vor, wenn wir sehr sicher sind. Wenn wir davon ausgehen, dass der Patient eine Krankheit hat, müssen wir ihn nach unserer Philosophie möglicherweise zu einer möglicherweise invasiven und teuren Behandlung schicken. Wenn die Folgen der Krankheit nicht so schlimm sind, auch wenn sie nicht aggressiv behandelt wird, möchten wir möglicherweise nur dann vorhersagen, dass y gleich 1 ist, wenn wir sehr zuversichtlich sind. In diesem Fall können wir einen höheren Schwellenwert festlegen, bei dem wir nur dann vorhersagen, dass y 1 ist, wenn f von x größer oder gleich 0,7 ist. Das bedeutet also, dass wir y nur dann 1 vorhersagen, wenn wir mindestens 70 sind Prozent sicher und nicht nur 50 Prozent sicher, sodass diese Zahl ebenfalls 0,7 beträgt. Beachten Sie, dass diese beiden Zahlen gleich sein müssen, da Sie nur dann 1 oder 0 vorhersagen, wenn Sie größer oder gleich oder kleiner als diese Zahl sind. Wenn Sie diesen Schwellenwert erhöhen, sagen Sie nur dann voraus, dass y gleich 1 ist, wenn Sie hübsch sind Zuversichtlich und das bedeutet, dass die Präzision zunimmt, denn wann immer Sie eine Vorhersage treffen, ist die Wahrscheinlichkeit größer, dass Sie richtig liegen. Eine Anhebung der Schwellenwerte führt also zu einer höheren Präzision, aber es führt auch zu einer geringeren Erinnerung, da wir nun seltener eine Vorhersage vorhersagen Daher werden wir von der Gesamtzahl der Patienten mit dieser Krankheit weniger korrekt diagnostizieren. Wenn Sie diesen Schwellenwert auf 0,7 erhöhen, erhalten Sie eine höhere Präzision, aber eine geringere Erinnerung. Wenn Sie nur dann vorhersagen möchten, dass y gleich 1 ist, wenn Sie sehr sicher sind, können Sie diesen sogar auf 0,9 erhöhen. Das führt zu einer noch höheren Präzision. Wenn Sie also vorhersagen, dass der Patient an der Krankheit leidet, haben Sie wahrscheinlich Recht und dadurch erhalten Sie eine sehr hohe Präzision. Der Rückruf wird noch weiter nach unten gehen. Angenommen, wir möchten andererseits vermeiden, zu viele Fälle der seltenen Krankheit zu übersehen. Wenn wir also im Zweifelsfall y gleich 1 vorhersagen möchten, könnte dies der Fall sein, wenn die Behandlung nicht zu invasiv, schmerzhaft oder teuer ist Aber wenn eine Krankheit unbehandelt bleibt, hat dies weitaus schlimmere Folgen für den Patienten. In diesem Fall könnte man sagen: Im Zweifelsfall sollten wir im Interesse der Sicherheit einfach vorhersagen, dass sie es haben, und sie für eine Behandlung in Betracht ziehen, denn unbehandelte Fälle könnten ziemlich schlimm sein. Wenn dies für Ihre Anwendung der bessere Weg ist, Entscheidungen zu treffen, dann würden Sie diesen Schwellenwert stattdessen senken, beispielsweise auf 0,3 setzen. In diesem Fall sagen Sie „Eins“ voraus, solange Sie davon ausgehen, dass die Wahrscheinlichkeit, dass die Krankheit vorliegt, bei 30 Prozent oder mehr liegt, und Sie sagen „Null“ nur dann voraus, wenn Sie ziemlich sicher sind, dass die Krankheit nicht vorliegt. Wie Sie sich vorstellen können, werden die Auswirkungen auf Präzision und Erinnerung entgegengesetzt zu dem sein, was Sie hier oben gesehen haben, und eine Senkung dieses Schwellenwerts wird zu einer geringeren Präzision führen, weil wir jetzt lockerer sind und eher dazu bereit sind, eins vorherzusagen, selbst wenn wir es sind. Ich bin mir nicht sicher, aber um eine höhere Erinnerung zu erreichen, werden wir wahrscheinlich mehr von ihnen korrekt identifizieren, da es so viele Patienten mit dieser Krankheit gibt. Allgemeiner gesagt haben wir die Flexibilität, einen Wert nur dann vorherzusagen, wenn f über einem bestimmten Schwellenwert liegt, und durch die Wahl dieses Schwellenwerts können wir verschiedene Kompromisse zwischen Präzision und Rückruf eingehen. Es stellt sich heraus, dass es bei den meisten Lernalgorithmen einen Kompromiss zwischen Präzision und Erinnerung gibt. Präzision und Rückruf liegen beide zwischen Null und Eins. Wenn Sie einen sehr hohen Schwellenwert festlegen, beispielsweise einen Schwellenwert von 0,99, geben Sie die Eingabe mit sehr hoher Präzision, aber niedrigerem Rückruf ein, und wenn Sie den Wert dieses Schwellenwerts verringern, beenden Sie die Eingabe Erstellen Sie eine Kurve, die Präzision und

Rückruf abwägt, bis Sie schließlich, wenn Sie einen sehr niedrigen Schwellenwert haben, der Schwellenwert also 0,01 beträgt, am Ende eine sehr geringe Präzision, aber einen relativ hohen Rückruf erhalten. Manchmal können Sie durch Zeichnen dieser Kurve versuchen, einen Schwellenwert auszuwählen, der der Auswahl eines Punktes auf dieser Kurve entspricht. Die Bilanzen, die Kosten für falsch-positive und falsch-negative Ergebnisse oder die Bilanzen, die Vorteile hoher Präzision und hoher Erinnerung. Durch die grafische Darstellung von Präzision und Rückruf für verschiedene Schwellenwerte können Sie einen gewünschten Punkt auswählen. Beachten Sie, dass die Auswahl des Schwellenwerts mit der Kreuzvalidierung nicht wirklich möglich ist, da es an Ihnen liegt, die besten Punkte anzugeben. Bei vielen Anwendungen müssen Sie letztendlich den Schwellenwert für den Kompromiss zwischen Präzision und Rückruf manuell auswählen. Es stellt sich heraus, dass es, wenn Sie Präzision und Rückruf automatisch abwägen möchten, anstatt dies selbst tun zu müssen, eine andere Metrik namens F1-Score gibt, die manchmal verwendet wird, um Präzisionsrückruf automatisch zu kombinieren, um Ihnen bei der Auswahl des besten oder besten Werts zu helfen Kompromiss zwischen den beiden. Eine Herausforderung beim Präzisionsrückruf besteht darin, dass Sie Ihre Algorithmen jetzt anhand von zwei verschiedenen Metriken bewerten. Wenn Sie also drei verschiedene Algorithmen trainiert haben und die Zahlen für den Präzisionsrückruf so aussehen, ist es nicht so offensichtlich, wie Sie den zu verwendenden Algorithmus auswählen. Wenn es einen Algorithmus gäbe, der eine bessere Präzision und einen besseren Rückruf bietet, dann würden Sie sich wahrscheinlich für diesen entscheiden. Aber in diesem Beispiel hat Algorithmus 2 die höchste Präzision, aber Algorithmus 3 hat den höchsten Rückruf, und Algorithmus 1 tauscht die beiden dazwischen aus, sodass offensichtlich kein Algorithmus die beste Wahl ist. Um Ihnen bei der Entscheidung zu helfen, welchen Algorithmus Sie wählen sollten, kann es hilfreich sein, eine Möglichkeit zu finden, Präzision und Erinnerung in einem einzigen Wert zu kombinieren, sodass Sie einfach schauen können, welcher Algorithmus den höchsten Wert hat, und sich vielleicht für diesen entscheiden. Eine Möglichkeit, Präzision und Erinnerung zu kombinieren, besteht darin, den Durchschnitt zu ermitteln. Dies erweist sich als keine gute Methode, daher empfehle ich diese nicht wirklich. Wenn wir jedoch den Durchschnitt nehmen würden, erhalten wir 0,45, 0,4 und 0,5. Aber es stellt sich heraus, dass die Berechnung des Durchschnitts und die Auswahl des Algorithmus mit dem höchsten Durchschnitt zwischen Präzision und Rückruf nicht so gut funktionieren, weil dieser Algorithmus eine sehr geringe Präzision hat, und tatsächlich entspricht dies möglicherweise einem Algorithmus, der tatsächlich y-Gleichheiten ausgibt 1 und die Diagnose, dass alle Patienten an der Krankheit leiden, deshalb ist die Erinnerung perfekt, aber die Präzision ist sehr gering. Algorithmus 3 ist eigentlich kein besonders nützlicher Algorithmus, auch wenn der Durchschnitt zwischen Präzision und Rückruf recht hoch ist. Lassen Sie uns nicht den Durchschnitt zwischen Präzision und Erinnerung verwenden. Stattdessen ist die gebräuchlichste Methode zum Kombinieren von Präzisionsrückrufen eine Berechnung, die als F1-Score bezeichnet wird. Der F1-Score ist eine Möglichkeit, P- und R-Präzision und -Recall zu kombinieren, wobei jedoch der niedrigere dieser Werte stärker hervorgehoben wird. Denn es stellt sich heraus, dass ein Algorithmus mit sehr geringer Präzision oder sehr geringem Rückruf nicht besonders nützlich ist. Der F1-Score ist eine Methode zur Berechnung eines Durchschnittswerts, bei dem der niedrigere Wert stärker berücksichtigt wird. Die Formel zur Berechnung des F1-Scores lautet wie folgt: Sie berechnen eins über P und eins über R, mitteln sie und bilden dann die Umkehrung davon. Anstatt den P- und R-Präzisionsrückruf zu mitteln, werden wir eins über P und eins über R mitteln und dann eins darüber nehmen. Wenn man diese Gleichung vereinfacht, kann sie auch wie folgt berechnet werden. Durch die Mittelung von eins über P und eins über R wird jedoch deutlich stärker hervorgehoben, ob entweder P oder R sehr klein sind. Wenn Sie den F1-Score für diese drei Algorithmen berechnen würden, würden Sie feststellen, dass der F1-Score für Algorithmus 1 0,444 und für den zweiten Algorithmus 0,175 beträgt. Sie bemerken, dass 0,175 viel näher am niedrigeren Wert liegt als am höheren Wert und für den dritten Algorithmus 0,0392 beträgt. Der F1-Score gibt Aufschluss über den

Kompromiss zwischen Präzision und Erinnerung, und in diesem Fall sagt er uns, dass der erste Algorithmus möglicherweise besser ist als der zweite oder der dritte Algorithmus. In der Mathematik wird diese Gleichung übrigens auch als harmonisches Mittel von P und R bezeichnet, und das harmonische Mittel ist eine Möglichkeit, einen Durchschnitt zu bilden, der die kleineren Werte stärker hervorhebt. Für die Zwecke dieses Kurses brauchen Sie sich jedoch keine Gedanken über die Terminologie des harmonischen Mittelwerts zu machen. Herzlichen Glückwunsch zum Erreichen des letzten Videos dieser Woche und vielen Dank auch dafür, dass Sie mir durch diese beiden optionalen Videos gefolgt sind. In dieser Woche haben Sie viele praktische Tipps und praktische Ratschläge für den Aufbau eines Systems für maschinelles Lernen gelernt. Wenn Sie diese Ideen anwenden, werden Sie meiner Meinung nach sehr effektiv bei der Entwicklung von Algorithmen für maschinelles Lernen sein. Nächste Woche werden wir noch einmal über einen weiteren sehr leistungsstarken Algorithmus für maschinelles Lernen sprechen. Tatsächlich denke ich, dass neuronale Netze und Entscheidungsbäume von den fortschrittlichen Techniken, die wir in vielen kommerziellen Produktionsumgebungen verwenden, ganz oben auf der Liste stehen. Nächste Woche werden wir über Entscheidungsbäume sprechen, die meiner Meinung nach eine weitere sehr leistungsstarke Technik sein werden, die Sie auch zum Erstellen vieler erfolgreicher Anwendungen verwenden werden. Ich freue mich darauf, Sie nächste Woche zu sehen.

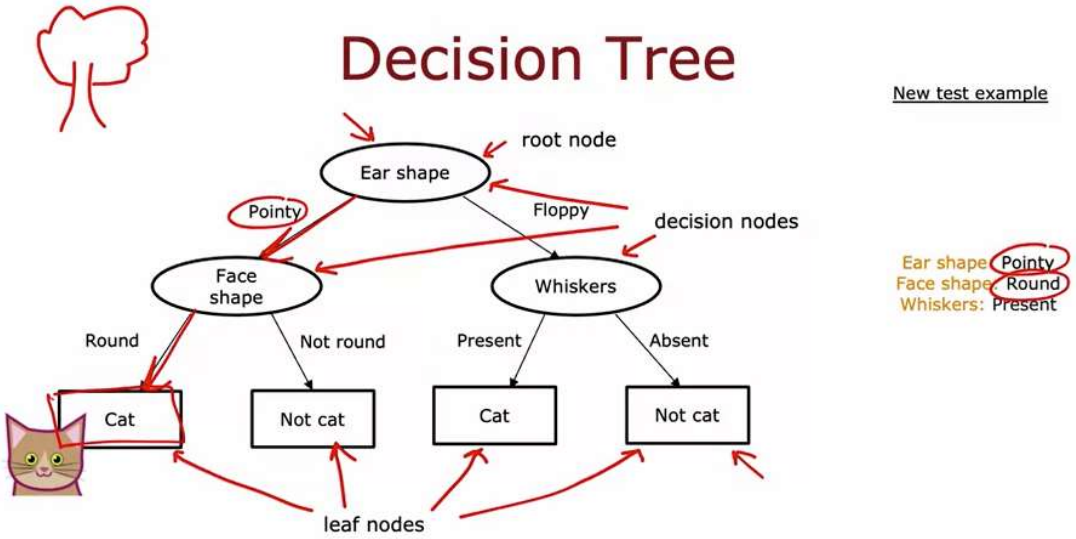
## Entscheidungsbaummodell

Willkommen zur letzten Woche dieses Kurses über fortgeschrittene Lernalgorithmen. Ein Grund dafür, dass der Lernalgorithmus sehr leistungsfähig ist, ist der Grund dafür, dass wir viele Anwendungen verwenden. Viele von ihnen verwenden auch Entscheidungsbäume und Baumensembles, um Wettbewerbe im maschinellen Lernen zu gewinnen. Trotz aller Erfolge von Entscheidungsbäumen haben sie in der Wissenschaft irgendwie nicht so viel Beachtung gefunden, und daher hört man vielleicht nicht annähernd so oft von Entscheidungsbäumen, aber es ist ein Werkzeug, das es sich lohnt, in Ihrem Werkzeugkasten zu haben. In dieser Woche lernen wir Entscheidungsbäume kennen und Sie erfahren, wie Sie sie für sich selbst nutzen können.



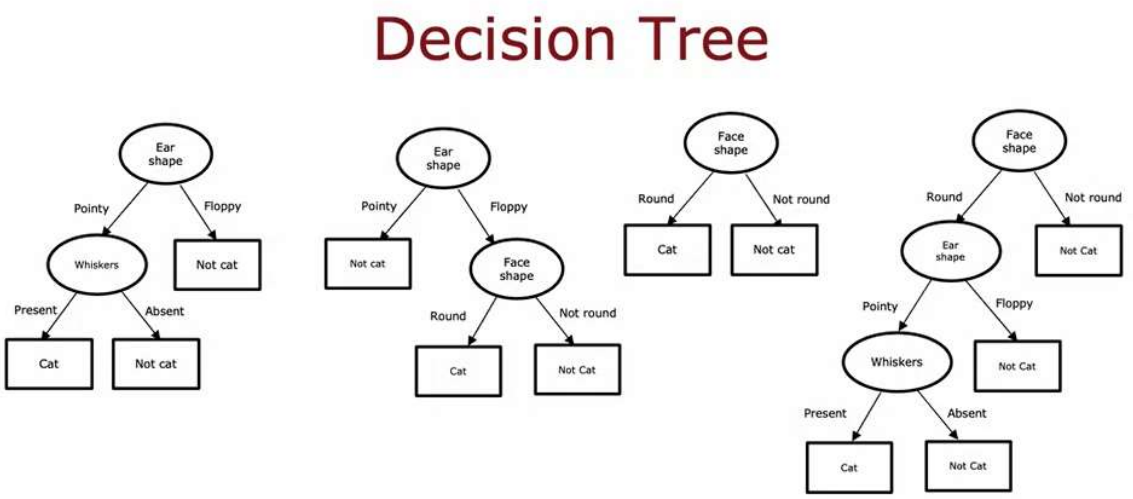

Um zu erklären, wie Entscheidungsbäume funktionieren, werde ich diese Woche als laufendes Beispiel ein Beispiel für eine Katzenklassifizierung verwenden. Sie leiten ein Katzenadoptionszentrum und möchten angesichts einiger Funktionen einen Klassifikator trainieren, der Ihnen schnell sagt, ob es sich bei einem Tier um eine Katze handelt oder nicht. Ich habe hier 10 Trainingsbeispiele. Zu jedem dieser 10 Beispiele werden wir Merkmale bezüglich der Ohrform und Gesichtsform des Tieres, ob es

Schnurrhaare hat, und dann die Grundwahrheitsbezeichnung angeben, die Sie dieser Tierkatze vorhersagen möchten. Das erste Exemplar hat spitze Ohren, ein rundes Gesicht, Schnurrhaare sind vorhanden und es handelt sich um eine Katze. Das zweite Beispiel hat Schlappohren, die Gesichtsform ist nicht rund, es sind Schnurrhaare vorhanden, und ja, das ist eine Katze, und so weiter für die restlichen Beispiele. Dieser Datensatz enthält fünf Katzen und fünf Hunde. Die Eingabemerkmale X sind diese drei Spalten, und die Zielausgabe, die Sie vorhersagen möchten, Y, ist diese letzte Spalte. Ist das eine Katze oder nicht? In diesem Beispiel nehmen die Features X kategoriale Werte an. Mit anderen Worten: Die Merkmale nehmen nur wenige diskrete Werte an. Ihre Formen sind entweder spitz oder schlaff. Die Gesichtsform ist entweder rund oder nicht rund und Schnurrhaare sind entweder vorhanden oder fehlen. Dies ist eine binäre Klassifizierungsaufgabe, da die Beschriftungen ebenfalls Eins oder Null sind. Derzeit nehmen die Merkmale X_1, X_2 und X_3 jeweils nur zwei mögliche Werte an.




Wir werden später in dieser Woche über Features sprechen, die mehr als zwei mögliche Werte annehmen können, sowie über Features mit kontinuierlichem Wert. Was ist ein Entscheidungsbaum? Hier ist ein Beispiel für ein Modell, das Sie erhalten könnten, nachdem Sie einen Entscheidungsbaum-Lernalgorithmus auf dem Datensatz trainiert haben, den Sie gerade gesehen haben. Das vom Lernalgorithmus ausgegebene Modell sieht aus wie ein Baum, und ein solches Bild nennen Informatiker einen Baum. Wenn es für Sie nicht so aussieht wie die biologischen Bäume, die Sie da draußen sehen, ist das in Ordnung, machen Sie sich darüber keine Sorgen. Wir gehen ein Beispiel durch, um sicherzustellen, dass diese Informatikdefinition eines Baums auch für Sie Sinn macht. Jedes dieser Ovale oder Rechtecke wird im Baum als Knoten bezeichnet. Dieses Modell funktioniert folgendermaßen: Wenn Sie ein neues Testexemplar haben, hat sie eine Katze, bei der die Ohrform spitz ist, die Gesichtsform rund ist und Schnurrhaare vorhanden sind. Die Art und Weise, wie dieses Modell dieses Beispiel betrachtet und eine Klassifizierungsentscheidung trifft, besteht darin, mit diesem Beispiel am obersten Knoten des Baums zu beginnen, der als Wurzelknoten des Baums bezeichnet wird, und wir werden uns das darin geschriebene Merkmal ansehen, nämlich Ohrform. Basierend auf dem Wert der Ohrform dieses Beispiels gehen wir entweder nach links oder nach rechts. Der Wert der Ohrform in diesem Beispiel ist spitz, und so gehen wir etwa den linken Ast des Baums hinunter und landen bei diesem ovalen Knoten hier drüben. Wir schauen uns dann die Gesichtsform dieses Beispiels an, die sich als rund herausstellt, und folgen daher diesem Pfeil hierher. Der Algorithmus schließt daraus, dass es sich hierbei um eine Katze handelt. Sie gelangen zu diesem Knoten und der Algorithmus trifft eine Vorhersage, dass es sich um eine Katze handelt. Was ich auf dieser Folie gezeigt habe, ist ein spezifisches Entscheidungsbaummodell. Um etwas mehr Terminologie einzuführen: Dieser oberste Knoten im Baum wird Wurzelknoten genannt. Alle diese

Knoten, also alle diese ovalen Formen, mit Ausnahme der Kästchen unten, werden alle als Entscheidungsknoten bezeichnet. Sie sind Entscheidungsknoten, weil sie ein bestimmtes Merkmal betrachten und Sie dann basierend auf dem Wert des Merkmals dazu veranlassen, zu entscheiden, ob Sie im Baum nach links oder rechts gehen möchten. Schließlich werden diese Knoten unten, diese rechteckigen Kästchen, Blattknoten genannt. Sie machen eine Vorhersage. Wenn Sie die Definitionen von Bäumen durch Informatiker noch nie gesehen haben, erscheint es vielleicht nicht intuitiv, dass die Wurzeln des Baumes oben sind und die Blätter des Baumes unten. Vielleicht kann man sich das so vorstellen, dass es sich eher um eine hängende Zimmerpflanze handelt, bei der die Wurzeln oben liegen und die Blätter dann dazu neigen, auf den Boden des Baumes zu fallen. Auf dieser Folie habe ich nur ein Beispiel eines Entscheidungsbaums gezeigt. Hier sind ein paar andere. Dies ist ein anderer Entscheidungsbaum für den Versuch, Katze und Nicht-Katze zu klassifizieren. Um in diesem Baum eine Klassifizierungsentscheidung zu treffen, würden Sie wieder bei diesem obersten Wurzelknoten beginnen. Abhängig von der Ohrform eines Beispiels würden Sie entweder nach links oder nach rechts gehen. Wenn die Ohrform spitz ist, schauen Sie sich die Schnurrhaare-Funktion an und je nachdem, ob Schnurrhaare vorhanden sind oder nicht, gehen Sie nach links oder rechts, um Katze und Nicht-Katze zu gewinnen und zu klassifizieren. Nur zum Spaß, hier ist ein zweites Beispiel eines Entscheidungsbaums, hier ein drittes und hier ein viertes. Unter diesen verschiedenen Entscheidungsbäumen schneiden einige bei den Trainingssätzen oder bei den Kreuzvalidierungs- und Testsätzen besser und andere schlechter ab.

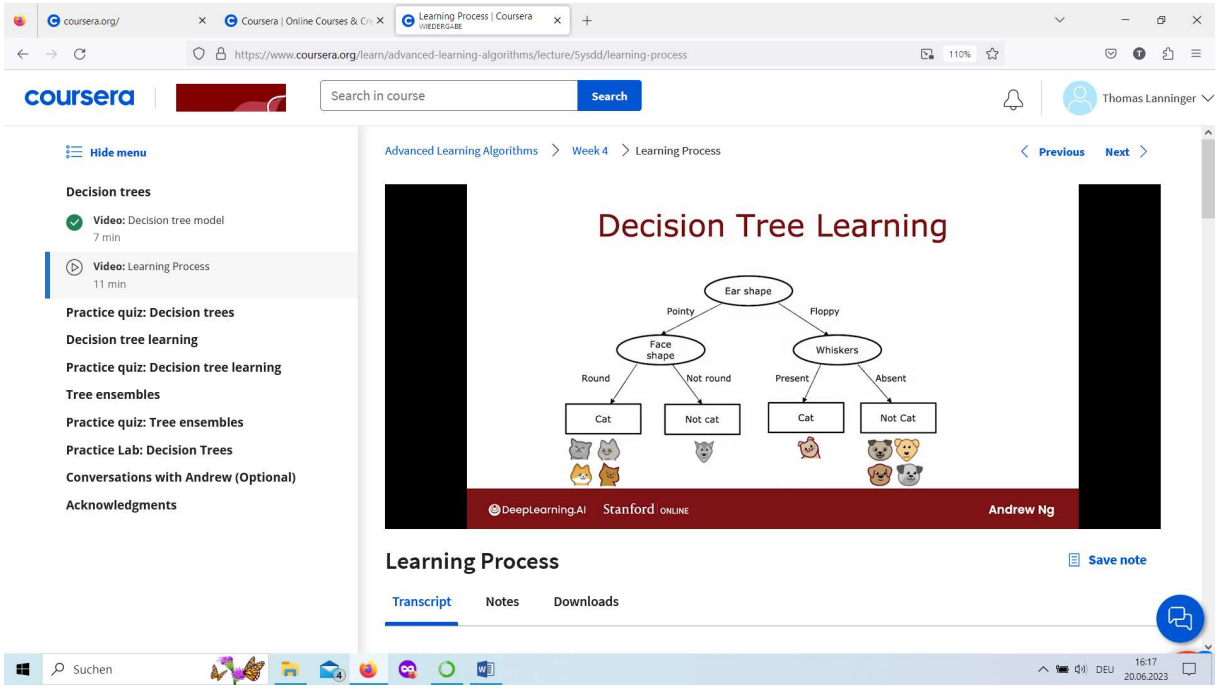



Die Aufgabe des Entscheidungsbaum-Lernalgorithmus besteht darin, aus allen möglichen Entscheidungsbäumen einen auszuwählen, der hoffentlich im Trainingssatz gut abschneidet und sich dann idealerweise auch gut auf neue Daten wie Ihre Kreuzvalidierungs- und Testsätze verallgemeinern lässt Also. Es scheint, als gäbe es viele verschiedene Entscheidungsbäume, die man für eine bestimmte Anwendung erstellen könnte. Wie bringt man einen Algorithmus dazu, einen bestimmten Entscheidungsbaum basierend auf einem Trainingssatz zu lernen? Schauen wir uns das im nächsten Video an.

## Lernprozess

Der Prozess zum Erstellen eines Entscheidungsbaums anhand eines Trainingssatzes umfasst einige Schritte. In diesem Video werfen wir einen Blick auf den Gesamtprozess dessen, was Sie tun müssen, um einen Entscheidungsbaum zu erstellen. Angenommen ein Trainingssatz mit 10 Beispielen von Katzen und Hunden, wie Sie ihn im letzten Video gesehen haben. Der erste Schritt des

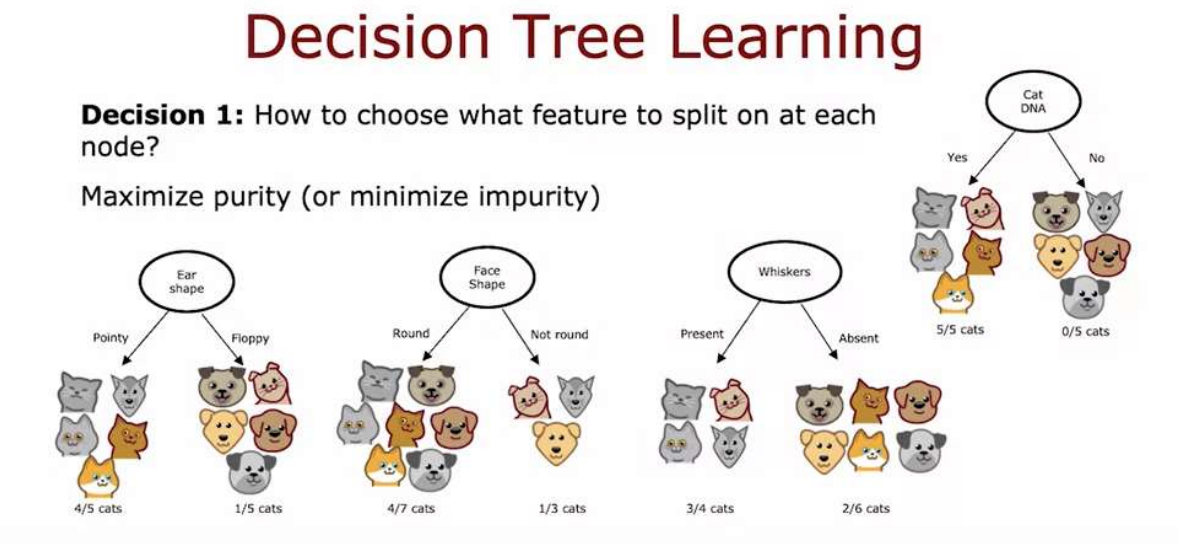
Entscheidungsbaumlernens besteht darin, zu entscheiden, welche Funktion am Wurzelknoten verwendet werden soll. Das ist der erste Knoten ganz oben im Entscheidungsbaum. Über einen Algorithmus, über den wir in den nächsten Videos sprechen werden. Nehmen wir an, wir haben uns entschieden, das Ohrform-Feature als Merkmal und Wurzelknoten auszuwählen. Das heißt, wir werden uns alle unsere Trainingsbeispiele ansehen, alle hier gezeigten Tangentenbeispiele. Ich teile sie nach dem Wert der Ohrformfunktion auf. Lassen Sie uns insbesondere die fünf Beispiele mit spitzen Ohren heraussuchen und sie nach links verschieben. Wählen wir die fünf Beispiele mit Schlappohren aus und verschieben wir sie nach rechts unten. Der zweite Schritt besteht darin, sich nur auf den linken Teil oder manchmal auch den linken Zweig des Entscheidungsbaums zu konzentrieren, um zu entscheiden, welche Knoten dort platziert werden sollen. Insbesondere, auf welche Funktion wir aufteilen möchten oder welche Funktion wir als nächstes verwenden möchten. Über einen Algorithmus, über den wir später in dieser Woche sprechen werden.




Nehmen wir an, Sie entscheiden sich, dort die Gesichtsformfunktion zu verwenden. Was wir jetzt tun werden, ist, diese fünf Beispiele zu nehmen und diese fünf Beispiele basierend auf ihrem Wert der Gesichtsform in zwei Teilmengen aufzuteilen. Wir nehmen die vier dieser fünf Beispiele mit runder Gesichtsform und verschieben sie nach links unten. Das eine Beispiel mit einer nicht runden Gesichtsform und verschieben Sie es nach rechts unten. Schließlich stellen wir fest, dass diese vier Beispiele alle Katzen sind, vier davon sind Katzen. Anstatt weiter zu spalten, wurde ein Blattknoten erstellt, der vorhersagt, dass Dinge, die darauf zurückzuführen sind, dass keine anderen Katzen vorhanden sind. Hier stellen wir fest, dass keines der Beispiele, null der einen Beispiele, Katzen sind oder dass 100 Prozent der Beispiele hier Hunde sind. Wir können hier einen Blattknoten erstellen, der eine Vorhersage macht, dass es sich nicht um eine Katze handelt. Nachdem wir dies im linken Teil bis zum linken Zweig dieses Entscheidungsbaums durchgeführt haben, wiederholen wir nun einen ähnlichen Vorgang im rechten Teil oder rechten Zweig dieses Entscheidungsbaums. Konzentrieren Sie sich auf diese fünf Beispiele, die einen Kapitän für Hunde enthalten. Wir müssten hier eine Funktion auswählen, um diese fünf Beispiele weiter aufzuteilen. Wenn wir uns am Ende für die Whiskers-Funktion entscheiden, würden wir diese fünf Beispiele dann aufteilen, je nachdem, wo die Whiskers vorhanden sind oder nicht, etwa so. Sie bemerken, dass eines von einem Beispiel auf der linken Seite für Katzen steht und Null von vier Beispielen Katzen sind. Jeder dieser Knoten ist völlig rein, das heißt, alle Katzen oder nicht Katzen und es gibt keine Mischung aus Katzen und Hunden mehr. Wir

können diese Blattknoten erstellen, indem wir links eine Katzenvorhersage und hier rechts eine Schlummertrunkvorhersage treffen. Dies ist ein Prozess zum Aufbau eines Entscheidungsbaums. Während dieses Prozesses mussten wir in verschiedenen Schritten des Algorithmus einige wichtige Entscheidungen treffen. Lassen Sie uns besprechen, was diese wichtigen Entscheidungen waren, und wir werden in den nächsten Videos weiter darüber sprechen, wie diese Entscheidungen zu treffen sind. Die erste wichtige Entscheidung war: Wie wählen Sie aus, welche Funktionen für die Aufteilung an jedem Knoten verwendet werden sollen? Am Wurzelknoten sowie am linken und rechten Zweig des Entscheidungsbaums mussten wir entscheiden, ob es an diesem Knoten einige Beispiele gab, die eine Mischung aus Katzen und Hunden umfassten. Möchten Sie nach dem ohrenförmigen Merkmal, dem Gesichtsmerkmal oder dem Schnurrbartmerkmal unterscheiden? Im nächsten Video werden wir sehen, dass Entscheidungsbäume auswählen, nach welcher Funktion aufgeteilt wird, um zu versuchen, die Reinheit zu maximieren.

Mit Reinheit meine ich, dass Sie zu den Teilmengen gelangen möchten, die allen Katzen oder allen Hunden möglichst nahe kommen. Wenn wir beispielsweise eine Funktion hätten, die besagt, ob dieses Tier Katzen-DNA hat, hätten wir diese Funktion eigentlich nicht. Aber wenn wir das getan hätten, hätten wir dieses Feature am Wurzelknoten aufteilen können, was zu fünf von fünf Katzen im linken Zweig und null der fünf Katzen im rechten Zweig geführt hätte. Sowohl diese linke als auch die rechte Teilmenge der Daten sind völlig rein, was bedeutet, dass es in beiden dieser linken und rechten Teilzweige nur eine Klasse gibt, entweder nur Katzen oder nicht nur Katzen, weshalb das Katzen-DNA-Merkmal wäre, wenn wir dieses Merkmal hätten , wäre eine tolle Funktion gewesen. Aber angesichts der Merkmale, die wir tatsächlich haben, mussten wir entscheiden, wie die Aufteilung nach Jahresform ist, was dazu führt, dass vier der fünf Beispiele auf der linken Seite Katzen sind und eines der fünf Beispiele auf der rechten Seite Katzen oder Gesichtsform sind wo es dazu führte, dass die vier der sieben auf der linken Seite und einer der drei auf der rechten Seite, oder Schnurrhaare, entstanden, was dazu führte, dass drei von vier Exemplaren auf der linken Seite geworfen wurden und zwei von sechs auf der rechten Seite keine Katzen waren.



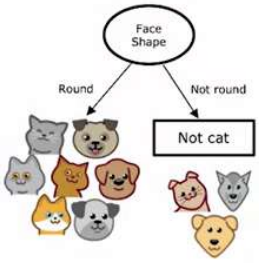

Der Entscheidungsbaum-Lernalgorithmus muss zwischen Ohrform, Gesichtsform und Schnurrhaaren wählen. Welches dieser Merkmale führt zu der größten Reinheit der Beschriftungen auf den linken und rechten Unterzweigen? Denn wenn Sie zu einer hochreinen Teilmenge von Beispielen gelangen, können Sie entweder Katze vorhersagen oder Nicht-Katze vorhersagen und es größtenteils richtig machen. Im nächsten Video zum Thema Entropie werden wir darüber sprechen, wie man Unreinheiten abschätzt und wie man sie minimiert.

Die erste Entscheidung, die wir beim Erlernen eines Entscheidungsbaums treffen müssen, ist die Auswahl der Funktion, des Salons und der einzelnen Knoten. Die zweite wichtige Entscheidung, die Sie beim Erstellen eines Entscheidungsbaums treffen müssen, besteht darin, zu entscheiden, wann Sie mit der Aufteilung aufhören. Die Kriterien, die wir gerade verwenden, waren, bis ich weiß, dass es entweder 100 Prozent, alle Katzen, oder 100 Prozent Hunde und keine Katzen gibt. Denn an diesem Punkt erscheint es natürlich, einen Blattknoten zu erstellen, der lediglich eine Klassifizierungsvorhersage macht. Alternativ können Sie sich auch dafür entscheiden, die Teilung zu stoppen, wenn die Teilung nicht mehr dazu führt, dass der Baum die maximale Tiefe überschreitet. Wobei die maximale Tiefe, die Sie dem Baum erlauben, ein Parameter ist, den Sie einfach sagen können. Im Entscheidungsbaum ist die Tiefe eines Knotens als die Anzahl der Sprünge definiert, die erforderlich sind, um vom Wurzelknoten, der ganz oben steht, zu diesem bestimmten Knoten zu gelangen. Der Wurzelknoten benötigt also null Sprünge, erhält sich selbst und befindet sich auf Tiefe 0. Die Notizen darunter befinden sich auf Tiefe eins und in den darunter liegenden Notizen wäre er auf Tiefe 2. Wenn Sie entschieden hätten, dass die maximale Tiefe des Entscheidungsbaums sagen wir mal Zweitens würden Sie sich dafür entscheiden, keine Knoten unterhalb dieser Ebene zu teilen, damit der Baum nie die Tiefe 3 erreicht. Ein Grund, warum Sie die Tiefe des Entscheidungsbaums möglicherweise begrenzen möchten, besteht darin, sicherzustellen, dass der Baum nicht zu tief wird groß und unhandlich, und zweitens ist der Baum dadurch weniger anfällig für Überanpassungen, wenn er klein gehalten wird. Ein weiteres Kriterium, anhand dessen Sie entscheiden könnten, die Aufteilung zu beenden, könnte darin bestehen, dass die Verbesserungen im Prioritätswert, die Sie in einem späteren Video sehen, unter einem bestimmten Schwellenwert liegen. Wenn die Aufteilung eines Knotens zu minimalen Verbesserungen der Reinheit führt oder Sie später feststellen, dass die Verunreinigung tatsächlich abnimmt. Aber wenn die Gewinne zu gering sind, stört es sie möglicherweise nicht. Auch hier gilt, sowohl die Bäume kleiner zu halten als auch das Risiko einer Überanpassung zu verringern. Wenn schließlich die Anzahl der Beispiele für einen Knoten unter einem bestimmten Schwellenwert liegt, können Sie sich auch dafür entscheiden, die Aufteilung zu beenden. Wenn wir beispielsweise am Wurzelknoten die Funktion „Gesichtsform“ geteilt haben, dann hatte der rechte Zweig nur drei Trainingsbeispiele mit einer Katze und zwei Hunden und anstatt diese in noch kleinere Teilmengen aufzuteilen, wenn Sie sich entschieden haben, nicht zu teilen Weitere Beispiele mit nur drei Ihrer Beispiele erstellen, dann erstellen Sie einfach einen Entscheidungsknoten, und da es hier möglicherweise Hunde zu anderen drei Erwachsenen gibt, wäre dies ein Knoten, der eine Vorhersage macht, dass es sich nicht um eine Katze handelt. Auch hier liegt ein Grund, warum Sie möglicherweise entscheiden, dass sich die Aufteilung nicht lohnt, darin, den Baum kleiner zu halten und eine Überanpassung zu vermeiden. Wenn ich mir selbst Lernaufgaben im Entscheidungsbaum ansehe, habe ich manchmal das Gefühl, Junge, in diesem Algorithmus passieren viele verschiedene Teile und viele verschiedene Dinge. Ein Grund dafür scheint möglicherweise in der Entwicklung von Entscheidungsbäumen zu liegen. Es gab einen Forscher, der eine Basisversion von Entscheidungsbäumen vorschlug, und dann sagte ein anderer Forscher: „Oh, wir können das Ding auf diese Weise modifizieren, wie zum Beispiel seine neuen Kriterien für die Aufteilung.“ Dann kommt ein anderer Forscher auf eine andere Idee, etwa: „Oh, vielleicht sollten wir aufhören zu schwitzen, wenn es eine bestimmte maximale Tiefe erreicht.“ Im Laufe der Jahre haben verschiedene Forscher unterschiedliche Verfeinerungen des Algorithmus entwickelt.

# Decision Tree Learning

**Decision 2:** When do you stop splitting?

- When a node is 100% one class
- When splitting a node will result in the tree exceeding a maximum depth
- When improvements in purity score are below a threshold
- When number of examples in a node is below a threshold



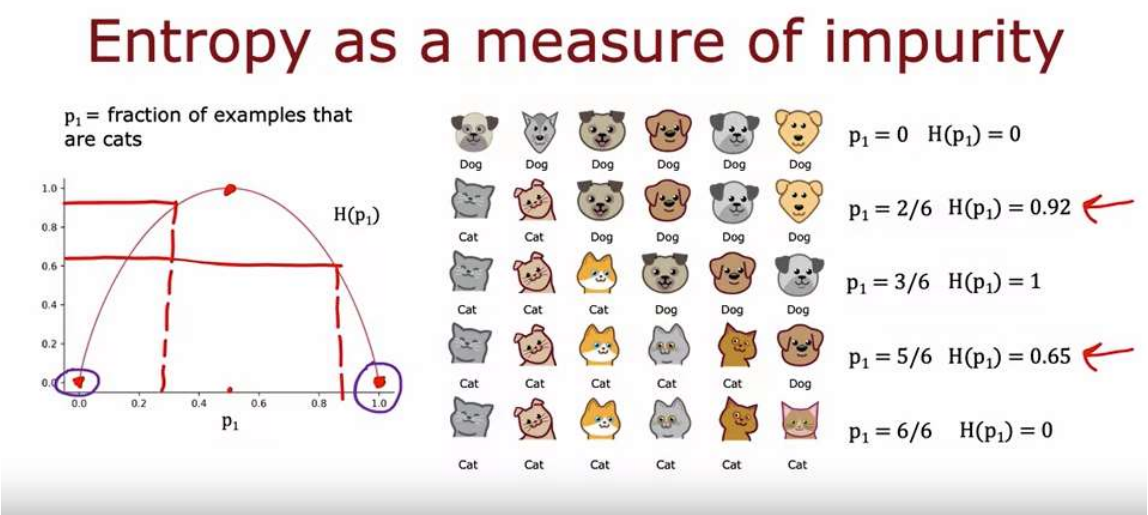

Infolgedessen funktioniert es wirklich gut, aber wir schauen uns alle Details an, wie eine Entscheidung umgesetzt werden kann. Wenn es sich für Sie wie ein etwas komplizierter, chaotischer Durchschnitt anfühlt, gilt das auch für mich. Aber diese verschiedenen Teile passen zu einem sehr effektiven Lernalgorithmus zusammen, und was Sie in diesem Kurs lernen, sind die wichtigsten Ideen, damit es gut funktioniert, und am Ende dieser Woche werde ich es auch tun Teilen Sie mit Ihnen einige Anleitungen und Vorschläge für die Verwendung von Open-Source-Paketen, damit Sie nicht zwei haben müssen, was das Verfahren zum Treffen all dieser Entscheidungen verkompliziert. Wie entscheide ich mich, mit dem Teilen aufzuhören? Man bringt diese Atome wirklich dazu, gut für sich selbst zu arbeiten. Aber ich möchte Ihnen versichern , dass dieser Algorithmus, auch wenn er kompliziert und chaotisch erscheint, mir ehrlich gesagt auch so vorkommt, aber er funktioniert gut. Die nächste wichtige Entscheidung, auf die ich tiefer eingehen möchte, ist die Frage, wie Sie entscheiden, wie ein Knoten aufgeteilt wird. Schauen wir uns im nächsten Video diese Definition der Entropie an, die uns eine Möglichkeit bieten würde, die Reinheit, oder genauer gesagt, die Unreinheit in einem Knoten zu messen. Kommen wir zum nächsten Video.

## Reinheit messen

In diesem Video schauen wir uns an, wie man die Reinheit einer Reihe von Beispielen misst. Wenn es sich bei allen Beispielen um Katzen einer einzelnen Klasse handelt, ist das sehr rein, wenn es sich bei allen um keine Katzen handelt, ist das auch sehr rein, aber wenn es irgendwo dazwischen liegt, wie kann man quantifizieren, wie rein die Menge der Beispiele ist? Werfen wir einen Blick auf die Definition der Entropie, die ein Maß für die Verunreinigung eines Datensatzes ist. Bei einem Satz von sechs Beispielen wie diesem haben wir drei Katzen und drei Hunde. Definieren wir p_1 als den Anteil der Beispiele, die Katzen sind, also den Anteil der Beispiele mit der Bezeichnung eins, das ist es, was der Index eins anzeigt. p_1 ist in diesem Beispiel gleich 3/6. Wir werden die Verunreinigung einer Reihe von Beispielen mithilfe einer Funktion namens Entropie messen, die so aussieht. Die Entropiefunktion wird üblicherweise als Großbuchstaben H dieser Zahl p_1 bezeichnet und die Funktion sieht wie diese Kurve hier aus, wobei die horizontale Achse p_1, der Anteil der Katzen in der Stichprobe, und die vertikale Achse der Wert der Entropie ist. In diesem Beispiel, in dem p_1 3/6 oder 0,5 beträgt, wäre der Wert der Entropie von p_1 gleich eins. Sie bemerken, dass diese Kurve am höchsten ist, wenn Ihre Beispielmenge 50-50 beträgt. Sie ist also am unreinsten als eine Verunreinigung von eins oder mit einer Entropie von eins, wenn Ihre Beispielmenge 50-50 beträgt, wohingegen sie im Gegensatz dazu am unreinsten ist, wenn Ihre Beispielmenge beträgt Waren entweder alle Katzen oder keine Katzen, dann ist die Entropie Null. Lassen Sie uns einfach noch ein

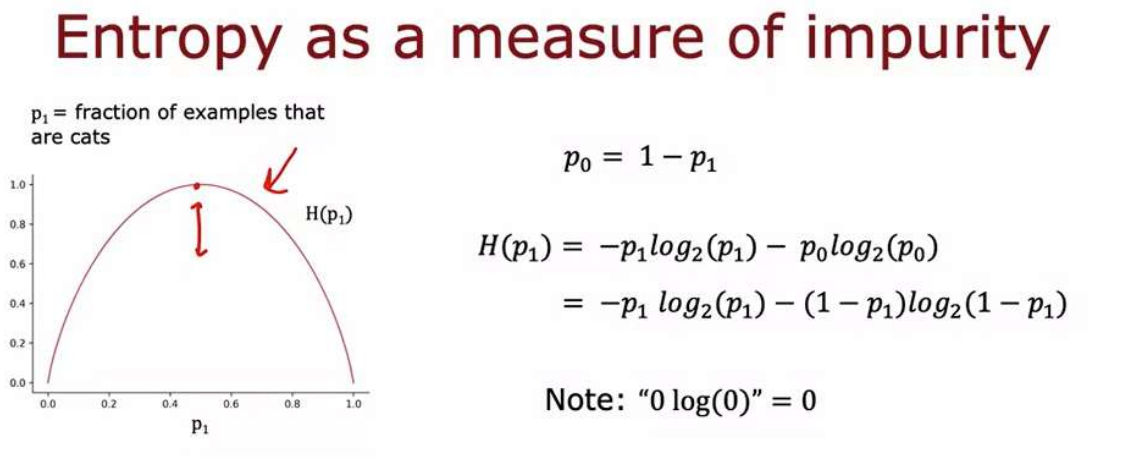
paar Beispiele durchgehen, um einen besseren Einblick in die Entropie und ihre Funktionsweise zu gewinnen. Hier ist ein anderer Satz von Beispielen mit fünf Katzen und einem Hund, also ist p_1 der Bruchteil der positiven Beispiele, ein Bruchteil der mit einem gekennzeichneten Beispiele ist 5/6 und p_1 ist also etwa 0,83. Wenn Sie diesen Wert bei etwa 0,83 ablesen, stellen wir fest, dass die Entropie von p_1 etwa 0,65 beträgt. Und hier schreibe ich es nur auf zwei signifikante Ziffern. Hier ist noch ein Beispiel. Diese Stichprobe von sechs Bildern hat alle Katzen, also ist p_1 sechs von sechs, weil alle sechs Katzen sind und die Entropie von p_1 dieser Punkt hier ist, der Null ist. Wir sehen, dass, wenn man von 3/6 auf sechs von sechs Katzen übergeht, die Verunreinigung von eins auf null abnimmt, oder mit anderen Worten, die Reinheit zunimmt, wenn man von einer 50:50-Mischung aus Katzen und Hunden auf alle Katzen übergeht. Schauen wir uns noch ein paar Beispiele an.

---




Hier ist eine weitere Stichprobe mit zwei Katzen und vier Hunden, also ist p_1 hier 2/6, also 1/3, und wenn man die Entropie bei 0,33 abliest, ergibt sich ein Wert von etwa 0,92. Das ist tatsächlich ziemlich unrein und insbesondere ist dieses Set unreiner als dieses Set, weil es eher einer 50:50-Mischung entspricht, weshalb die Verunreinigung hier 0,92 im Gegensatz zu 0,65 beträgt. Zum Schluss noch ein letztes Beispiel: Wenn wir eine Menge aller sechs Hunde haben, dann ist p_1 gleich 0 und die Entropie von p_1 ist genau diese Zahl hier unten, die gleich 0 ist, also gibt es keine Verunreinigung, sonst wäre dies eine völlig reine Menge alle nicht Katzen oder alle Hunde. Schauen wir uns nun die tatsächliche Gleichung für die Entropiefunktion H( p_1) an. Denken Sie daran, dass p_1 der Bruchteil der Beispiele ist, die Katzen entsprechen. Wenn Sie also eine Stichprobe haben, die 2/3 Katzen enthält, muss diese Stichprobe 1/3 Nicht-Katzen enthalten. Lassen Sie mich p_0 als gleich dem Bruchteil der Beispiele, die keine Katzen sind, als gerade gleich 1 minus p_1 definieren. Die Entropiefunktion ist dann als negativ p_1log_2 (p_1) definiert, und bei der Berechnung der Entropie nehmen wir gemäß Konvention Logarithmen zur Basis zwei statt zur Basis e und dann minus p_0log_2( p_0 ) . Alternativ ist dies auch gleich negativ p_1log_ 2( p_1) minus 1 minus p_1 log_2(1 minus p_1). Wenn Sie diese Funktion in einem Computer grafisch darstellen würden, werden Sie feststellen, dass es genau diese Funktion auf der linken Seite ist. Wir nehmen log_2, nur um die Spitze dieser Kurve gleich eins zu machen. Wenn wir log_e oder die Basis natürlicher Logarithmen nehmen würden, dann skaliert das diese Funktion nur vertikal, und es wird immer noch funktionieren, aber die Zahlen sind etwas schwer zu interpretieren weil der Spitzenwert der Funktion keine schöne runde Zahl mehr ist. Ein Hinweis zur Berechnung dieser Funktion: Wenn p_1 oder p_0 gleich 0 ist, sieht ein Ausdruck wie dieser wie 0log( 0) aus, und log(0) ist technisch gesehen undefiniert, es ist eigentlich eine negative Unendlichkeit. Für die Zwecke der Berechnung der Entropie gehen wir jedoch vereinbarungsgemäß davon aus, dass 0log( 0) gleich 0 ist, und das wird die

Entropie korrekt als Null oder als Eins als gleich Null berechnen. Wenn Sie denken, dass diese Definition der Entropie ein wenig der Definition des logistischen Verlusts ähnelt, die wir im letzten Kurs kennengelernt haben, gibt es tatsächlich eine mathematische Begründung dafür, warum diese beiden Formeln so ähnlich aussehen. Aber Sie müssen sich darüber keine Sorgen machen und wir werden in diesem Kurs nicht darauf eingehen.

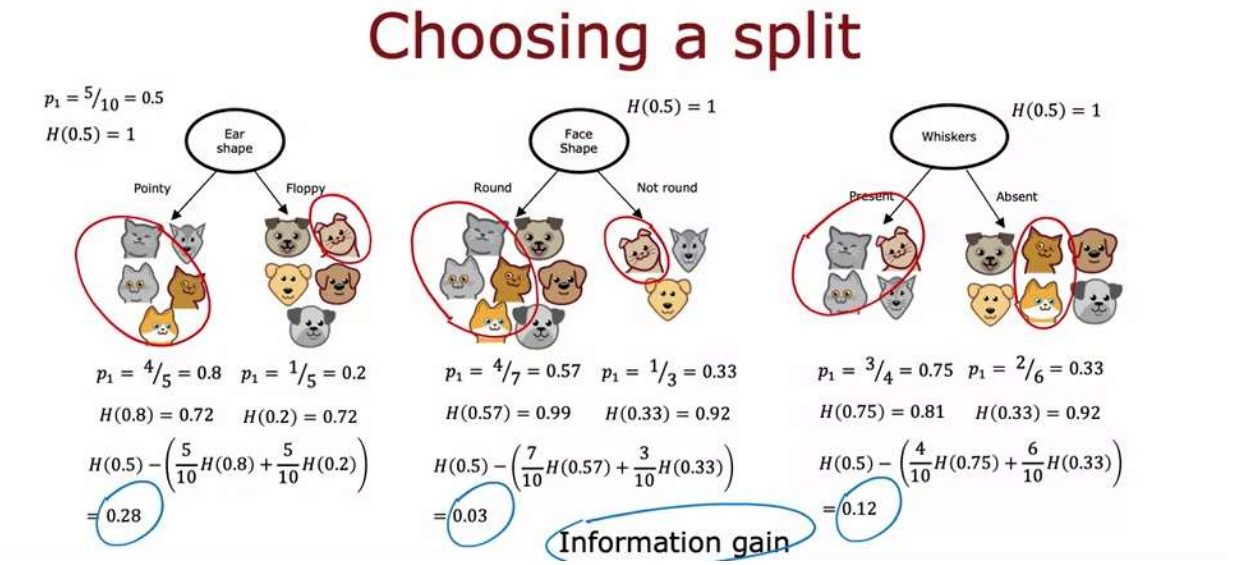



Die Anwendung dieser Formel für die Entropie sollte jedoch problemlos funktionieren, wenn Sie einen Entscheidungsbaum erstellen. Zusammenfassend ist die Entropiefunktion ein Maß für die Verunreinigung eines Datensatzes. Sie beginnt bei Null, steigt auf Eins und sinkt dann als Funktion des Anteils positiver Beispiele in Ihrer Stichprobe wieder auf Null. Es gibt andere Funktionen, die so aussehen: Sie gehen von Null auf Eins und dann wieder zurück. Wenn Sie beispielsweise in Open-Source-Paketen nachsehen, hören Sie möglicherweise auch von den sogenannten Gini-Kriterien, einer weiteren Funktion, die der Entropiefunktion sehr ähnlich ist und sich auch gut zum Erstellen von Entscheidungsbäumen eignet. Aber der Einfachheit halber werde ich mich in diesen Videos auf die Verwendung der Entropiekriterien konzentrieren, die normalerweise für die meisten Anwendungen gut funktionieren. Nachdem wir nun diese Definition der Entropie haben, werfen wir im nächsten Video einen Blick darauf, wie Sie sie tatsächlich verwenden können, um Entscheidungen darüber zu treffen, welche Funktion in den Knoten eines Entscheidungsbaums aufgeteilt werden soll.

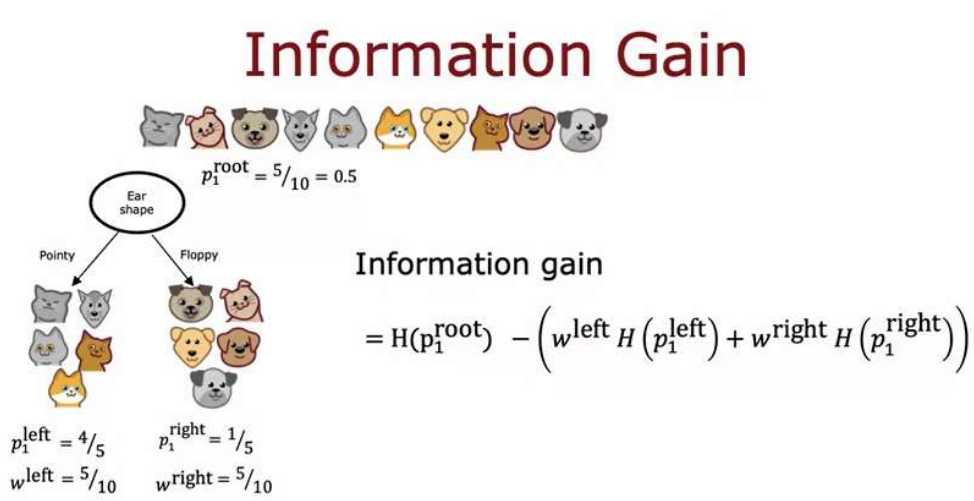
## Eine Aufteilung wählen: Informationsgewinn

Beim Erstellen eines Entscheidungsbaums basiert die Art und Weise, wie wir entscheiden, welches Feature an einem Knoten aufgeteilt werden soll, darauf, welche Feature-Auswahl die Entropie am meisten reduziert. Reduziert die Entropie oder reduziert Verunreinigungen oder maximiert die Reinheit. Beim Lernen im Entscheidungsbaum wird die Verringerung der Entropie als Informationsgewinn bezeichnet. Schauen wir uns in diesem Video an, wie man den Informationsgewinn berechnet und daher auswählt, welche Features für die Aufteilung an jedem Knoten in einem Entscheidungsbaum verwendet werden sollen. Nehmen wir als Beispiel die Entscheidung, welche Funktion am Wurzelknoten des Entscheidungsbaums verwendet werden soll, den wir gerade erstellt haben, um Katzen gegenüber Nicht-Katzen zu erkennen. Wenn wir anhand der Ohrformfunktion am Wurzelknoten geteilt hätten, hätten wir Folgendes erhalten: fünf Beispiele links und fünf rechts. Auf der linken Seite hätten wir vier von fünf Katzen, also wäre P1 gleich 4/5 oder 0,8. Auf der rechten Seite ist eine von fünf Katzen, also ist P1 gleich 1/5 oder 0,2. Wenn Sie die

Entropieformel aus dem letzten Video auf diese linke Datenteilmenge und diese rechte Datenteilmenge anwenden, stellen wir fest, dass der Verunreinigungsgrad auf der linken Seite eine Entropie von 0,8 ist, was etwa 0,72 entspricht, und auf der rechten Seite die Entropie von 0,2 erweist sich auch als 0,72. Dies wäre die Entropie am linken und rechten Unterzweig, wenn wir das Ohrformmerkmal aufteilen würden. Eine andere Möglichkeit wäre die Aufteilung nach Gesichtsformmerkmalen. Wenn wir das gemacht hätten, wären auf der linken Seite vier der sieben Beispiele Katzen, also ist P1 4/7, und auf der rechten Seite sind 1/3 Katzen, also ist P1 auf der rechten Seite 1/3. Die Entropie von 4/7 und die Entropie von 1/3 betragen 0,99 und 0,92. Daher scheint der Grad der Verunreinigung im linken und rechten Knoten viel höher zu sein, nämlich 0,99 und 0,92 im Vergleich zu 0,72 und 0,72. Schließlich wäre die dritte mögliche Auswahl an Features, die am Wurzelknoten verwendet werden sollen, die Whiskers-Funktion. In diesem Fall unterteilen Sie basierend darauf, ob Whiskers vorhanden sind oder nicht. In diesem Fall beträgt P1 links 3/4, P1 rechts 2/6 und die Entropiewerte sind wie folgt. Die Schlüsselfrage, die wir beantworten müssen, ist angesichts dieser drei Optionen einer Funktion, die am Stammknoten verwendet werden soll: Welche funktioniert unserer Meinung nach am besten? Es stellt sich heraus, dass es sinnvoller wäre, einen gewichteten Durchschnitt daraus zu ermitteln, anstatt diese Entropiezahlen zu betrachten und zu vergleichen, und hier ist, was ich meine. Wenn es einen Knoten mit vielen Beispielen und hoher Entropie gibt, scheint das schlimmer zu sein, als wenn es einen Knoten mit nur wenigen Beispielen und hoher Entropie gäbe. Denn die Entropie als Maß für die Verunreinigung ist bei einem sehr großen und unreinen Datensatz schlechter als bei nur wenigen Beispielen und einem Zweig des Baums, der sehr unrein ist. Die wichtigste Entscheidung besteht darin, welche dieser drei möglichen Funktionen zur Verwendung am Wurzelknoten wir verwenden möchten. Mit jeder dieser Aufteilungen sind zwei Zahlen verbunden, die Entropie auf dem linken Unterzweig und die Entropie auf dem rechten Unterzweig. Um daraus eine Auswahl zu treffen, kombinieren wir diese beiden Zahlen gerne zu einer einzigen Zahl. Sie können also einfach eine dieser drei Möglichkeiten auswählen. Welche davon ist am besten? Wir werden diese beiden Zahlen kombinieren, indem wir einen gewichteten Durchschnitt bilden. Denn wie wichtig es ist, beispielsweise im linken oder rechten Unterzweig eine niedrige Entropie zu haben, hängt auch davon ab, wie viele Beispiele in den linken oder rechten Unterzweig gelangt sind. Denn wenn es beispielsweise im linken Unterzweig viele Beispiele gibt, scheint es wichtiger zu sein, sicherzustellen, dass der Entropiewert dieses linken Unterzweigs niedrig ist. In diesem Beispiel gingen fünf der zehn Beispiele in den linken Unterzweig, sodass wir den gewichteten Durchschnitt als 5/10-fache Entropie von 0,8 berechnen und dann hinzufügen können, dass 5/10 Beispiele ebenfalls in den rechten Unterzweig gingen Unterzweig, plus 5/10-fache Entropie von 0,2. Für dieses Beispiel in der Mitte hatte der linke Unterzweig nun sieben von zehn Beispielen erhalten. und so werden wir das 7/10-fache der Entropie von 0,57 plus berechnen, der rechte Unterzweig hatte drei von 10 Beispielen, also plus 3/10-fache Entropie von 0,3 von 1/3. Schließlich berechnen wir rechts die 4/10-fache Entropie von 0,75 plus die 6/10-fache Entropie von 0,33.



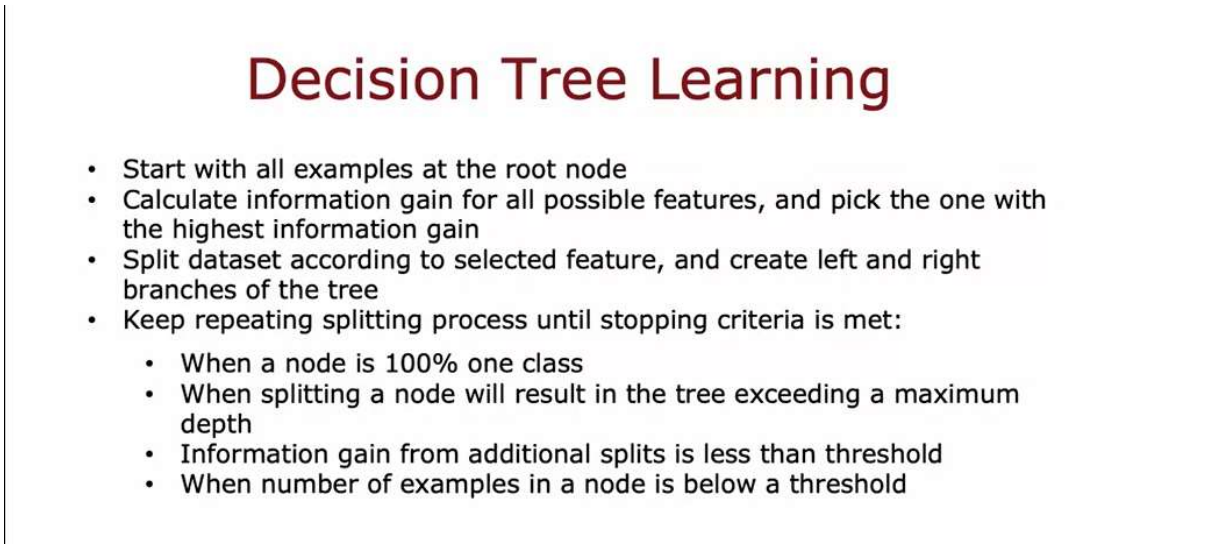

Die Art und Weise, wie wir eine Aufteilung wählen, besteht darin, diese drei Zahlen zu berechnen und die niedrigste auszuwählen, da wir dadurch den linken und rechten Unterzweig mit der niedrigsten durchschnittlich gewichteten Entropie erhalten. An der Art und Weise, wie Entscheidungsbäume erstellt werden, werden wir tatsächlich eine weitere Änderung an diesen Formeln vornehmen, um die Konventionen beim Erstellen von Entscheidungsbäumen einzuhalten, aber das wird das Ergebnis nicht wirklich ändern. Anstatt diese gewichtete durchschnittliche Entropie zu berechnen, berechnen wir die Verringerung der Entropie im Vergleich dazu, wenn wir überhaupt nicht gespalten hätten. Wenn wir zum Wurzelknoten gehen, denken Sie daran, dass wir mit allen 10 Beispielen im Wurzelknoten mit fünf Katzen und Hunden begonnen haben und dass wir am Wurzelknoten p_1 gleich 5/10 oder 0,5 hatten. Die Entropie der Wurzelknoten, eine Entropie von 0,5, war tatsächlich gleich 1. Dies war die höchste Reinheit, da es sich um fünf Katzen und fünf Hunde handelte. Die Formel, die wir tatsächlich für die Auswahl einer Aufteilung verwenden werden, ist nicht die Entropie am linken und rechten Unterzweig, sondern die Entropie am Wurzelknoten, also eine Entropie von 0,5, dann minus dieser Formel. Wenn Sie in diesem Beispiel nachrechnen, ergibt sich ein Wert von 0,28. Für das Beispiel der Gesichtsform können wir die Entropie des Wurzelknotens berechnen, eine Entropie von 0,5 minus dieser, was sich als 0,03 herausstellt, und für Whiskers, die sich als 0,12 herausstellt. Diese Zahlen, die wir gerade berechnet haben, 0,28, 0,03 und 0,12, werden als Informationsgewinn bezeichnet und messen die Verringerung der Entropie, die Sie in Ihrem Baum aufgrund einer Teilung erhalten. Da die Entropie am Wurzelknoten ursprünglich eins war und Sie die Aufteilung vornehmen, erhalten Sie am Ende einen niedrigeren Entropiewert und die Differenz zwischen diesen beiden Werten ist eine Verringerung der Entropie, und das sind 0,28 im Fall der Aufteilung an der Ohrform . Warum machen wir uns die Mühe, die Entropiereduzierung zu berechnen und nicht nur die Entropie am linken und rechten Unterzweig? Es stellt sich heraus, dass eines der Stoppkriterien für die Entscheidung, wann man sich nicht weiter spalten sollte, darin besteht, dass die Entropiereduzierung zu gering ist. In diesem Fall könnten Sie entscheiden, dass Sie den Baum nur unnötig vergrößern und eine Überanpassung durch Aufteilung riskieren, und sich einfach nicht darum kümmern, wenn die Entropiereduzierung zu gering ist oder unter einem Schwellenwert liegt.




In diesem anderen Beispiel führt das Spucken auf die Ohrform zur größten Verringerung der Entropie. 0,28 ist größer als 0,03 oder 0,12, und daher würden wir uns für die Aufteilung auf das Ohrformmerkmal am Wurzelknoten entscheiden. Auf der nächsten Folie geben wir eine formellere Definition des Informationsgewinns. Eine weitere Notation, die wir übrigens auch auf der nächsten Folie vorstellen werden, sind diese Zahlen: 5/10 und 5/10. Ich werde dies w^left nennen, weil das der Bruchteil der Beispiele ist, die in den linken Zweig gegangen sind, und ich werde es w^right nennen, weil das der Bruchteil der Beispiele ist, die in den rechten Zweig gegangen sind. Für dieses andere Beispiel wäre w^left 7/10 und w^right 3/10. Schreiben wir nun die allgemeine Formel zur Berechnung des Informationsgewinns auf. Lassen Sie mich anhand des Beispiels der Aufteilung des Ohrformmerkmals p_1^left als gleich dem Bruchteil der Beispiele im linken Teilbaum definieren, die eine positive Bezeichnung haben, also Katzen. In diesem Beispiel ist p_1^left gleich 4/5. Lassen Sie mich außerdem w^left als den Bruchteil der Beispiele aller Beispiele des Wurzelknotens definieren, die zum linken Unterzweig gingen. In diesem Beispiel wäre w^left also 5/10. In ähnlicher Weise definieren wir p_1^right als eines aller Beispiele im rechten Zweig. Der Anteil, bei dem es sich um positive Beispiele handelt und bei dem es sich bei einem dieser fünf Beispiele um Katzen handelt, beträgt 1/5, und in ähnlicher Weise ist w^right 5/10 der Anteil der Beispiele, die in den rechten Unterzweig gelangten. Definieren wir außerdem p_1^root als den Anteil der Beispiele, die im Wurzelknoten positiv sind. In diesem Fall wäre dies 5/10 oder 0,5. Der Informationsgewinn wird dann als die Entropie von p_1^root definiert, also wie hoch ist die Entropie am Wurzelknoten, abzüglich der gewichteten Entropieberechnung, die wir auf der vorherigen Folie hatten, minus w^left, das waren im Beispiel 5/10 mal das Entropie angewendet auf p_1^left, das ist die Entropie auf dem linken Unterzweig, plus w^right der Anteil der Beispiele, die zum rechten Zweig gingen, multipliziert mit der Entropie von p_1^right. Mit dieser Definition der Entropie können Sie den Informationsgewinn berechnen, der mit der Auswahl eines bestimmten Features zur Aufteilung im Knoten verbunden ist. Dann können Sie aus allen möglichen Zukünften, die Sie aufteilen möchten, diejenige auswählen, die Ihnen den höchsten Informationsgewinn bringt. Dies führt hoffentlich zu einer Erhöhung der Reinheit Ihrer Datenteilmengen, die Sie in den linken und rechten Unterzweigen Ihres Entscheidungsbaums erhalten, und zur Auswahl einer Funktion zur Aufteilung, die die Reinheit Ihrer Datenteilmengen erhöht sowohl im linken als auch im rechten Unterzweig Ihres Entscheidungsbaums. Nachdem Sie nun wissen, wie Sie den Informationsgewinn oder die Entropiereduzierung berechnen, wissen Sie, wie Sie ein Feature auswählen, das auf einem anderen Knoten aufgeteilt werden soll. Lassen Sie uns alle Dinge, über die wir gesprochen haben, in den Gesamtalgorithmus zum Aufbau eines Entscheidungsbaums unter Berücksichtigung eines Trainingssatzes zusammenfassen. Schauen wir uns das im nächsten Video an.

## Etwas zusammensetzen

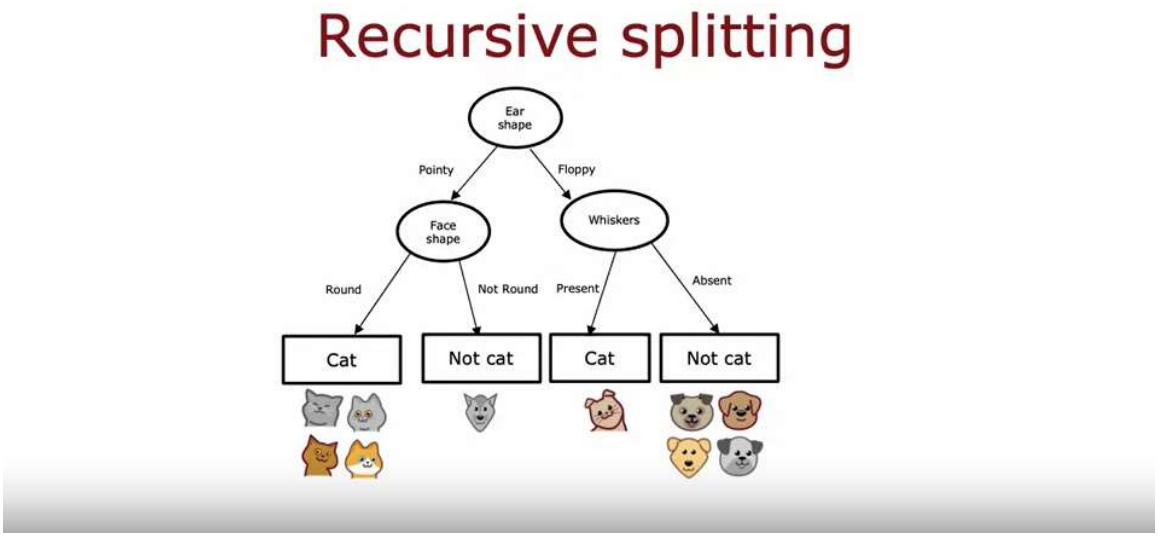
Anhand der Kriterien für den Informationsgewinn können Sie entscheiden, wie Sie ein Feature auswählen, um einen einzelnen Knoten aufzuteilen. Nehmen wir das und nutzen es an mehreren Stellen in einem Entscheidungsbaum, um herauszufinden, wie man einen großen Entscheidungsbaum mit mehreren Knoten erstellt. Hier ist der Gesamtprozess zum Erstellen eines Entscheidungsbaums. Beginnen Sie mit allen Trainingsbeispielen am Wurzelknoten des Baums, berechnen Sie den Informationsgewinn für alle möglichen Features und wählen Sie das Feature zur Aufteilung aus, das den höchsten Informationsgewinn bietet. Nachdem Sie diese Funktion ausgewählt haben, teilen Sie den Datensatz entsprechend der ausgewählten Funktion in zwei Teilmengen auf, erstellen linke und rechte Zweige des Baums und senden die Trainingsbeispiele je nach Wert dieser Funktion entweder an den linken oder den rechten Zweig für dieses Beispiel. Dadurch können Sie eine Aufteilung am Wurzelknoten vorgenommen haben. Danach wiederholen Sie den Teilungsvorgang immer wieder am linken Ast des Baumes, am rechten Ast des Baumes usw. Machen Sie damit weiter, bis das Stoppkriterium erfüllt ist. Die Stoppkriterien können sein: Wenn ein Knoten zu 100 Prozent aus einer einzelnen Klausel besteht, hat jemand die Entropie Null erreicht, oder wenn eine weitere Aufteilung eines Knotens dazu führt, dass der Baum die von Ihnen festgelegte maximale Tiefe überschreitet, oder wenn der Informationsgewinn aus Eine zusätzliche Aufteilung liegt unter dem Schwellenwert oder die Anzahl der Beispiele in einem Knoten liegt unter einem Schwellenwert.




Sie wiederholen den Aufteilungsvorgang so lange, bis das von Ihnen gewählte Stoppkriterium, bei dem es sich um eines oder mehrere dieser Kriterien handeln kann, erfüllt ist. Schauen wir uns eine Illustration an, wie dieser Prozess funktionieren wird. Wir haben alle Beispiele an den Wurzelknoten begonnen und auf der Grundlage der Berechnung des Informationsgewinns für alle drei Features entschieden, dass die Ohrform das beste Feature für die Aufteilung ist. Auf dieser Grundlage erstellen wir einen linken und einen rechten Unterzweig und senden die Teilmengen der Daten mit spitzem oder schlaffem Ohr an den linken und rechten Unterzweig. Lassen Sie mich den Wurzelknoten und den rechten Unterzweig behandeln und mich nur auf den linken Unterzweig konzentrieren, wo wir diese fünf Beispiele haben.

Sehen wir uns die Aufteilungskriterien an: So lange aufteilen, bis alles im Knoten zu einer einzigen Klasse gehört, also entweder alle Knoten aufheben. Wir werden uns diesen Knoten ansehen und prüfen, ob er die Aufteilungskriterien erfüllt. Dies ist jedoch nicht der Fall, da es hier eine Mischung

aus Katzen und Hunden gibt. Der nächste Schritt besteht darin, ein Feature auszuwählen, nach dem geteilt werden soll. Anschließend gehen wir die Features einzeln durch und berechnen den Informationsgewinn jedes dieser Features, als ob dieser Knoten der neue Wurzelknoten eines Entscheidungsbaums wäre, der anhand von nur fünf hier gezeigten Trainingsbeispielen trainiert wurde. Wir würden den Informationsgewinn für die Aufteilung beim Whisker-Feature und den Informationsgewinn bei der Aufteilung beim V-Shape-Feature berechnen. Es stellt sich heraus, dass der Informationsgewinn für die Aufteilung nach Ohrform Null ist, da alle diese Punkte die gleiche Ohrform haben. Zwischen Whiskern und V-Form weist die V-Form den höchsten Informationsgewinn auf. Wir werden die V-Form aufteilen und das ermöglicht uns, wie folgt linke und rechte Unterzweige zu erstellen. Für den linken Unterzweig prüfen wir die Kriterien dafür, ob wir mit der Aufteilung aufhören sollen oder nicht, und wir haben hier alle Katzen. Das Stoppkriterium ist erfüllt und wir erstellen einen Blattknoten, der eine Vorhersage der Katze macht.




Für den rechten Unterzweig stellen wir fest, dass es sich um alle Hunde handelt. Wir werden auch mit der Aufteilung aufhören, da wir die Aufteilungskriterien erfüllt haben, und dort einen Blattknoten platzieren, der keine Katze vorhersagt. Nachdem wir diesen linken Teilbaum erstellt haben, können wir uns nun dem Aufbau des rechten Teilbaums zuwenden. Lassen Sie mich nun noch einmal den Wurzelknoten und den gesamten linken Teilbaum verdecken. Um den richtigen Teilbaum aufzubauen, haben wir hier diese fünf Beispiele. Auch hier prüfen wir als Erstes, ob die Kriterien zum Stoppen der Aufteilung erfüllt sind, ob ihre Kriterien erfüllt sind oder nicht. Bei allen Beispielen handelt es sich um eine einzelne Klausel, wir haben dieses Kriterium nicht erfüllt. Wir werden uns dafür entscheiden, auch in diesem rechten Unterzweig weiter aufzuteilen. Tatsächlich wird das Verfahren zum Erstellen des richtigen Unterzweigs weitgehend so aussehen, als würden Sie einen Entscheidungsbaum-Lernalgorithmus von Grund auf trainieren, wobei der Datensatz, über den Sie verfügen, nur aus diesen fünf Trainingsbeispielen besteht. Auch hier stellt man bei der Berechnung des Informationsgewinns für alle möglichen Features zur Aufteilung fest, dass das Whiskers-Feature den höchsten Informationsgewinn nutzt. Teilen Sie diesen Satz aus fünf Beispielen danach auf, ob Whiskers vorhanden sind oder nicht. Überprüfen Sie hier, ob die Kriterien zum Stoppen der Aufteilung in den linken und rechten Unterzweigen erfüllt sind, und entscheiden Sie, dass dies der Fall ist. Am Ende erhält man Blattknoten, die Katze und Hund Katze vorhersagen. Dies ist der Gesamtprozess zum Aufbau des Entscheidungsbaums. Beachten Sie, dass es interessante Aspekte unserer Arbeit gibt. Nachdem wir uns entschieden hatten, was am Wurzelknoten aufgeteilt werden soll, haben wir den linken Teilbaum dadurch erstellt, dass wir einen Entscheidungsbaum auf einer Teilmenge von fünf Beispielen erstellt haben. Die Art und Weise, wie wir den richtigen Teilbaum

erstellt haben, bestand wiederum darin, einen Entscheidungsbaum anhand einer Teilmenge von fünf Beispielen zu erstellen. In der Informatik ist dies ein Beispiel für einen rekursiven Algorithmus. Das bedeutet lediglich, dass Sie einen Entscheidungsbaum an der Wurzel aufbauen, indem Sie weitere kleinere Entscheidungsbäume in den linken und rechten Unterzweigen erstellen. Unter Rekursion versteht man in der Informatik das Schreiben von Code, der sich selbst aufruft. Die Art und Weise, wie dies beim Erstellen eines Entscheidungsbaums geschieht, besteht darin, dass Sie den gesamten Entscheidungsbaum erstellen, indem Sie kleinere Unterentscheidungsbäume erstellen und diese dann alle zusammenfügen. Wenn Sie sich daher Softwareimplementierungen von Entscheidungsbäumen ansehen, werden Sie manchmal Hinweise auf einen rekursiven Algorithmus finden. Aber wenn Sie das Gefühl haben, dieses Konzept rekursiver Algorithmen nicht vollständig verstanden zu haben, machen Sie sich darüber keine Sorgen. Sie können die Aufgaben dieser Woche immer noch vollständig erledigen und Bibliotheken nutzen, um Entscheidungsbäume für sich selbst zum Laufen zu bringen. Wenn Sie jedoch einen Entscheidungsbaumalgorithmus von Grund auf implementieren, stellt sich heraus, dass ein rekursiver Algorithmus einer der Schritte ist, die Sie implementieren müssen. Übrigens fragen Sie sich vielleicht, wie Sie den Parameter für die maximale Tiefe auswählen. Es gibt viele verschiedene Auswahlmöglichkeiten, aber einige der Open-Source-Bibliotheken verfügen über gute Standardoptionen, die Sie verwenden können. Eine Intuition besagt, dass der Entscheidungsbaum, den Sie erstellen möchten, umso größer ist, je größer die maximale Tiefe ist. Dies ähnelt ein wenig der Anpassung eines Polynoms höheren Grades oder dem Training eines größeren neuronalen Netzwerks. Dadurch kann der Entscheidungsbaum ein komplexeres Modell lernen, aber es erhöht auch das Risiko einer Überanpassung, wenn dadurch eine sehr komplexe Funktion an Ihre Daten angepasst wird. Theoretisch könnten Sie die Kreuzvalidierung verwenden, um Parameter wie die maximale Tiefe auszuwählen, indem Sie verschiedene Werte der maximalen Tiefe ausprobieren und auswählen, was im Kreuzvalidierungssatz am besten funktioniert. Allerdings haben die Open-Source-Bibliotheken in der Praxis noch etwas bessere Möglichkeiten, diesen Parameter für Sie auszuwählen. Ein weiteres Kriterium, anhand dessen Sie entscheiden können, wann die Aufteilung beendet werden soll, ist, ob die durch einen zusätzlichen Schlitz gewonnenen Informationen unter einem bestimmten Schwellenwert liegen. Wenn eine Funktion eingeschränkt ist und nur eine geringe Entropiereduzierung oder einen sehr geringen Informationsgewinn erzielt, können Sie sich auch dafür entscheiden, sich nicht darum zu kümmern. Schließlich können Sie auch entscheiden, mit der Aufteilung zu beginnen, wenn die Anzahl der Beispiele im Knoten unter einem bestimmten Schwellenwert liegt. Das ist der Prozess der Erstellung eines Entscheidungsbaums. Nachdem Sie nun den Entscheidungsbaum kennengelernt haben, können Sie, wenn Sie eine Vorhersage treffen möchten, dem Verfahren folgen, das Sie im allerersten Video dieser Woche gesehen haben, wo Sie ein neues Beispiel, beispielsweise ein Testbeispiel, nehmen und beginnen Legen Sie eine Route fest und folgen Sie den Entscheidungen weiter, bis Sie zur Blattnotiz gelangen, die dann die Vorhersage trifft. Nachdem Sie nun den grundlegenden Entscheidungsbaum-Lernalgorithmus kennen, möchte ich in den nächsten Videos auf einige weitere Verfeinerungen dieses Algorithmus eingehen. Bisher haben wir Funktionen nur verwendet, um zwei mögliche Werte anzunehmen. Aber manchmal gibt es eine Funktion, die kategoriale oder diskrete Werte annimmt, vielleicht aber auch mehr als zwei Werte. Schauen wir uns im nächsten Video an, wie man mit diesem Fall umgeht.

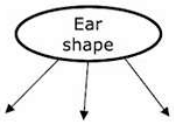
## Verwendung der One-Hot-Codierung kategorialer Merkmale

In dem Beispiel, das wir bisher gesehen haben, konnte jedes der Features nur einen von zwei möglichen Werten annehmen. Die Ohrform war entweder spitz oder schlaff, die Gesichtsform war entweder rund oder nicht rund und Schnurrhaare waren entweder vorhanden oder nicht vorhanden.

Wenn Sie jedoch über Funktionen verfügen, die mehr als zwei diskrete Werte annehmen können, sehen wir uns in diesem Video an, wie Sie solche Funktionen mithilfe der One-Hot-Codierung ansprechen können. Hier ist ein neues Trainingsset für unsere Haustieradoptionszentrumsanwendung, bei dem alle Daten bis auf die ohrförmige Funktion gleich sind. Statt nur spitz und schlaff zu sein, kann die Ohrform nun auch eine ovale Form annehmen.

## Features with three possible values

| Ear shape ($x_1$) | Face shape ($x_2$) | Whiskers ($x_3$) | Cat ($y$) |
|---|---|---|---|
| Pointy | Round | Present | 1 |
| Oval | Not round | Present | 1 |
| Oval | Round | Absent | 0 |
| Pointy | Not round | Present | 0 |
| Oval | Round | Present | 1 |
| Pointy | Round | Absent | 1 |
| Floppy | Not round | Absent | 0 |
| Oval | Round | Absent | 1 |
| Floppy | Round | Absent | 0 |
| Floppy | Round | Absent | 0 |




3 possible values

Daher ist das Anfangsmerkmal immer noch ein kategoriales Wertmerkmal, kann jedoch drei mögliche Werte anstelle von nur zwei möglichen Werten annehmen. Und das bedeutet, dass Sie bei der Aufteilung dieser Funktion am Ende drei Teilmengen der Daten und drei Unterzweige für diesen Baum erstellen. In diesem Video möchte ich jedoch eine andere Art der Adressierung von Funktionen beschreiben, die mehr als zwei Werte annehmen können, nämlich die Verwendung der One-Hot-Kodierung.

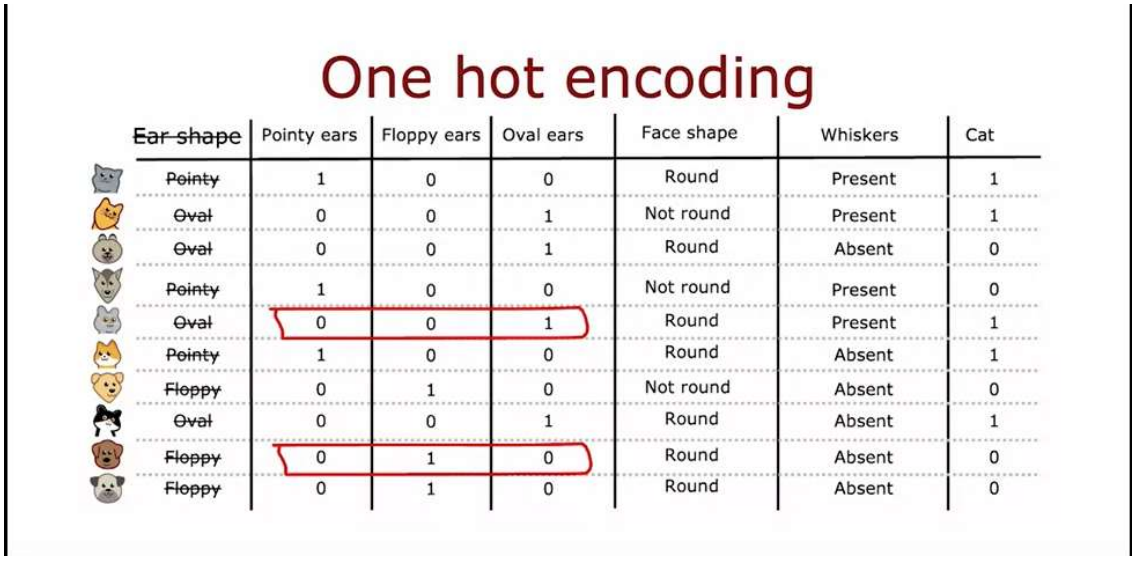
## One hot encoding

| ~~Ear shape~~ | Pointy ears | Floppy ears | Oval ears | Face shape | Whiskers | Cat |
|---|---|---|---|---|---|---|
| ~~Pointy~~ | 1 | 0 | 0 | Round | Present | 1 |
| ~~Oval~~ | 0 | 0 | 1 | Not round | Present | 1 |
| ~~Oval~~ | 0 | 0 | 1 | Round | Absent | 0 |
| ~~Pointy~~ | 1 | 0 | 0 | Not round | Present | 0 |
| ~~Oval~~ | 0 | 0 | 1 | Round | Present | 1 |
| ~~Pointy~~ | 1 | 0 | 0 | Round | Absent | 1 |
| ~~Floppy~~ | 0 | 1 | 0 | Not round | Absent | 0 |
| ~~Oval~~ | 0 | 0 | 1 | Round | Absent | 1 |
| ~~Floppy~~ | 0 | 1 | 0 | Round | Absent | 0 |
| ~~Floppy~~ | 0 | 1 | 0 | Round | Absent | 0 |

Insbesondere können sie, anstatt ein ohrenförmiges Merkmal zu verwenden, einen von drei möglichen Werten annehmen. Wir werden stattdessen drei neue Features erstellen, wobei ein Feature lautet: Hat dieses Tier spitze Ohren, ein zweites ist, ob es Schlappohren hat, und das dritte ist, ob es ovale Ohren hat. Während wir im ersten Beispiel zuvor die Ohrform als spitz angegeben hatten, sagen wir jetzt stattdessen, dass dieses Tier einen Wert für das Merkmal „Spitze Ohren“ von 1 und 0 für schlaff und oval hat.

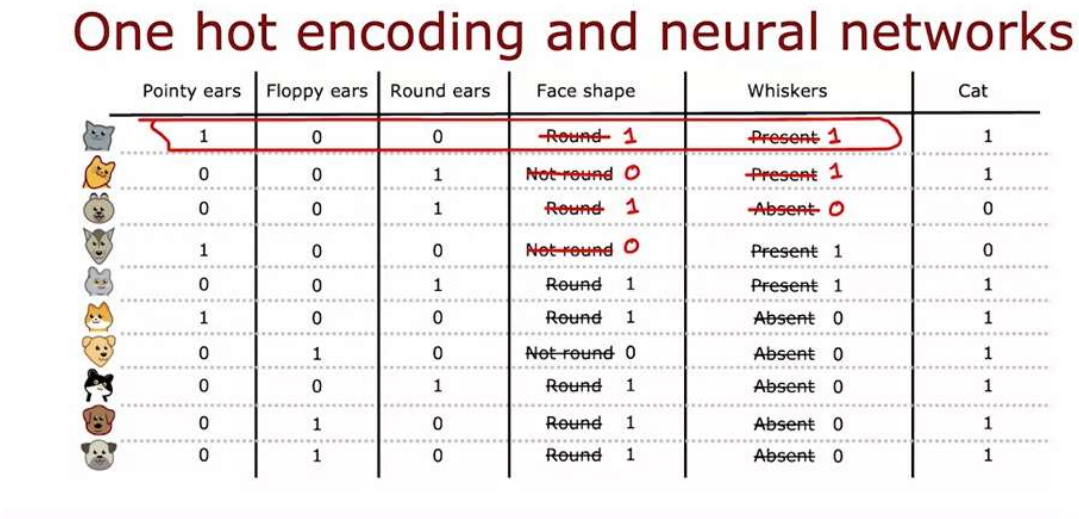
Während wir zuvor beim zweiten Beispiel gesagt haben, dass es ovale Ohren hat, sagen wir jetzt, dass es für spitze Ohren einen Wert von 0 hat, weil es keine spitzen Ohren hat. Es hat auch keine Schlappohren, aber ovale Ohren, weshalb dieser Wert hier 1 ist und so weiter für die restlichen Beispiele im Datensatz. Anstatt also, dass ein Feature drei mögliche Werte annimmt, haben wir jetzt drei neue Features erstellt, von denen jedes nur einen von zwei möglichen Werten annehmen kann, entweder 0 oder 1. Etwas detaillierter, wenn ein kategoriales Feature dies kann k mögliche Werte annehmen, k war in unserem Beispiel drei, dann werden wir es ersetzen, indem wir k binäre Merkmale erstellen, die nur die Werte 0 oder 1 annehmen können. Und Sie bemerken das unter all diesen drei Merkmalen, wenn Sie sich eines ansehen Hier ist genau einer der Werte gleich 1. Und das gibt dieser Methode der zukünftigen Konstruktion den Namen One-Hot-Codierung.

Und weil eines dieser Features immer den Wert 1 annimmt, ist das das Hot-Feature und daher der Name One-Hot-Codierung. Und mit dieser Auswahl an Merkmalen sind wir nun wieder bei der ursprünglichen Einstellung, bei der jedes Merkmal nur einen von zwei möglichen Werten annimmt, und daher wird der Entscheidungsbaum-Lernalgorithmus, den wir zuvor gesehen haben, ohne weitere Änderungen auf diese Daten angewendet . Ganz nebenbei: Obwohl sich das Material dieser Woche auf das Training von Entscheidungsbaummodellen konzentrierte, funktioniert die Idee, One-Hot-Codierungen zum Codieren kategorialer Merkmale zu verwenden, auch für das Training neuronaler Netze.




Insbesondere, wenn Sie die Funktion „Gesichtsform“ verwenden und „rund“ und „nicht rund“ durch 1 und 0 ersetzen würden, wobei „rund“ die Materie 1 erhält, „nicht rund“ die Materie 0 und so weiter. Und für Schnurrhaare ersetzen Sie in ähnlicher Weise das Vorhandensein durch 1 und das Fehlen durch 0. Sie bemerkten, dass wir alle kategorialen Merkmale, die uns zur Verfügung standen, wobei wir drei mögliche Werte für die Ohrform, zwei für die Gesichtsform und einen für die Schnurrhaare hatten, übernommen und als Liste davon kodiert haben fünf Funktionen.

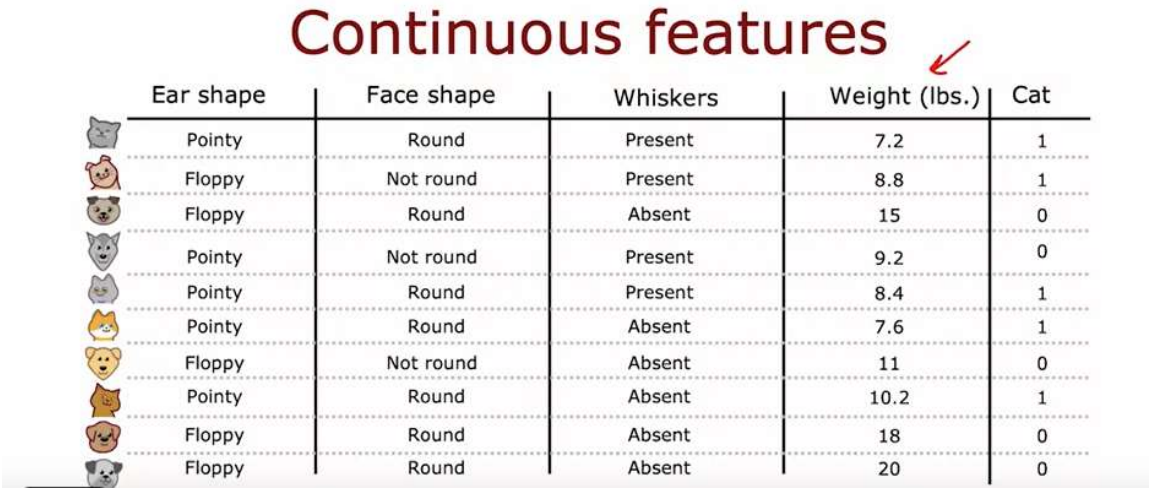
Drei aus der One-Hot-Codierung der Ohrform, eines aus der Gesichtsform und der Schnurrhaare, und jetzt kann diese Liste mit fünf Merkmalen auch einem neuen Netzwerk oder einer logistischen Regression zugeführt werden, um zu versuchen, einen Katzenklassifikator zu trainieren. One-Hot-Codierung ist also eine Technik, die nicht nur für das Lernen von Entscheidungsbäumen geeignet ist, sondern es Ihnen auch ermöglicht, kategoriale Merkmale mit Einsen und Nullen zu codieren, sodass sie auch als Eingaben in ein neuronales Netzwerk eingespeist werden können, das Zahlen als Eingaben erwartet.




Mit einer One-Hot-Codierung können Sie Ihren Entscheidungsbaum auf Funktionen anwenden, die mehr als zwei diskrete Werte annehmen können, und Sie können dies auch auf neue Netzwerk- oder lineare Regressions- oder logistische Regressionstrainings anwenden. Aber wie wäre es mit Merkmalen, bei denen es sich um Zahlen handelt, die jeden Wert annehmen können, nicht nur eine kleine Anzahl diskreter Werte? Sehen wir uns im nächsten Video an, wie Sie den Entscheidungsbaum dazu bringen können, kontinuierliche Wertfunktionen zu verarbeiten, die eine beliebige Zahl sein können.

## Kontinuierlich geschätzte Funktionen

Sehen wir uns an, wie Sie den Entscheidungsbaum ändern können, um mit Funktionen zu arbeiten, die nicht nur einen Streitwert, sondern einen kontinuierlichen Wert haben. Das sind Features, die beliebig viele sein können. Beginnen wir mit einem Beispiel: Ich habe den Datensatz des Katzenadoptionszentrums geändert, um eine weitere Funktion hinzuzufügen, nämlich das Gewicht des Tieres. In Pfund sind Katzen und Hunde im Durchschnitt etwas leichter als Hunde, obwohl es einige Katzen gibt, die schwerer sind als manche Hunde. Das Gewicht eines Tieres ist jedoch ein nützliches Kriterium für die Entscheidung, ob es sich um eine Katze handelt oder nicht. Wie erhält man also einen Entscheidungsbaum, um eine solche Funktion zu verwenden?

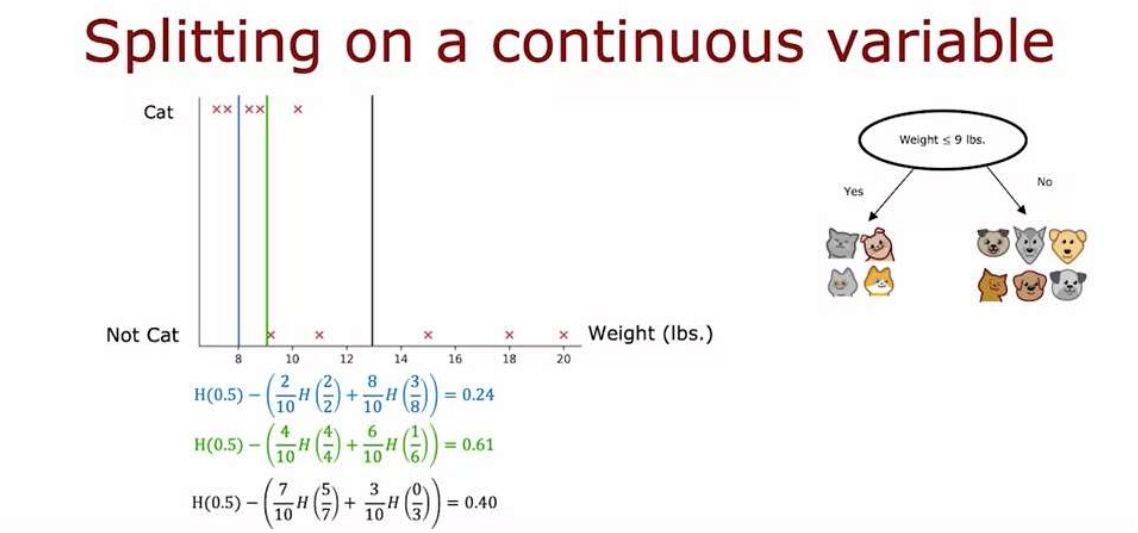



Der Entscheidungsbaum-Lernalgorithmus wird ähnlich wie zuvor vorgehen, mit der Ausnahme, dass keine Einschränkungsaufteilung nur nach Ohrform, Gesichtsform und Schnurrhaaren vorgenommen

wird. Sie müssen die Aufteilung nach Ohrform, Gesichtsform, Schnurrbart oder Gewicht vornehmen. Und wenn die Aufteilung nach Gewicht einen besseren Informationsgewinn bringt als die anderen Optionen. Anschließend erfolgt eine Aufteilung nach der Gewichtsfunktion. Aber wie entscheiden Sie, wie Sie die Gewichtsfunktion aufteilen? Lass uns einen Blick darauf werfen. Hier ist eine Darstellung der Daten an der Wurzel. Nicht auf der horizontalen Achse aufgetragen. Der Weg zum Tier und zur vertikalen Achse ist die Katze oben und nicht die Katze unten. Die vertikale Achse zeigt also die Bezeichnung an, wobei y 1 oder 0 ist. Die Art und Weise, wie wir nach der Gewichtsfunktion aufgeteilt würden, wäre, wenn wir die Daten danach aufteilen würden, ob die Gewichtung kleiner oder gleich einem bestimmten Wert ist oder nicht. Sagen wir 8 oder eine andere Zahl. Das wird die Aufgabe des Lernalgorithmus sein, dies auszuwählen. Und was wir tun sollten, wenn Einschränkungen für die Gewichtsfunktion aufgeteilt werden, ist, viele verschiedene Werte dieses Schwellenwerts zu berücksichtigen und dann den besten auszuwählen. Und mit „das Beste“ meine ich dasjenige, das den größten Informationsgewinn bringt. Wenn Sie also insbesondere erwägen, die Beispiele basierend darauf aufzuteilen, ob die Gewichtung kleiner oder gleich 8 ist, dann werden Sie diesen Datensatz in zwei Teilmengen aufteilen. Dabei hat die Teilmenge links zwei Katzen und die Teilmenge rechts drei Katzen und fünf Hunde. Wenn Sie also unsere übliche Informationsgewinnberechnung durchführen würden, würden Sie die Entropie am Grundton NC pf 0,5 minus jetzt 2/10 mal Entropie der linken Aufteilung berechnen, die zwei weitere Katzen hat. Es sollte also 2/2 sein und die rechte Aufteilung hat acht von zehn Beispielen und eine Entropie F. Das sind von den acht Beispielen auf der rechten Seite drei Katzen. Bei der Eingabe von 3/8 ergibt sich ein Wert von 0,24. Das wäre also ein Informationsgewinn, wenn Sie aufteilen würden, ob die Gewichtung kleiner als 8 ist, aber wir sollten auch andere Werte ausprobieren.

Was wäre also, wenn Sie aufteilen würden, ob die Gewichtung kleiner als 9 ist oder nicht, und das entspricht dieser neuen Linie hier? Und die Berechnung des Informationsgewinns ergibt H (0,5) minus. Jetzt haben wir also vier Beispiele und links alle Katzen aufgeteilt. Das ist also das 4/10-fache der Entropie von 4/4 plus sechs Beispiele, auf deren rechten Seite Sie eine Katze haben. Das ist also jeweils das 6/10-fache von 1/6, was 0,61 entspricht. Der Informationsgewinn sieht hier also viel besser aus und beträgt 0,61, was viel höher ist als 0,24. Oder wir könnten einen anderen Wert ausprobieren, beispielsweise 13. Und die Berechnung sieht so aus, nämlich 0,40. Im allgemeineren Fall werden wir tatsächlich nicht nur drei Werte ausprobieren, sondern mehrere Werte entlang der X-Achse.

Und eine Konvention wäre, alle Beispiele nach der Gewichtung oder dem Wert dieser Funktion zu sortieren und alle Werte zu nehmen, die in der Mitte der sortierten Trainingsliste liegen. Beispiele für die zu berücksichtigenden Werte für diesen Schwellenwert finden Sie hier. Wenn Sie über 10 Trainingsbeispiele verfügen, testen Sie auf diese Weise neun verschiedene mögliche Werte für diesen Schwellenwert und versuchen dann, denjenigen auszuwählen, der Ihnen den höchsten Informationsgewinn bringt.



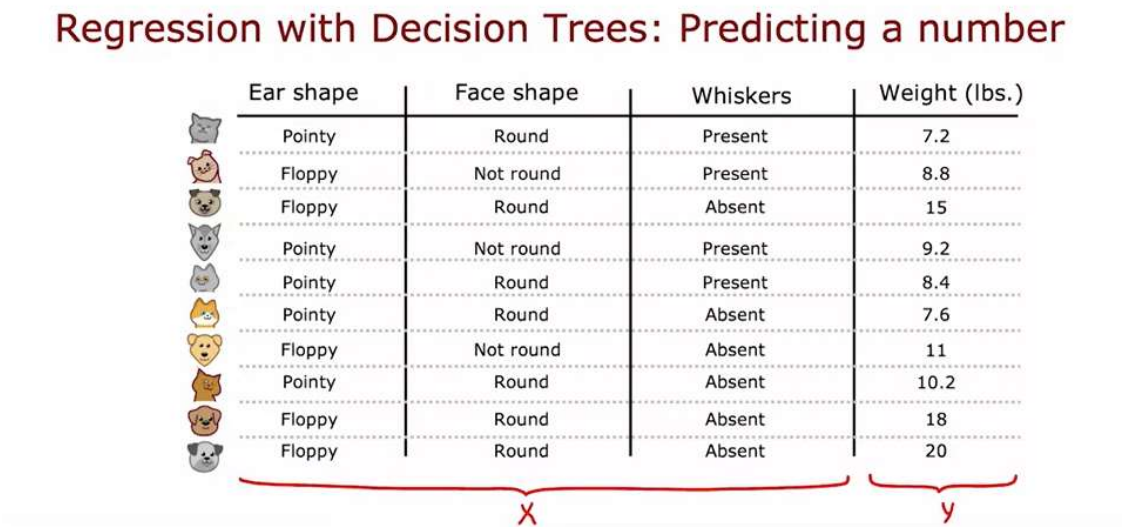

Und schließlich, wenn die durch die Aufteilung auf einen bestimmten Wert dieses Schwellenwerts gewonnenen Informationen besser sind als der Informationsgewinn durch die Aufteilung auf ein anderes Merkmal, werden Sie sich dafür entscheiden, diese Notiz an diesem Merkmal zu teilen. Und in diesem Beispiel erweist sich ein Informationsgewinn von 0,61 als höher als bei jedem anderen Feature. Es stellt sich heraus, dass es sich tatsächlich um zwei Schwellenwerte handelt. Unter der Annahme, dass der Algorithmus diese Funktion zur Aufteilung auswählt, teilen Sie den Datensatz letztendlich danach auf, ob das Gewicht des Tieres weniger als 9 £ beträgt oder nicht. Am Ende erhalten Sie zwei Teilmengen der Daten wie diese und können dann rekursive, zusätzliche Entscheidungsbäume erstellen, indem Sie diese beiden Teilmengen der Daten verwenden, um den Rest des Baums aufzubauen.

Um es also zusammenzufassen, damit der Entscheidungsbaum bei jeder Note an kontinuierlichen Wertmerkmalen arbeitet. Bei der Verwendung von Teilungen würden Sie einfach verschiedene Werte für die Teilung in Betracht ziehen, die übliche Berechnung des Informationsgewinns durchführen und sich für die Teilung anhand dieser kontinuierlichen Wertfunktion entscheiden, wenn sie den größtmöglichen Informationsgewinn bietet. Auf diese Weise bringen Sie den Entscheidungsbaum dazu, mit kontinuierlichen Wertmerkmalen zu arbeiten. Probieren Sie verschiedene Schwellenwerte aus, führen Sie die übliche Berechnung des Informationsgewinns durch und teilen Sie die kontinuierliche Wertfunktion mit dem ausgewählten Schwellenwert auf, wenn Sie so den bestmöglichen Informationsgewinn aus allen möglichen Funktionen für die Aufteilung erhalten. Und das ist alles mit den erforderlichen Videos zum Kernalgorithmus des Entscheidungsbaums. Anschließend gibt es ein optionales Video, das Sie ansehen können oder nicht und das den Entscheidungsbaum-Lernalgorithmus auf Regressionsbäume verallgemeinert. Bisher haben wir nur über die Verwendung von Entscheidungsbäumen gesprochen, um Vorhersagen zu treffen, bei denen es sich um Klassifizierungen handelt, die eine diskrete Kategorie vorhersagen, beispielsweise „Katze“ oder „Nicht-Katze“. Was aber, wenn Sie ein Regressionsproblem haben, bei dem Sie eine Zahl im nächsten Video vorhersagen möchten?

Ich werde über eine Verallgemeinerung von Entscheidungsbäumen sprechen, um damit umzugehen.

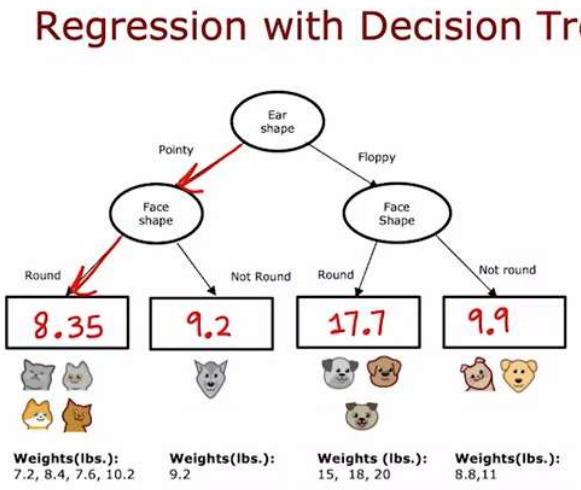
## Regressionsbäume (optional)

Bisher haben wir nur über Entscheidungsbäume als Klassifizierungsalgorithmen gesprochen. In diesem optionalen Video verallgemeinern wir Entscheidungsbäume als Regressionsalgorithmen, damit wir eine Zahl vorhersagen können. Lass uns einen Blick darauf werfen. Das Beispiel, das ich für dieses Video verwenden werde, besteht darin, die drei Funktionen zu verwenden, die wir zuvor hatten, d. h. diese Funktionen X, um das Gewicht des Tieres vorherzusagen, Y.

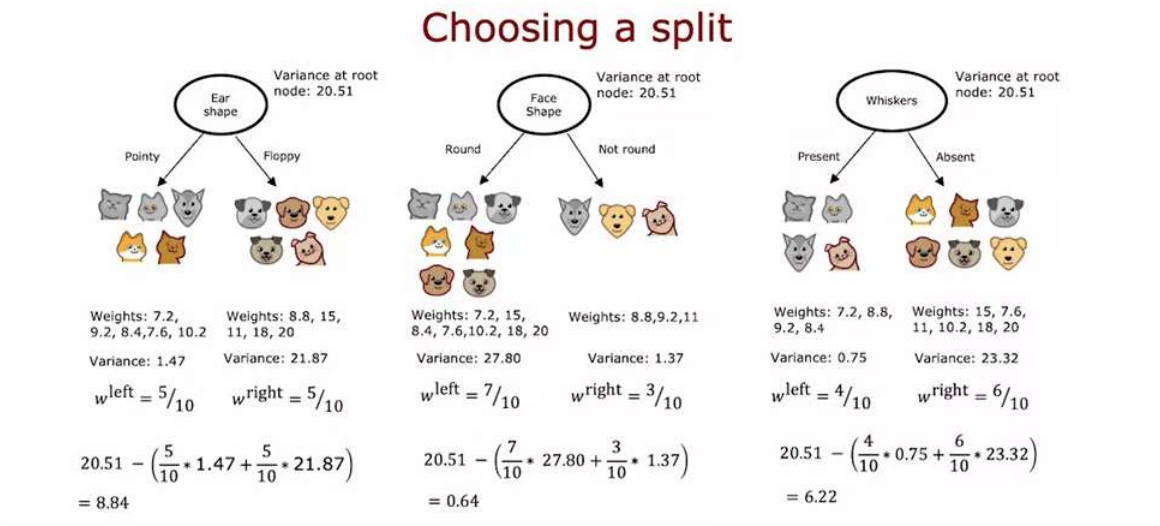



Um es klarzustellen, das Gewicht ist hier im Gegensatz zum vorherigen Video kein Eingabemerkmal mehr. Stattdessen ist dies die Zielausgabe Y, die wir vorhersagen möchten, anstatt vorherzusagen, ob ein Tier eine Katze ist oder nicht. Dies ist ein Regressionsproblem, da wir eine Zahl Y vorhersagen möchten. Schauen wir uns an, wie ein Regressionsbaum aussehen wird.

Hier habe ich bereits einen Baum für dieses Regressionsproblem erstellt, bei dem der Wurzelknoten nach der Ohrform aufgeteilt wird und dann der linke und der rechte Unterbaum nach der Gesichtsform und hier rechts auch nach der Gesichtsform aufgeteilt werden. Und es ist nichts falsch an einem Entscheidungsbaum, der sich für die Aufteilung nach demselben Merkmal sowohl im linken als auch im rechten Seitenzweig entscheidet. Es ist vollkommen in Ordnung, wenn der Aufteilungsalgorithmus dies vornimmt. Wenn Sie sich während des Trainings für diese Aufteilungen entschieden hätten, dann hätte dieser Knoten hier unten diese vier Tiere mit den Gewichten 7,2, 7,6 und 10,2. Dieser Knoten hätte dieses eine Tier mit dem Gewicht 9,2 und so weiter für diese verbleibenden zwei Knoten.




Das Letzte, was wir für diesen Entscheidungsbaum ausfüllen müssen, ist: Wenn es ein Testbeispiel gibt, das bis zu diesem Knoten reicht, welche Gewichte sollten wir für ein Tier mit spitzen Ohren und runder Gesichtsform vorhersagen? Der Entscheidungsbaum wird eine Vorhersage treffen, die auf dem Durchschnitt der Gewichtungen in den Trainingsbeispielen hier unten basiert. Und wenn man diese vier Zahlen mittelt, erhält man 8,35. Wenn ein Tier hingegen spitze Ohren und eine nicht runde Gesichtsform hat, wird es 9,2 oder 9,2 Pfund prognostizieren, weil das das Gewicht dieses einen Tieres hier unten ist. Und in ähnlicher Weise werden dies 17,70 und 9,90 sein. Dieses Modell wird also ein neues Testbeispiel erhalten, den Entscheidungsknoten wie üblich nach unten folgen, bis es zu einem Blattknoten gelangt, und dann den Wert am Blattknoten vorhersagen, den ich gerade berechnet habe, indem ich einen Durchschnitt der Gewichte von gebildet habe die Tiere, die während des Trainings bis zu demselben Blattknoten gelangt waren. Wenn Sie also mithilfe dieses Datensatzes einen Entscheidungsbaum von Grund auf erstellen würden, um die Gewichtung vorherzusagen. Die wichtigste Entscheidung wird, wie Sie Anfang dieser Woche gesehen haben, sein: Wie wählen Sie aus, welche Funktion Sie aufteilen möchten? Lassen Sie mich anhand eines Beispiels veranschaulichen, wie diese Entscheidung getroffen werden kann. Am Wurzelknoten könnten Sie beispielsweise die Ohrform aufteilen. Wenn Sie das tun, erhalten Sie linke und rechte Zweige des Baums mit fünf Tieren links und rechts mit den folgenden Gewichten. Würde man die Spieß-auf-Gesicht-Form wählen, erhält man links und rechts diese Tiere mit den entsprechenden Gewichten, die unten angeschrieben sind. Und wenn Sie sich für eine Aufteilung nach Vorhandensein oder Fehlen von Schnurrhaaren entscheiden würden, kämen Sie zu diesem Ergebnis. Die Frage ist also: Welches möchten Sie angesichts dieser drei möglichen Merkmale zur Aufteilung am Wurzelknoten auswählen, das die besten Vorhersagen für das Gewicht des Tieres liefert? Wenn wir einen Regressionsbaum erstellen, versuchen wir statt zu versuchen, die Entropie zu reduzieren, die das Maß für die Verunreinigung war, das wir für ein Klassifizierungsproblem hatten, stattdessen die Varianz der Gewichtung der Werte Y in jeder dieser Teilmengen der Daten zu reduzieren. Wenn Sie also den Begriff der Varianten in anderen Zusammenhängen gesehen haben, ist das großartig. Dies ist der statistische mathematische Begriff von Varianten, den wir gleich verwenden werden. Aber wenn Sie noch nie gesehen haben, wie man die Varianz einer Reihe von Zahlen berechnet, machen Sie sich darüber keine Sorgen. Alles, was Sie für diese Folie wissen müssen, ist, dass Varianten informell berechnen, wie stark eine Reihe von Zahlen variiert.




Für diese Zahlenmenge 7,2, 9,2 usw. bis 10,2 ergibt sich also eine Varianz von 1,47, also keine großen Schwankungen. Während hier 8,8, 15, 11, 18 und 20 sind, reichen diese Zahlen von 8,8 bis hin zu 20. Und so ist die Varianz viel größer, es ergibt sich eine Varianz von 21,87. Und so bewerten wir die Qualität der Aufteilung wie zuvor: W links und W rechts als Anteil der Beispiele, die zum linken und rechten Zweig gingen. Und die durchschnittliche Varianz nach der Aufteilung beträgt 5/10, also W links mal 1,47, also die Varianz links, und plus 5/10 mal die Varianz rechts, also 21,87. Diese gewichtete durchschnittliche Varianz spielt also eine sehr ähnliche Rolle wie die gewichtete durchschnittliche Entropie, die wir bei der Entscheidung, welche Aufteilung wir für ein Klassifizierungsproblem verwenden sollten, verwendet hatten. Und wir können diese Berechnung dann für die anderen möglichen Auswahlmöglichkeiten für die Aufteilung wiederholen. Hier im Baum in der Mitte beträgt die Varianz dieser Zahlen 27,80. Die Varianz beträgt hier 1,37. Wenn also W links sieben Zehntel und W rechts drei Zehntel entspricht, können Sie mit diesen Werten die gewichtete Varianz wie folgt berechnen.

Zum letzten Beispiel: Wenn Sie die Whiskers-Funktion aufteilen würden, ist dies die Varianz links und rechts, es gibt W links und W rechts. Das Gewicht der Varianz ist also folgendes. Eine gute Möglichkeit, eine Aufteilung zu wählen, wäre, einfach den Wert der gewichteten Varianz auszuwählen, der am niedrigsten ist. Ähnlich wie bei der Berechnung des Informationsgewinns werde ich an dieser Gleichung nur noch eine Modifikation vornehmen. Was das Klassifizierungsproblem betrifft, haben wir nicht nur die durchschnittliche gewichtete Entropie gemessen, sondern auch die Verringerung der Entropie, und das war ein Informationsgewinn.

Für einen Regressionsbaum messen wir auf ähnliche Weise auch die Verringerung der Varianz. Es stellt sich heraus, dass die Varianz aller Beispiele 20,51 beträgt, wenn Sie sich alle Beispiele im Trainingssatz, alle zehn Beispiele, ansehen und die Varianz aller Beispiele berechnen. Und das ist natürlich in allen Fällen derselbe Wert für den Root-Knoten, da es sich um die gleichen zehn Beispiele am Root-Knoten handelt. Was wir also tatsächlich berechnen, ist die Varianz des Wurzelknotens, die 20,51 minus diesem Ausdruck hier unten beträgt, was 8,84 ergibt.

Am Wurzelknoten betrug die Varianz also 20,51 und nach der Aufteilung nach Ohrform ist die durchschnittliche gewichtete Varianz an diesen beiden Knoten 8,84 niedriger. Die Varianzreduzierung beträgt also 8,84. Und wenn Sie den Ausdruck für die Varianzreduzierung für dieses Beispiel in der Mitte berechnen, beträgt er 20,51 minus diesem Ausdruck, den wir zuvor hatten, was sich als gleich 0,64 herausstellt. Das ist also eine sehr kleine Verringerung der Varianz. Und für die Whiskers-Funktion ergibt sich ein Wert von 6,22. Von allen drei Beispielen ergibt sich also bei 8,84 die größte

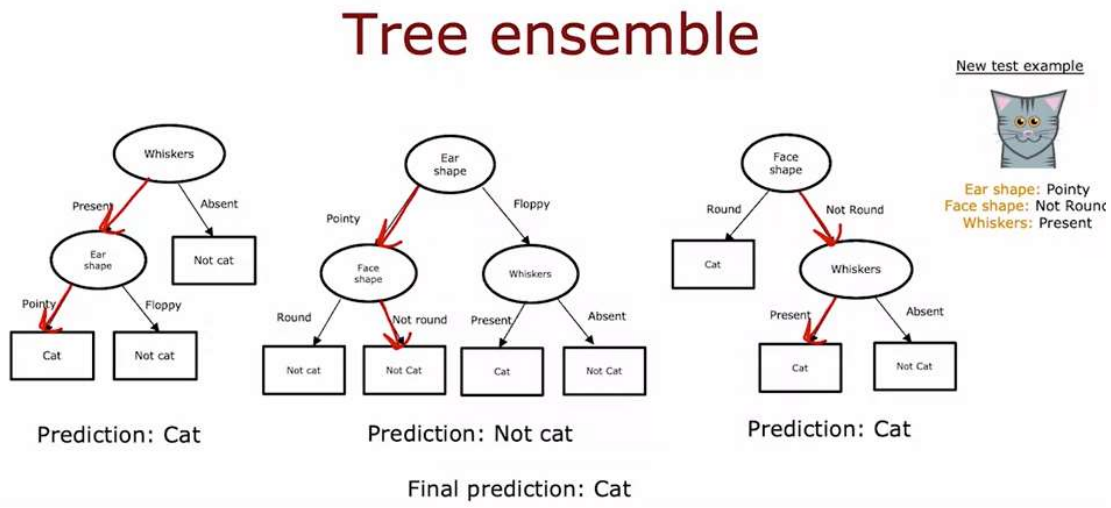
Reduzierung der Varianz. So wie wir zuvor die Funktion ausgewählt haben, die Ihnen den größten Informationsgewinn für einen Regressionsbaum bietet, wählen Sie nun die Funktion aus, die Ihnen die größte Verringerung der Varianz bietet.

Aus diesem Grund wählen Sie die Ohrform als Funktion für die Aufteilung. Nachdem Sie die jahresförmigen Merkmale ausgewählt haben, auf die Sie spucken möchten, verfügen Sie nun über zwei Teilmengen von fünf Beispielen im linken und rechten Seitenzweig, und Sie würden dann wiederum, wir sagen rekursiv, diese fünf Beispiele nehmen und einen neuen Entscheidungsbaum erstellen, auf den Sie sich konzentrieren Nur diese fünf Beispiele: Hier werden verschiedene Optionen für die Aufteilung bewertet und diejenige ausgewählt, die Ihnen die größte Varianzreduzierung bietet. Und ebenso rechts. Und Sie teilen sich so lange auf, bis Sie die Kriterien erfüllen, die eine weitere Aufteilung verhindern. Und das ist es. Mit dieser Technik können Sie Ihre Entscheidungsfindung nicht nur auf Klassifizierungsprobleme, sondern auch auf Regressionsprobleme anwenden. Bisher haben wir darüber gesprochen, wie man einen einzelnen Entscheidungsbaum trainiert. Es stellt sich heraus, dass Sie ein viel besseres Ergebnis erzielen können , wenn Sie viele Entscheidungsbäume trainieren, wir nennen dies ein Ensemble von Entscheidungsbäumen . Schauen wir uns im nächsten Video an, warum und wie das geht.

## Verwendung mehrerer Entscheidungsbäume

Eine der Schwächen der Verwendung eines einzelnen Entscheidungsbaums besteht darin, dass dieser Entscheidungsbaum sehr empfindlich auf kleine Änderungen in den Daten reagieren kann. Eine Lösung, um den Pfeil weniger empfindlich oder robuster zu machen, besteht darin, nicht einen Entscheidungsbaum, sondern viele Entscheidungsbäume zu erstellen, und wir nennen das ein Baumensemble. Lass uns einen Blick darauf werfen. Bei dem Beispiel, das wir verwendet haben, stellte sich heraus, dass die Ohrform das beste Merkmal für die Aufteilung am Wurzelknoten war, was zu diesen beiden Teilmengen der Daten führte und dann weitere Teilbäume auf diesen beiden Teilmengen der Daten baute. Aber es stellt sich heraus, dass, wenn Sie nur eines der zehn Beispiele nehmen und es in eine andere Katze verwandeln würden, so dass diese neue Katze statt spitzer Ohren, rundem Gesicht und fehlenden Schnurrhaaren Schlappohren, rundes Gesicht und vorhandene Schnurrhaare hat . Mit nur der Änderung eines einzigen Trainingsbeispiels wird die Funktion mit dem höchsten Informationsgewinn, auf die man sich aufteilen kann, zur Schnurrhaare-Funktion anstelle der Ohrform-Funktion.

Dadurch werden die Teilmengen der Daten, die Sie im linken und rechten Teilbaum erhalten, völlig unterschiedlich, und wenn Sie den Entscheidungsbaum-Lernalgorithmus weiterhin rekursiv ausführen, bauen Sie im linken und rechten Teil völlig unterschiedliche Teilbäume auf. Die Tatsache, dass die Änderung nur eines Trainingsbeispiels dazu führt, dass der Algorithmus eine andere Aufteilung an der Wurzel und damit einen völlig anderen Baum erstellt, macht diesen Algorithmus einfach nicht so robust. Aus diesem Grund erhalten Sie bei der Verwendung von Entscheidungsbäumen häufig ein viel besseres Ergebnis, d . h. Sie erhalten genauere Vorhersagen, wenn Sie nicht nur einen einzelnen Entscheidungsbaum, sondern eine ganze Reihe verschiedener Entscheidungsbäume trainieren. Dies nennen wir ein Baumensemble, was einfach eine Ansammlung mehrerer Bäume bedeutet. In den nächsten Videos werden wir sehen, wie man dieses Baumensemble aufbaut. Aber wenn Sie dieses Ensemble aus drei Bäumen hätten, wäre jeder einzelne davon vielleicht eine plausible Möglichkeit, Katze und Nicht-Katze zu klassifizieren.




Wenn Sie ein neues Testbeispiel hätten, das Sie klassifizieren möchten, würden Sie alle drei dieser Bäume in Ihrem neuen Beispiel ausführen und sie dazu bringen, darüber abzustimmen, ob es sich um die endgültige Vorhersage handelt. Dieses Testbeispiel hat spitze Ohren, eine nicht runde Gesichtsform und Schnurrhaare sind vorhanden, sodass der erste Baum Schlussfolgerungen wie diese ziehen und vorhersagen würde, dass es sich um eine Katze handelt. Die Schlussfolgerung des zweiten Baums würde diesem Pfad durch den Baum folgen und daher vorhersagen, dass es sich nicht um eine Katze handelt. Der dritte Baum würde diesem Weg folgen und daher vorhersagen, dass es sich um eine Katze handelt.

Diese drei Bäume haben unterschiedliche Vorhersagen gemacht und wir werden sie tatsächlich dazu bringen, abzustimmen. Die Mehrheit der Vorhersagen unter diesen drei Bäumen ist, Cat. Die endgültige Vorhersage dieses Baumensembles ist, dass es sich um eine Katze handelt, was zufällig die richtige Vorhersage ist. Der Grund, warum wir ein Ensemble von Bäumen verwenden, liegt darin, dass wir viele Entscheidungsbäume haben und diese abstimmen.

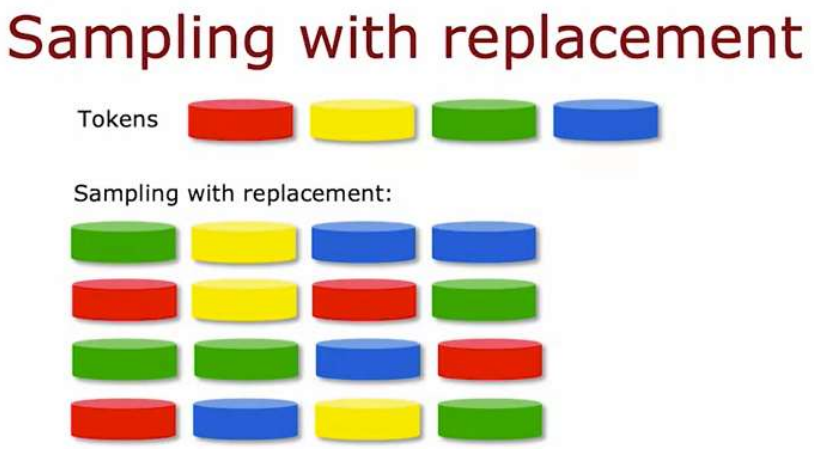
Dadurch wird Ihr Gesamtalgorithmus weniger empfindlich gegenüber dem, was ein einzelner Baum tun könnte, da er nur eine Stimme von drei oder eine Stimme von vielen erhält , viele verschiedene Stimmen und es macht Ihren gesamten Algorithmus robuster. Aber wie kommt man auf all diese verschiedenen plausiblen, aber vielleicht leicht unterschiedlichen Entscheidungsbäume, um sie zum Wählen zu bewegen? Im nächsten Video werden wir über eine Technik aus der Statistik sprechen, die Sampling mit Ersetzung genannt wird. Dies wird sich als eine Schlüsseltechnik herausstellen, die wir

im folgenden Video verwenden werden, um dieses Baumensemble aufzubauen. Fahren wir mit dem nächsten Video fort, um über die Probenahme mit Austausch zu sprechen.

## Ziehen mit Zurücklegen

Um ein Baumensemble aufzubauen, benötigen wir eine Technik namens „Sampling with Replacement“. Werfen wir einen Blick darauf, was das bedeutet. Um zu veranschaulichen, wie das Sampling mit Ersetzung funktioniert, zeige ich Ihnen eine Demonstration des Sampling mit Ersetzung unter Verwendung von vier Token in den Farben Rot, Gelb, Grün und Blau. Ich habe hier tatsächlich vier Farbmarken bei mir: Rot, Gelb, Grün und Blau. Ich werde zeigen, wie das Sampling mit Ersatz unter Verwendung dieser Systeme aussieht.

Hier ist ein schwarzer Samtbeutel, leer. Ich nehme dieses Beispiel von vier Token und lege sie hinein. Ich werde viermal mit Ersatz aus dieser Tasche probieren. Das heißt, ich werde es durcheinander bringen und kann nicht sehen, wann ich einen Token auswähle, der sich als grün herausstellt. Der Begriff „Ersetzen“ bedeutet, dass ich, wenn ich den nächsten Spielstein herausnehme, diesen nehme, ihn wieder hineinstecke, ihn erneut schüttle und dann einen anderen nehme, gelb. Ersetze es. Das ist ein kleines Ersatzteil. Dann gehen Sie noch einmal, ersetzen Sie es erneut durch Blau und wählen Sie dann ein weiteres aus, das wieder blau ist.

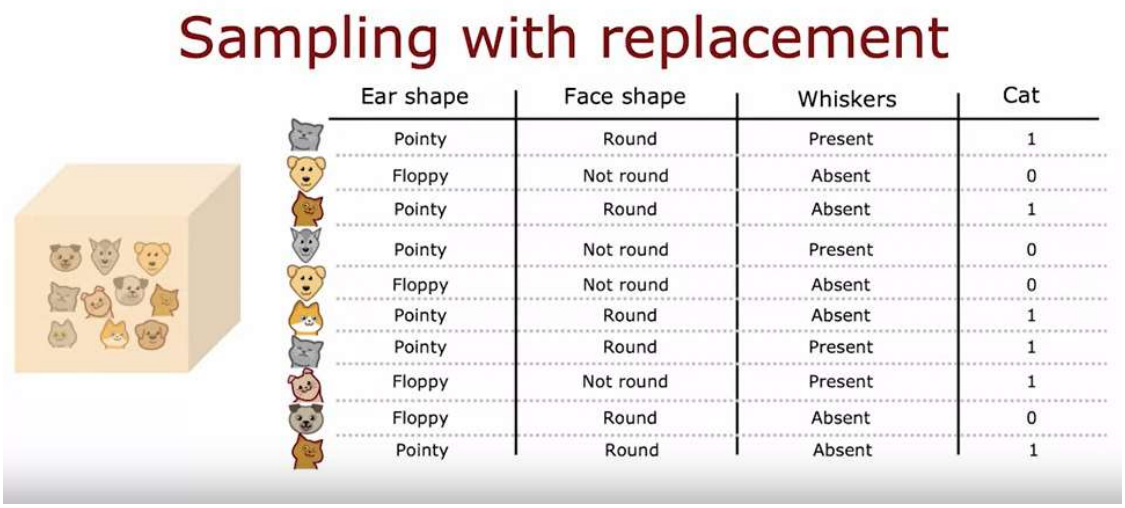



Die Reihenfolge der Spielsteine, die ich bekam, war grün, gelb, blau, blau. Beachten Sie, dass ich zweimal blau wurde und kein einziges Mal rot. Wenn Sie diesen Probenahme- und Austauschvorgang mehrmals wiederholen würden, könnten Sie Rot, Gelb, Rot, Grün oder Grün, Grün, Blau, Rot erhalten. Oder Sie erhalten auch Rot, Blau, Gelb und Grün. Beachten Sie, dass der Teil mit dem Austausch von entscheidender Bedeutung ist, denn wenn ich nicht jedes Mal, wenn ich probiere, einen Token ersetze, würde ich, wenn ich vier Token aus meinem Viererbeutel einschütte, immer genau die gleichen vier Token erhalten.

Deshalb ist es wichtig, jedes Mal einen Token auszutauschen, nachdem ich ihn herausgezogen habe, um sicherzustellen, dass ich nicht jedes Mal die gleichen vier Token bekomme. Die Art und Weise, wie die Probenahme mit Ersatz beim Aufbau eines Baumensembles angewendet wird, ist wie folgt.

Wir werden mehrere zufällige Trainingssätze erstellen, die sich alle geringfügig von unserem ursprünglichen Trainingssatz unterscheiden. Insbesondere werden wir unsere 10 Beispiele von Katzen und Hunden heranziehen. Wir packen die 10 Trainingsbeispiele in eine Theorietasche. Ich verwende diese theoretische Tasche und wir werden einen neuen zufälligen Trainingssatz mit 10 Beispielen erstellen, der genau die gleiche Größe wie der ursprüngliche Datensatz hat. Wir tun dies,

indem wir ein zufälliges Trainingsbeispiel auswählen. Nehmen wir an, wir erhalten dieses Trainingsbeispiel. Dann legen wir es zurück in die Tasche und wählen dann wieder zufällig ein Trainingsbeispiel aus, und so erhalten Sie das. Du wählst immer und immer wieder aus. Beachten Sie, dass dieses fünfte Trainingsbeispiel mit dem zweiten identisch ist, das wir da draußen hatten. Aber das ist in Ordnung.

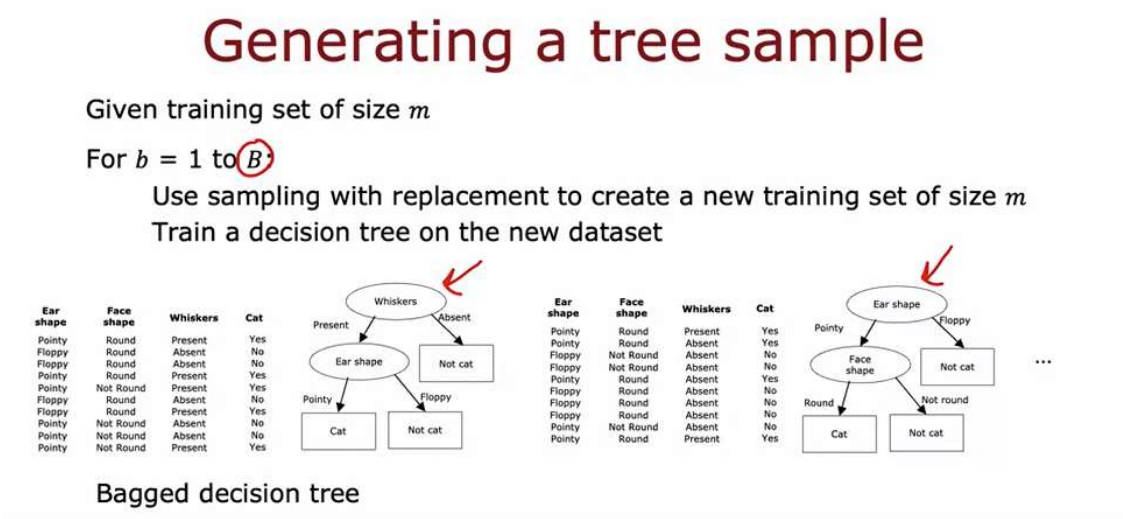



Man macht weiter und weiter, und wir erhalten eine weitere Wiederholung des Beispiels und so weiter und so fort. Bis Sie schließlich bei 10 Trainingsbeispielen stehen, von denen einige Wiederholungen sind. Sie bemerken auch, dass dieser Trainingssatz nicht alle 10 Original-Trainingsbeispiele enthält, aber das ist in Ordnung. Das ist Teil des Probenahme- und Austauschverfahrens. Durch den Prozess der Stichprobenziehung mit Ersetzung können Sie einen neuen Trainingssatz erstellen, der Ihrem ursprünglichen Trainingssatz ein wenig ähnelt, sich aber auch deutlich von ihm unterscheidet. Es stellt sich heraus, dass dies der Schlüsselbaustein für den Aufbau eines Baumensembles wäre. Schauen wir uns im nächsten Video an, wie Sie das machen können.

## Random-Forest-Algorithmus

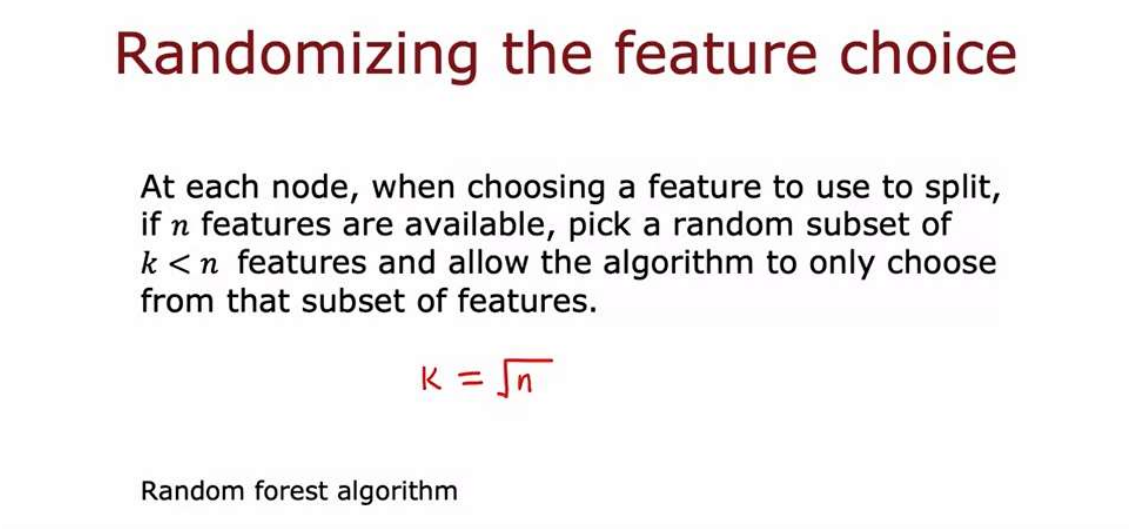
Jetzt haben wir die Möglichkeit, etwas mit Ersatz zu verwenden, um neue Trainingssätze zu erstellen, die dem ursprünglichen Trainingssatz ein wenig ähneln, sich aber auch deutlich von ihm unterscheiden. Wir sind bereit, unseren ersten Baumensemble-Algorithmus zu erstellen. In diesem Video werden wir insbesondere über den Random-Forest-Algorithmus sprechen, der ein leistungsstarker Tree-on-Sample-Algorithmus ist, der viel besser funktioniert als die Verwendung eines einzelnen Entscheidungsbaums. So können wir ein Ensemble von Bäumen erzeugen. Wenn Ihnen ein Trainingssatz der Größe M gegeben wird, dann ist B gleich 1 mit Großbuchstabe b, also machen wir das Großbuchstabe B mal. Sie können etwas mit Ersatz verwenden, um einen neuen Trainingssatz der Größe M zu erstellen. Wenn Sie also 10 Trainingsbeispiele haben, legen Sie die 10 Trainingsbeispiele 10 Mal in diese virtuelle Tasche und probieren den Ersatz aus, um einen neuen Trainingssatz mit ebenfalls 10 zu erstellen Beispiele, und dann würden Sie einen Entscheidungsbaum auf diesem Datensatz trainieren. Hier ist also der Datensatz, den ich mit etwas mit Ersetzung generiert habe. Wenn Sie genau hinschauen, werden Sie möglicherweise bemerken, dass einige der Trainingsbeispiele wiederholt werden, und das ist in Ordnung. Und wenn Sie die Entscheidung anhand dieser Daten trainieren, erhalten Sie diesen Entscheidungsbaum. Und nachdem wir das einmal gemacht hatten, wiederholten wir es ein zweites Mal. Verwenden Sie etwas mit Ersetzung, um einen weiteren Trainingssatz mit M oder 10 Trainingsbeispielen zu generieren. Auch dieses sieht ein wenig wie das Original-Trainingsset aus, ist aber auch ein wenig anders. Anschließend trainieren

Sie den Entscheidungsbaum anhand dieses neuen Datensatzes und erhalten am Ende einen etwas anderen Entscheidungsbaum.




Usw. Und Sie können dies insgesamt B-mal tun. Typische Wahl von Großbuchstaben B: Die Anzahl solcher Bäume, die Sie gebaut haben, könnte bei etwa 100 Personen liegen, jeder Wert von sagen wir 64.228. Und nachdem Sie ein Ensemble von sagen wir 100 verschiedenen Bäumen gebaut haben, würden Sie das tun, wenn Sie versuchen, eine Vorhersage zu treffen , Holen Sie sich alle Stimmen dieser Bäume für die richtige endgültige Vorhersage. Es stellt sich heraus, dass die Festlegung eines größeren Kapitals B niemals der Leistung schadet, aber ab einem bestimmten Punkt führt dies zu sinkenden Renditen, und es wird nicht wirklich besser, wenn B viel größer als sagen wir 100 oder so ist. Und deshalb verwende ich nie, sagen wir, 1000 Bäume, was die Berechnung nur erheblich verlangsamt, ohne die Leistung des Gesamtalgorithmus wesentlich zu steigern. Nur um diesem speziellen Album einen Namen zu geben. Diese spezielle Instanzerstellung eines Baumensembles wird manchmal auch als Bagged Decision Tree bezeichnet. Und das bezieht sich darauf, dass Sie Ihre Trainingsbeispiele in diese virtuelle Tasche stecken. Und deshalb verwenden wir hier auch das Kleinbuchstabe B und das Großbuchstabe B, denn das steht für Tasche. Es gibt eine Modifikation an diesem Album, die dafür sorgt, dass es noch viel besser funktioniert, und die diesen Algorithmus vom Bagged-Decision-Tree in den Random-Forest-Algorithmus umwandelt. Der Kerngedanke besteht darin, dass man trotz dieser Sampling- und Ersetzungsprozedur manchmal immer die gleiche Aufteilung am Grundknoten und sehr ähnliche Aufteilungen in der Nähe des Grundtons verwendet. Das ist in diesem speziellen Beispiel nicht passiert, wo eine kleine Änderung der Übungen zu einer anderen Teilung am Grundton führte. Aber bei anderen Trainingssätzen ist es nicht ungewöhnlich, dass Sie bei vielen oder sogar allen großen B-Trainingssätzen am Grundknoten und bei einigen Noten in der Nähe der Grundnote die gleiche Auswahl an Funktionen haben. Es gibt also eine Modifikation am Algorithmus, um weiter zu versuchen, die Merkmalsauswahl bei jeder Note zu randomisieren, was dazu führen kann, dass sich die Bäume und Sie lernen, sich voneinander zu unterscheiden. Wenn Sie also für sie abstimmen, erhalten Sie am Ende eine noch genauere Vorhersage. Dies geschieht normalerweise bei jeder Note, wenn ein Feature zum Schlitzen ausgewählt wird, sofern Endfeatures verfügbar sind. In unserem Beispiel standen uns also drei Features zur Verfügung, anstatt aus allen End-Features auszuwählen. Stattdessen wählen wir eine zufällige Teilmenge von K weniger als N Features aus. Und erlauben Sie dem Algorithmus, nur aus dieser Teilmenge von K Merkmalen auszuwählen. Mit anderen Worten: Sie würden K Features als zulässige Features auswählen und dann aus diesen K Features dasjenige mit dem höchsten Informationsgewinn als Feature zur Verwendung der Aufteilung auswählen. Wenn N groß ist, sagen wir, dass n Dutzende oder Zehner

oder sogar Hunderter ist. Eine typische Wahl für den Wert von K wäre, ihn als Quadratwurzel von N zu wählen.



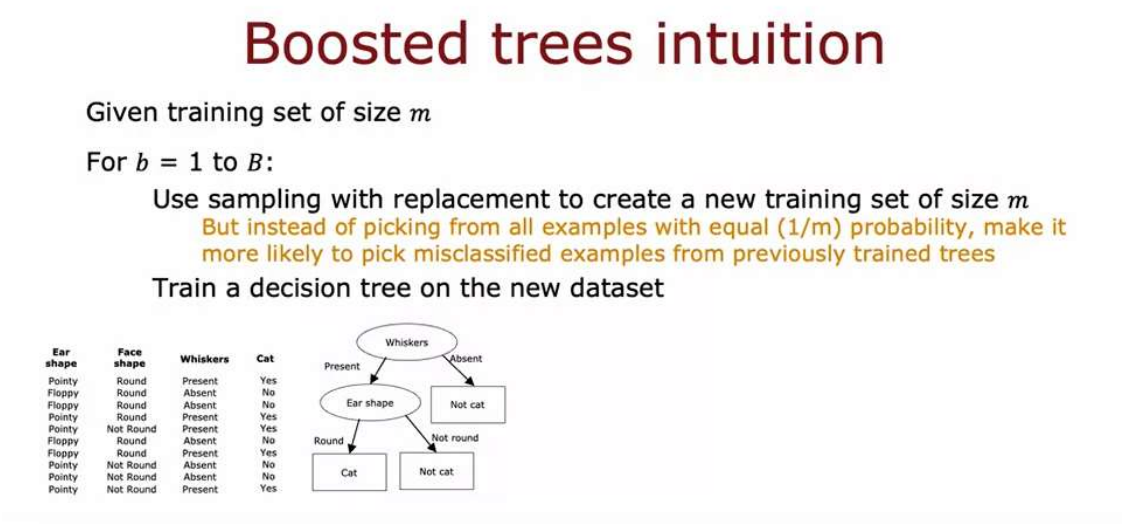

In unserem Beispiel haben wir nur drei Merkmale und diese Technik wird tendenziell eher für größere Probleme mit einer größeren Anzahl von Merkmalen verwendet. Und wenn Sie den Algorithmus nur noch weiter ändern, erhalten Sie am Ende den Random-Forest-Algorithmus, der in der Regel viel besser funktioniert und viel robuster wird als nur ein einzelner Entscheidungsbaum. Eine Möglichkeit, darüber nachzudenken, warum dies robuster ist als ein einzelner Entscheidungsbaum, besteht darin, dass etwas mit der Ersetzungsprozedur dazu führt, dass der Algorithmus bereits viele kleine Änderungen an den Daten untersucht, verschiedene Entscheidungsbäume trainiert und über alle diese Änderungen mittelt zu den Daten, die das Etwas mit Ersatzvorgang verursacht. Das bedeutet also, dass jede kleine Änderung am Trainingssatz weniger wahrscheinlich einen großen Einfluss auf die Gesamtausgabe des gesamten Random-Forest-Algorithmus hat. Weil es bereits erforscht ist und über viele kleine Änderungen am Trainingssatz gemittelt wird. Bevor ich dieses Video abschließe, möchte ich noch einen Gedanken mitteilen. Wo zeltet ein Machine-Learning-Ingenieur? In einem zufälligen Wald. In Ordnung. Geh und erzähl deinen Freunden diesen Witz. Ich hoffe du genießt es. Der Zufallswald ist ein effektiver Raum für uns und ich hoffe, Sie nutzen ihn besser für Ihre Arbeit. Jenseits der Zufallsstruktur stellt sich heraus, dass es einen anderen Algorithmus gibt, der noch besser funktioniert. Das ist ein verstärkter Entscheidungsbaum. Lassen Sie uns im nächsten Video über einen verstärkten Entscheidungsbaumalgorithmus namens XG Boost sprechen.

## XGBoost

Im Laufe der Jahre haben Forscher des maschinellen Lernens viele verschiedene Möglichkeiten entwickelt, Entscheidungsbäume und Entscheidungsbäume anhand von Stichproben zu erstellen. Die heute mit Abstand am häufigsten verwendete Methode oder Implementierung von Entscheidungsbaum-Ensembles oder Entscheidungsbäumen ist ein Algorythmus namens XGBoost. Es läuft schnell, die Open-Source-Implementierungen sind einfach zu verwenden und wurden auch sehr erfolgreich bei vielen Machine-Learning-Wettbewerben sowie in vielen kommerziellen Anwendungen eingesetzt.

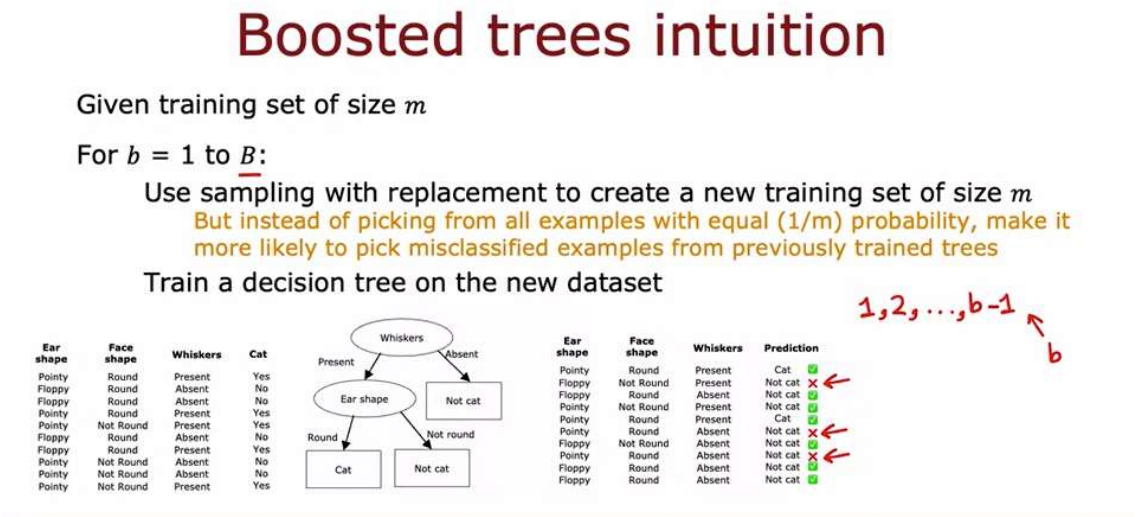
Werfen wir einen Blick darauf, wie XGBoost funktioniert. Es gibt eine Modifikation am Back-Decision-Tree-Algorithmus, die wir im letzten Video gesehen haben und die dafür sorgen kann, dass er viel besser funktioniert. Hier ist noch einmal das Album, das wir zuvor geschrieben hatten. Angesichts des Trainingssatzes zur Größenbestimmung wiederholen Sie B-mal, verwenden etwas mit Ersetzung, um

einen neuen Trainingssatz der Größe M zu erstellen, und trainieren dann den Entscheidungsbaum anhand des neuen Datensatzes. Beim ersten Durchlaufen dieser Schleife erstellen wir möglicherweise einen solchen Trainingssatz und trainieren einen solchen Entscheidungsbaum. Aber hier werden wir den Algorithmus ändern, und zwar jedes Mal, wenn diese Schleife durchlaufen wird, außer beim ersten Mal, also beim zweiten Mal, dritten Mal und so weiter. Anstatt bei der Stichprobenziehung aus allen m Beispielen gleicher Wahrscheinlichkeit mit einer Wahrscheinlichkeit von eins über m auszuwählen, erhöhen wir die Wahrscheinlichkeit, dass wir falsch klassifizierte Beispiele auswählen, bei denen die zuvor trainierten Bäume schlecht abschneiden.

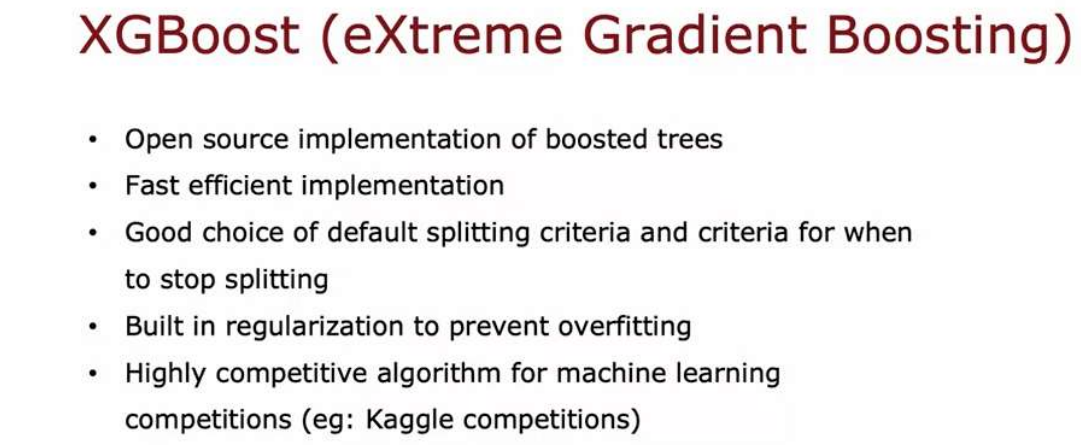



In der Ausbildung und Ausbildung gibt es eine Idee, die man bewusstes Üben nennt. Wenn Sie zum Beispiel Klavier spielen lernen und versuchen, ein Stück auf dem Klavier zu beherrschen, anstatt immer wieder das ganze fünfminütige Stück zu üben, was ziemlich zeitaufwändig ist. Wenn Sie stattdessen das Stück spielen und dann Ihre Aufmerksamkeit nur auf die Teile des Stücks richten, die Sie noch nicht so gut spielen, üben Sie diese kleineren Teile immer wieder. Dann erweist sich das als eine effizientere Möglichkeit für Sie, gutes Klavierspielen zu erlernen.

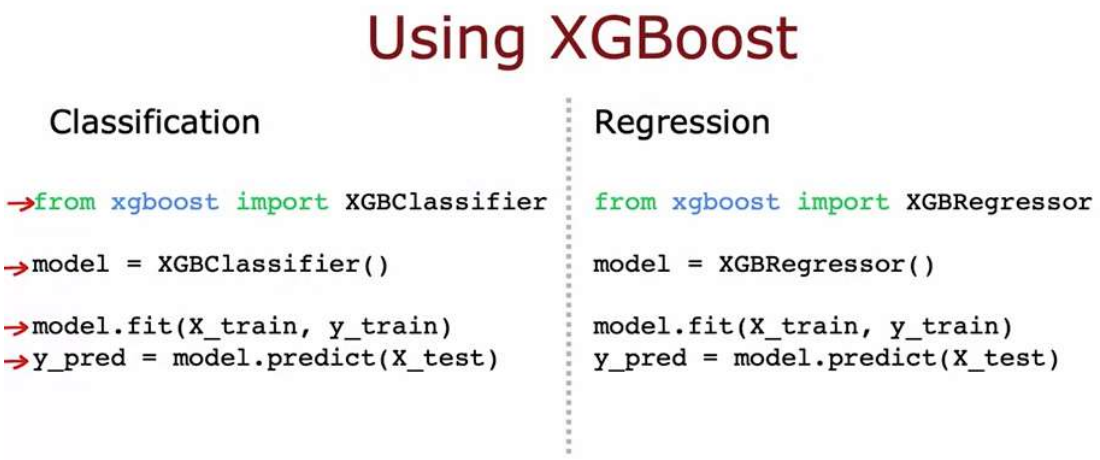
Und daher ist diese Idee des Boostens ähnlich. Wir schauen uns die Entscheidungsbäume an, die wir bisher trainiert haben, und schauen uns an, wo wir noch nicht gut vorankommen. Und wenn wir dann den nächsten Entscheidungsbaum erstellen, werden wir uns stärker auf die Beispiele konzentrieren, bei denen wir noch nicht gut abschneiden. Anstatt also alle Trainingsbeispiele zu betrachten, konzentrieren wir uns mehr auf die Teilmenge der Beispiele, bei denen es noch nicht gut läuft, und holen uns den neuen Entscheidungsbaum, das nächste Entscheidungsbaum-Reporting-Ensemble, um zu versuchen, bei ihnen gut abzuschneiden. Und das ist die Idee hinter dem Boosting, und es stellt sich heraus, dass es dem Lernalgorithmus dabei hilft, schneller zu lernen, bessere Leistungen zu erbringen. Daher werden wir uns diesen Baum, den wir gerade erstellt haben, im Detail ansehen und zum ursprünglichen Trainingssatz zurückkehren. Beachten Sie, dass es sich hierbei um den ursprünglichen Trainingssatz handelt und nicht um einen, der durch einen Ersatz generiert wurde. Und wir gehen alle zehn Beispiele durch und schauen uns an, was dieser erlernte Entscheidungsbaum für alle zehn Beispiele vorhersagt.




Diese vierthäufigste Spalte enthält also ihre Vorhersagen. Neben jedem Beispiel wird ein Häkchen gesetzt, je nachdem, ob die Klassifizierung der Bäume richtig oder falsch war. Was wir also beim zweiten Durchlauf dieser Schleife tun werden, ist, dass wir sozusagen etwas mit Ersetzung verwenden, um einen weiteren Trainingssatz mit zehn Beispielen zu generieren. Aber jedes Mal, wenn wir ein Beispiel aus diesen zehn auswählen, steigt die Chance, eines dieser drei Beispiele auszuwählen, die immer noch falsch klassifiziert wurden. Und so lenkt dies die Aufmerksamkeit des zweiten Entscheidungsbaums durch einen Prozess wie bewusstes Üben auf die Beispiele, dass das Album immer noch nicht so gut läuft. Und das Boosting-Verfahren wird dies insgesamt B Mal tun, wobei Sie bei jeder Iteration sehen, was das Ensemble der Bäume für die Bäume 1, 2 bis (b-1) noch nicht so gut macht. Und wenn Sie Baum Nummer b erstellen, ist die Wahrscheinlichkeit höher, dass Sie Beispiele auswählen, bei denen das Ensemble der zuvor ausgewählten Bäume immer noch nicht gut abschneidet. Die mathematischen Details, um wie viel genau die Wahrscheinlichkeit erhöht werden soll, dieses Beispiel im Vergleich zu jenem auszuwählen, sind ziemlich komplex, aber Sie müssen sich darüber keine Gedanken machen, um Russet-Tree-Implementierungen zu verwenden. Und von den verschiedenen Möglichkeiten, Boosting zu implementieren, ist XGBoost heute die am weitesten verbreitete, was für Extreme Gradient Boosting steht und eine Open-Source-Implementierung von Boosted Trees ist, die sehr schnell und effizient ist. XGBoost verfügt außerdem über eine gute Auswahl an Standard-Aufteilungskriterien und Kriterien dafür, wann die Aufteilung beendet werden soll. Und eine der Neuerungen in XGBoost besteht darin, dass es auch eine Regularisierung integriert hat, um eine Überanpassung zu verhindern, und in Wettbewerben für maschinelles Lernen eingebunden ist, wie beispielsweise bei einer weit verbreiteten Wettbewerbsseite namens Kaggle. XGBoost ist oft ein hart umkämpfter Algorithmus. Tatsächlich scheinen XGBoost und Deep-Learning-Algorithmen die beiden Arten von Algorithmen zu sein, die viele dieser Wettbewerbe gewinnen. Und noch ein technischer Hinweis: Anstatt etwas mit Ersatz zu tun, weist XGBoost tatsächlich verschiedenen Trainingsbeispielen unterschiedliche Wege zu. Es müssen also nicht wirklich viele zufällig ausgewählte Trainingssätze generiert werden, was es sogar ein wenig effizienter macht als die Verwendung eines Stichprobenverfahrens mit Ersetzung. Aber die Intuition, die Sie auf der vorherigen Folie gesehen haben, ist immer noch richtig, wenn es darum geht, wie XGBoost Beispiele auswählt, auf die man sich konzentrieren möchte. Die Details von XGBoost sind recht komplex zu implementieren, weshalb viele Praktiker die Open-Source-Bibliotheken verwenden, die XGBoost implementieren.




Das ist alles, was Sie tun müssen, um XGBoost zu verwenden. Sie importieren die XGBoost-Bibliothek wie folgt und initialisieren ein Modell als XGBoost-Klassifikator. Weiteres Modell und schließlich können Sie mithilfe dieses verstärkten Entscheidungsbaumalgorithmus Vorhersagen treffen. Ich hoffe, dass Sie diesen Algorithmus für viele Anwendungen nützlich finden, die Sie in Zukunft erstellen werden. Oder alternativ, wenn Sie XGBoost für die Regression statt für die Klassifizierung verwenden möchten, dann wird diese Zeile hier einfach zu XGBRegressor und der Rest des Codes funktioniert ähnlich. Das war's also mit dem XGBoost-Algorithmus.

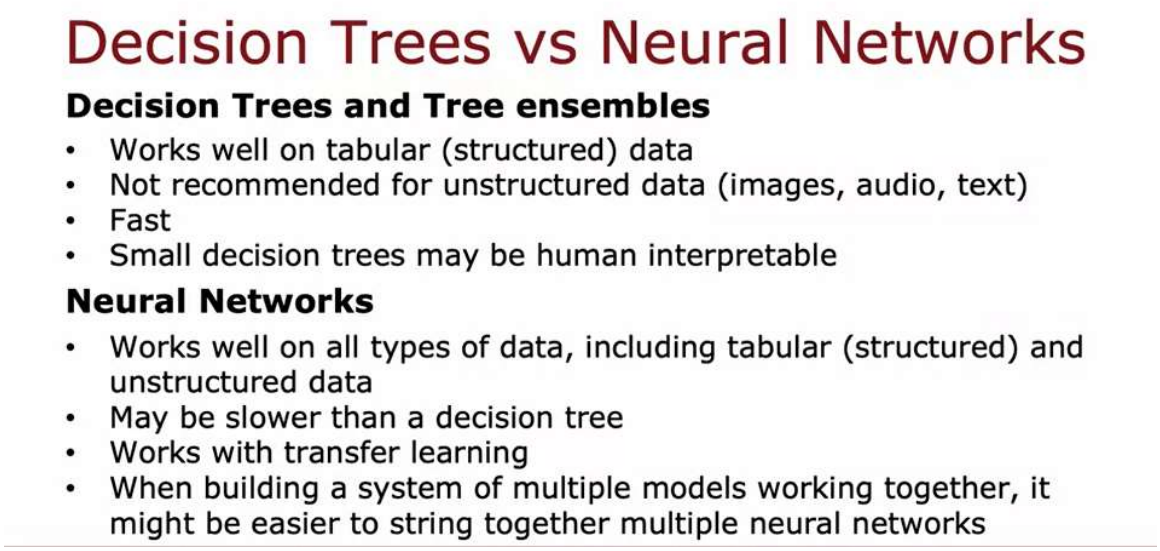



Wir haben nur ein letztes Video für diese Woche und für diesen Kurs, in dem wir abschließen und auch darüber sprechen, wann Sie einen Entscheidungsbaum oder vielleicht das neuronale Netzwerk verwenden sollten. Kommen wir zum letzten und letzten Video dieser Woche.

## Wann werden Entscheidungsbäume verwendet?

Sowohl Entscheidungsbäume, einschließlich Baumensembles als auch neuronale Netze, sind sehr leistungsfähige und sehr effektive Lernalgorithmen. Wann sollten Sie sich für das eine oder andere entscheiden? Schauen wir uns einige der Vor- und Nachteile jedes einzelnen an.

Entscheidungsbäume und Baumensembles eignen sich oft gut für tabellarische Daten, auch strukturierte Daten genannt. Das heißt, wenn Ihr Datensatz wie eine riesige Tabelle aussieht, sind Entscheidungsbäume eine Überlegung wert. Beispielsweise verfügten wir in der Anwendung zur Vorhersage von Immobilienpreisen über einen Datensatz mit Merkmalen, die der Größe des Hauses, der Anzahl der Schlafzimmer, der Anzahl der Stockwerke und dem Alter des Hauses entsprachen. Diese Art von Daten, die in einer Tabelle mit entweder kategorialen oder kontinuierlich bewerteten Merkmalen gespeichert werden und sowohl zur Klassifizierung als auch für Regressionsaufgaben dienen, bei denen Sie versuchen, eine diskrete Kategorie oder eine Zahl vorherzusagen. All diese Probleme können mit Entscheidungsbäumen gut gelöst werden. Im Gegensatz dazu werde ich die Verwendung von Entscheidungsbäumen und Baumensembles für unstrukturierte Daten nicht empfehlen. Dabei handelt es sich um Daten wie Bilder, Videos, Audiodateien und Texte, die Sie wahrscheinlich nicht in einem Tabellenkalkulationsformat speichern.




Wie wir gleich sehen werden, funktionieren neuronale Netze tendenziell besser für unstrukturierte Datenaufgaben. Ein großer Vorteil von Entscheidungsbäumen und Baumensembles besteht darin, dass sie sehr schnell trainiert werden können. Vielleicht erinnern Sie sich an dieses Diagramm aus der Vorwoche, in dem wir über die iterative Schleife der maschinellen Lernentwicklung gesprochen haben. Wenn das Training Ihres Modells viele Stunden dauert, schränkt dies die Geschwindigkeit ein, mit der Sie diese Schleife durchlaufen und die Leistung Ihres Algorithmus verbessern können. Da sich Entscheidungsbäume, einschließlich Baumensembles, jedoch in der Regel recht schnell trainieren lassen, können Sie diese Schleife schneller durchlaufen und möglicherweise die Leistung Ihres Lernalgorithmus effizienter verbessern. Schließlich können kleine Entscheidungsbäume möglicherweise von Menschen interpretiert werden. Wenn Sie nur einen einzelnen Entscheidungsbaum trainieren und dieser Entscheidungsbaum beispielsweise nur ein paar Dutzend Notizen enthält, können Sie möglicherweise einen Entscheidungsbaum ausdrucken, um genau zu verstehen, wie er Entscheidungen trifft. Ich denke, dass die Interpretierbarkeit von Entscheidungsbäumen manchmal etwas überbewertet wird, denn wenn man ein Ensemble aus 100 Bäumen erstellt und jeder dieser Bäume Hunderte von Knoten hat, wird es schwierig und möglicherweise notwendig, sich dieses Ensemble anzusehen, um herauszufinden, was es tut einige separate Visualisierungstechniken. Aber wenn Sie einen kleinen Entscheidungsbaum haben, können Sie ihn tatsächlich betrachten und sehen, oh, es geht darum, zu klassifizieren, ob etwas ein Schnitt ist, indem bestimmte Merkmale auf bestimmte Weise betrachtet werden. Wenn Sie sich für die Verwendung eines Entscheidungsbaums oder eines Baumensembles entschieden haben, würde ich wahrscheinlich XGBoost für die meisten Anwendungen verwenden, an denen ich arbeiten werde. Ein

kleiner Nachteil eines Baumensembles besteht darin, dass es etwas teurer ist als ein einzelner Entscheidungsbaum. Wenn Sie ein sehr begrenztes Rechenbudget hätten, könnten Sie einen einzelnen Entscheidungsbaum verwenden, aber abgesehen von dieser Einstellung würde ich fast immer ein Baumensemble verwenden und insbesondere XGBoost verwenden. Wie wäre es mit neuronalen Netzen? Im Gegensatz zu Entscheidungsbäumen und Baumensembles funktioniert es gut für alle Arten von Daten, einschließlich tabellarischer oder strukturierter Daten sowie unstrukturierter Daten. Sowie gemischte Daten, die sowohl strukturierte als auch unstrukturierte Komponenten enthalten. Während bei tabellarisch strukturierten Daten häufig sowohl neuronale Netze als auch Entscheidungsbäume konkurrieren, ist bei unstrukturierten Daten wie Bildern, Videos, Audio und Text tatsächlich ein neuronales Netz der bevorzugte Algorithmus und nicht der Entscheidungsbaum oder ein Baumensemble. Der Nachteil besteht jedoch darin, dass neuronale Netze möglicherweise langsamer sind als ein Entscheidungsbaum. Das Training eines großen neuronalen Netzwerks kann einfach lange dauern. Zu den weiteren Vorteilen neuronaler Netze gehört, dass sie mit Transfer-Learning arbeiten. Dies ist wirklich wichtig, da uns für viele Anwendungen nur ein kleiner Datensatz zur Verfügung steht, sodass wir Transfer-Learning nutzen und ein Vortraining an einem viel größeren Datensatz durchführen können, was für die Wettbewerbsfähigkeit von entscheidender Bedeutung ist Leistung. Wenn Sie schließlich ein System aus mehreren zusammenarbeitenden maschinellen Lernmodellen aufbauen, ist es möglicherweise einfacher, mehrere neuronale Netze aneinanderzureihen und zu trainieren als mehrere Entscheidungsbäume. Die Gründe dafür sind recht technischer Natur und Sie brauchen sich im Rahmen dieses Kurses darüber keine Gedanken zu machen. Aber es hängt damit zusammen, dass Sie, selbst wenn Sie mehrere neuronale Netze aneinanderreihen, sie alle gemeinsam mithilfe des Gradientenabstiegs trainieren können. Bei Entscheidungsbäumen hingegen kann jeweils nur ein Entscheidungsbaum trainiert werden. Das ist es. Sie haben das Ende der Videos für diesen Kurs zu fortgeschrittenen Lernalgorithmen erreicht. Vielen Dank, dass Sie mir die ganze Zeit treu geblieben sind, und herzlichen Glückwunsch, dass Sie das Ende der Videos zu fortgeschrittenen Lernalgorithmen erreicht haben. Sie haben nun gelernt, wie man sowohl neuronale Netze als auch Entscheidungsbäume aufbaut und nutzt, und haben außerdem von einer Reihe von Tipps und praktischen Ratschlägen gehört, wie Sie diese Algorithmen für sich optimal einsetzen können . Aber selbst wenn alles, was Sie über überwachtes Lernen gesehen haben, nur ein Teil dessen ist, was Lernalgorithmen leisten können. Für überwachtes Lernen sind in Ihrem Trainingssatz gekennzeichnete Datensätze mit der Bezeichnung „Y“ erforderlich. Es gibt eine weitere Reihe sehr leistungsfähiger Algorithmen, die als unüberwachte Lernalgorithmen bezeichnet werden und bei denen Sie nicht einmal die Bezeichnung Y benötigen, damit der Algorithmus sehr interessante Muster herausfinden und mit den vorhandenen Daten Dinge anstellen kann. Ich freue mich darauf, Sie auch im dritten und letzten Kurs dieser Spezialisierung zu sehen, der sich mit unbeaufsichtigtem Lernen befassen sollte. Bevor Sie nun mit diesem Kurs fertig sind, wünsche ich Ihnen viel Spaß beim Üben der Ideen von Entscheidungsbäumen in ihren Übungsquiz und in ihren Übungslaboren. Ich wünsche Ihnen viel Glück in den Übungslaboren oder denjenigen unter Ihnen, die vielleicht Star Wars-Fans sind, lassen Sie mich sagen, möge der Wald mit Ihnen sein. Sowohl Entscheidungsbäume, einschließlich Baumensembles als auch neuronale Netze sind sehr leistungsstarke und sehr effektive Lernalgorithmen. Wann sollten Sie sich für das eine oder andere entscheiden? Schauen wir uns einige der Vor- und Nachteile jedes einzelnen an. Entscheidungsbäume und Baumensembles eignen sich oft gut für tabellarische Daten, auch strukturierte Daten genannt. Das heißt, wenn Ihr Datensatz wie eine riesige Tabelle aussieht, sind Entscheidungsbäume eine Überlegung wert. Beispielsweise verfügten wir in der Anwendung zur Vorhersage von Immobilienpreisen über einen Datensatz mit Merkmalen, die der Größe des Hauses, der Anzahl der Schlafzimmer, der Anzahl der Stockwerke und dem Alter des Hauses entsprachen. Diese Art von Daten, die in einer Tabelle mit entweder kategorialen oder kontinuierlich bewerteten Merkmalen gespeichert werden und sowohl zur

Klassifizierung als auch für Regressionsaufgaben dienen, bei denen Sie versuchen, eine diskrete Kategorie oder eine Zahl vorherzusagen. All diese Probleme können mit Entscheidungsbäumen gut gelöst werden. Im Gegensatz dazu werde ich die Verwendung von Entscheidungsbäumen und Baumensembles für unstrukturierte Daten nicht empfehlen. Dabei handelt es sich um Daten wie Bilder, Videos, Audiodateien und Texte, die Sie wahrscheinlich nicht in einem Tabellenkalkulationsformat speichern. Wie wir gleich sehen werden, funktionieren neuronale Netze tendenziell besser für unstrukturierte Datenaufgaben. Ein großer Vorteil von Entscheidungsbäumen und Baumensembles besteht darin, dass sie sehr schnell trainiert werden können. Vielleicht erinnern Sie sich an dieses Diagramm aus der Vorwoche, in dem wir über die iterative Schleife der maschinellen Lernentwicklung gesprochen haben. Wenn das Training Ihres Modells viele Stunden dauert, schränkt dies die Geschwindigkeit ein, mit der Sie diese Schleife durchlaufen und die Leistung Ihres Algorithmus verbessern können. Da sich Entscheidungsbäume, einschließlich Baumensembles, jedoch in der Regel recht schnell trainieren lassen, können Sie diese Schleife schneller durchlaufen und möglicherweise die Leistung Ihres Lernalgorithmus effizienter verbessern. Schließlich können kleine Entscheidungsbäume möglicherweise von Menschen interpretiert werden. Wenn Sie nur einen einzelnen Entscheidungsbaum trainieren und dieser Entscheidungsbaum beispielsweise nur ein paar Dutzend Notizen enthält, können Sie möglicherweise einen Entscheidungsbaum ausdrucken, um genau zu verstehen, wie er Entscheidungen trifft. Ich denke, dass die Interpretierbarkeit von Entscheidungsbäumen manchmal etwas überbewertet wird, denn wenn man ein Ensemble aus 100 Bäumen erstellt und jeder dieser Bäume Hunderte von Knoten hat, wird es schwierig und möglicherweise notwendig, sich dieses Ensemble anzusehen, um herauszufinden, was es tut einige separate Visualisierungstechniken. Aber wenn Sie einen kleinen Entscheidungsbaum haben, können Sie ihn tatsächlich betrachten und sehen, oh, es geht darum, zu klassifizieren, ob etwas ein Schnitt ist, indem bestimmte Merkmale auf bestimmte Weise betrachtet werden. Wenn Sie sich für die Verwendung eines Entscheidungsbaums oder eines Baumensembles entschieden haben, würde ich wahrscheinlich XGBoost für die meisten Anwendungen verwenden, an denen ich arbeiten werde. Ein kleiner Nachteil eines Baumensembles besteht darin, dass es etwas teurer ist als ein einzelner Entscheidungsbaum. Wenn Sie ein sehr begrenztes Rechenbudget hätten, könnten Sie einen einzelnen Entscheidungsbaum verwenden, aber abgesehen von dieser Einstellung würde ich fast immer ein Baumensemble verwenden und insbesondere XGBoost verwenden. Wie wäre es mit neuronalen Netzen? Im Gegensatz zu Entscheidungsbäumen und Baumensembles funktioniert es gut für alle Arten von Daten, einschließlich tabellarischer oder strukturierter Daten sowie unstrukturierter Daten. Sowie gemischte Daten, die sowohl strukturierte als auch unstrukturierte Komponenten enthalten. Während bei tabellarisch strukturierten Daten häufig sowohl neuronale Netze als auch Entscheidungsbäume konkurrieren, ist bei unstrukturierten Daten wie Bildern, Videos, Audio und Text tatsächlich ein neuronales Netz der bevorzugte Algorithmus und nicht der Entscheidungsbaum oder ein Baumensemble. Der Nachteil besteht jedoch darin, dass neuronale Netze möglicherweise langsamer sind als ein Entscheidungsbaum. Das Training eines großen neuronalen Netzwerks kann einfach lange dauern. Zu den weiteren Vorteilen neuronaler Netze gehört, dass sie mit Transfer-Learning arbeiten. Dies ist wirklich wichtig, da uns für viele Anwendungen nur ein kleiner Datensatz zur Verfügung steht, sodass wir Transfer-Learning nutzen und ein Vortraining an einem viel größeren Datensatz durchführen können, was für die Wettbewerbsfähigkeit von entscheidender Bedeutung ist Leistung. Wenn Sie schließlich ein System aus mehreren zusammenarbeitenden maschinellen Lernmodellen aufbauen, ist es möglicherweise einfacher, mehrere neuronale Netze aneinanderzureihen und zu trainieren als mehrere Entscheidungsbäume. Die Gründe dafür sind recht technischer Natur und Sie brauchen sich im Rahmen dieses Kurses darüber keine Gedanken zu machen. Aber es hängt damit zusammen, dass Sie, selbst wenn Sie mehrere neuronale Netze aneinanderreihen, sie alle gemeinsam mithilfe des Gradientenabstiegs trainieren können. Bei Entscheidungsbäumen hingegen kann jeweils nur ein

Entscheidungsbaum trainiert werden. Das ist es. Sie haben das Ende der Videos für diesen Kurs zu fortgeschrittenen Lernalgorithmen erreicht. Vielen Dank, dass Sie mir die ganze Zeit treu geblieben sind, und herzlichen Glückwunsch, dass Sie das Ende der Videos zu fortgeschrittenen Lernalgorithmen erreicht haben. Sie haben nun gelernt, wie man sowohl neuronale Netze als auch Entscheidungsbäume aufbaut und nutzt, und haben außerdem von einer Reihe von Tipps und praktischen Ratschlägen gehört, wie Sie diese Algorithmen für sich optimal einsetzen können. Aber selbst wenn alles, was Sie über überwachtes Lernen gesehen haben, nur ein Teil dessen ist, was Lernalgorithmen leisten können. Für überwachtes Lernen sind in Ihrem Trainingssatz gekennzeichnete Datensätze mit der Bezeichnung „Y“ erforderlich. Es gibt eine weitere Reihe sehr leistungsfähiger Algorithmen, die als unüberwachte Lernalgorithmen bezeichnet werden und bei denen Sie nicht einmal die Bezeichnung Y benötigen, damit der Algorithmus sehr interessante Muster herausfinden und mit den vorhandenen Daten Dinge anstellen kann. Ich freue mich darauf, Sie auch im dritten und letzten Kurs dieser Spezialisierung zu sehen, der sich mit unbeaufsichtigtem Lernen befassen sollte. Bevor Sie nun mit diesem Kurs fertig sind, wünsche ich Ihnen viel Spaß beim Üben der Ideen von Entscheidungsbäumen in ihren Übungsquiz und in ihren Übungslaboren. Ich wünsche Ihnen viel Glück in den Übungslaboren oder denjenigen unter Ihnen, die vielleicht Star Wars-Fans sind, lassen Sie mich sagen: Möge der Wald mit Ihnen sein. [ MUSIK ]

## Andrew Ng und Chris Manning über die Verarbeitung natürlicher Sprache

Hallo, ich freue mich, mit meinem alten Freund und Mitarbeiter, Professor Chris Manning, hier zu sein. Chris hat eine sehr lange und beeindruckende Biografie, aber um es kurz zu machen: Er ist Professor für Informatik an der Stanford University und außerdem Direktor des Stanford AI Lab. Und er gilt auch als der am häufigsten zitierte Forscher im Bereich NLP oder Verarbeitung natürlicher Sprache. Also, wirklich schön, hier bei dir zu sein, Chris. >> Schön, die Gelegenheit zu haben, mit Andrew zu plaudern. >> Wir kennen uns also seit vielen Jahren und haben zusammengearbeitet. Ein interessanter Teil Ihres Hintergrunds war für mich immer, dass Sie, obwohl Sie heute ein angesehener Forscher im Bereich maschinelles Lernen im NLP sind, eigentlich in einem ganz anderen Bereich angefangen haben. Wenn ich mich recht erinnere, haben Sie in Linguistik promoviert und sich mit der Syntax von Sprachen beschäftigt. Wie sind Sie vom Studium der Syntax zum NLP-Forscher gekommen? >> Das kann ich Ihnen durchaus sagen, aber ich möchte auch darauf hinweisen, dass ich eigentlich immer noch Professor für Linguistik bin. Ich habe einen gemeinsamen Termin in Stanford. Und ab und zu unterrichte ich tatsächlich immer noch echte Linguistik und computergestützte Verarbeitung natürlicher Sprache. Von Anfang an interessierte ich mich sehr für menschliche Sprachen und dafür, wie sie funktionieren, wie Menschen sie verstehen und wie sie erlernt werden. Ich hatte also diese Art von Anziehungskraft, ich sah diese Anziehungskraft in menschlichen Sprachen. Aber das hat mich auch dazu gebracht, über Ideen nachzudenken, die wir heute eher als maschinelles Lernen oder rechnerische Ideen betrachten. Zwei der zentralen Ideen der menschlichen Sprache: Wie erlernen kleine Kinder die menschliche Sprache? Und was die Erwachsenen betrifft, nun ja, wir reden jetzt einfach miteinander und verstehen uns ziemlich gut. Und es ist tatsächlich erstaunlich, wie wir das schaffen. Welche Art der Verarbeitung ermöglicht das? Und so wurde mein Interesse schon früh geweckt, mich mit maschinellem Lernen zu befassen. Tatsächlich habe ich, noch bevor ich es in die Graduiertenschule geschafft hatte, in kleinen Schritten damit begonnen, maschinelles Lernen zu erlernen, das aus diesen Interessen resultierte. >> Dass die gesamte menschliche Sprache gelernt wurde, wir irgendwann in unserem Leben gelernt hatten, Englisch zu sprechen, und wenn wir an einem anderen Ort aufgewachsen wären, hätten wir eine völlig andere Sprache gelernt. Es ist erstaunlich, wie Menschen das tun, und jetzt lernen vielleicht auch Maschinen Sprache. Aber erzählen Sie uns einfach mehr über Ihre Reise. Sie hatten also einen

Doktortitel in Linguistik und wie kam es dann dazu? >> Davor gibt es also auch einiges. Ich meine, als ich noch im Grundstudium war, habe ich offiziell drei Hauptfächer belegt. Dies war in Australien, einer in Mathematik, einer in Informatik und einer in Linguistik. Jetzt bekommen die Leute ein etwas übertriebenes Gefühl dafür, was das bedeutet, wenn man sich in einem amerikanischen Kontext befindet, weil es meiner Meinung nach unmöglich wäre, drei Hauptfächer und den Bachelor in Stanford zu absolvieren. Aber eigentlich habe ich als Student ein Kunststudium gemacht, damit ich machen konnte, was ich wollte, zum Beispiel Linguistik. Man musste zwei Hauptfächer absolvieren, um das Kunststudium abzuschließen. Für die USA war es also eher ein Doppelstudium. >> Sie wissen das wahrscheinlich nicht über mich, aber Mellon, ich habe eigentlich ein Dreifachstudium absolviert, das war einmal Statistik und Wirtschaft. Okay, wir sind beide Triple-Major-Kollegen. >> Ja, jedenfalls hatte ich Hintergrundwissen und Interesse daran, etwas mit Informatik zu tun. Meine Interessen waren also irgendwie gemischt, und ich meine, als ich mich an Graduiertenschulen beworben habe, war einer der Orte, an denen ich mich beworben habe, Carnegie Mellon, weil sie stark in Computerlinguistik waren . Und wenn ich dorthin gegangen wäre, wäre ich als Informatik-Student eingeschrieben worden, aber am Ende bin ich als Linguistik-Student gelandet, weil es zu dieser Zeit in der Informatik-Abteilung noch keine Verarbeitung natürlicher Sprache gab. Aber ich war immer noch daran interessiert, Ideen zur Verarbeitung natürlicher Sprache zu verfolgen. Doch zu diesem Zeitpunkt, Anfang der Neunzigerjahre, begannen sich die Dinge gerade zu ändern. Der Großteil der Verarbeitung natürlicher Sprache bestand jedoch aus regelbasierten logischen deklarativen Systemen. Aber es war auch in jenen Jahren, Anfang der 1990er Jahre, als man begann, große Mengen menschlicher Sprachmaterialien, Texte und Sprache digital verfügbar zu machen. Das war also eigentlich kurz bevor das World Wide Web explodierte. Aber es handelte sich bereits um Dinge wie juristische Materialien, Zeitungsartikel und parlamentarisches SARS, wo man zuletzt Millionen von Wörtern der menschlichen Sprache in die Hände bekommen konnte. Und es schien einfach ganz klar, dass es spannende Dinge geben musste, die man machen konnte, indem man empirisch mit viel menschlicher Sprache arbeitete. Und das hat mich wirklich dazu gebracht, mich mit einer neuen Art der Verarbeitung natürlicher Sprache zu beschäftigen, die dann zu meiner späteren Karriere geführt hat. >> Es hört sich so an, als ob Ihre Karriere ursprünglich eher in der Linguistik verlief und sich mit dem Aufkommen von Daten, maschinellem Lernen und empirischen Methoden zu NLP, maschinellem Lernen und NLP verlagerte. >> Ja, ich meine, es hat sich auf jeden Fall verändert, und ich habe mich auf jeden Fall viel mehr auf die Verarbeitung natürlicher Sprache und Modelle für maschinelles Lernen verlagert. Aber bis zu einem gewissen Grad hat sich das Gleichgewicht verändert. Aber damit beschäftige ich mich schon seit einiger Zeit, eigentlich ging es als Student meiner Abschlussarbeit darum, die Formen von Wörtern zu lernen. Das ist zu einem bekannten Problem beim Erlernen früherer englischer Verben und der frühen Verbindungsliteratur geworden. Und ich habe versucht, Paradigmen für Verbformen zu lernen. Und ich lernte Regeln für die verschiedenen Formen mithilfe des C 4.5-Entscheidungsbaum-Lernalgorithmus. [LACHEN] Wenn Sie sich daran erinnern. >> Ja, richtig, gute Zeiten. Ja, und es ist überraschenderweise nicht intuitiv, oder? Ich weiß nicht, wie der Übergang vom Präsens zum Präteritum und allen anderen Sonderfällen sein kann. >> Ja. >> Ja, also haben wir viel über die NLP-Verarbeitung natürlicher Sprache gesprochen. Können Sie für einige der Lernenden, die zum ersten Mal mit maschinellem Lernen beginnen, sagen, was NLP ist ? >> Sicher, absolut, ja. NLP steht also für die Verarbeitung natürlicher Sprache. Ein anderes Wort oder ein Begriff, der manchmal dafür verwendet wird, ist Computerlinguistik, es ist dasselbe. Ich meine, die Verarbeitung natürlicher Sprache ist eigentlich ein seltsamer Begriff, oder? Das bedeutet also, dass wir Dinge mit menschlichen Sprachen machen. Sie müssen also davon ausgehen, dass Sie ein ausreichender Informatiker sind, sodass Sie, wenn Sie Sprache sagen, in der Programmiersprache Ihres Gehirns denken. Und deshalb muss man „natürliche Sprache“ sagen, um zu meinen, dass man über die Sprachbilder spricht, die Menschen verwenden. Insgesamt bedeutet die Verarbeitung natürlicher Sprache also, alles Intelligente mit menschlichen Sprachen zu tun. In gewisser Hinsicht

lässt sich das also aufteilen in das Verstehen menschlicher Sprachen, die Produktion menschlicher Sprachen und den Erwerb menschlicher Sprachen, obwohl die Menschen oft auch im Hinblick auf verschiedene Anwendungen darüber nachdenken. Und dann denken Sie vielleicht über Dinge wie maschinelle Übersetzung oder die Beantwortung von Fragen oder die Erstellung von Werbetexten oder Zusammenfassungen nach. Es gibt so viele verschiedene Aufgaben, an denen Menschen mit bestimmten Zielen arbeiten und bei denen man Dinge mit menschlicher Sprache erledigt. Und es gibt viel Verarbeitung natürlicher Sprache, weil so viel von dem, was die Welt auf unsere menschliche Welt einwirkt, in Form von menschlichem Sprachmaterial verarbeitet und übertragen wird. Also, wegen all dieser Anwendungen oder sogar der Websuche, oder? Die meisten von uns nutzen NLP. >> Ja. >> Viele, viele Male [UNVERSTÄNDLICH] >> Sie haben Recht, in gewisser Weise ist die Websuche die größte Anwendung natürlicher Sprache, oder? [LACHEN] Und das ist wirklich das große Problem, ich meine, traditionell war es eher ein einfaches, oder? In den guten alten Zeiten gab es zwar verschiedene Wartefaktoren und so weiter, aber es ging hauptsächlich um die Art der passenden Schlüsselwörter, dann um Ihre Suchbegriffe und dann um einige Faktoren für die Qualität der Seite. Es fühlte sich nicht wirklich wie ein Sprachverständnis an, aber das hat sich im Laufe der Jahre wirklich verändert. Wenn Sie heutzutage also einer Suchmaschine eine Frage stellen, wird Ihnen oft ein Antwortfeld angezeigt, in dem sie einen Text extrahiert hat und die ihrer Meinung nach richtige Antwort fett, farbig oder so ähnlich darstellt . Das ist dann diese Aufgabe des Beantwortens von Fragen und dann ist es wirklich eine Aufgabe zum Verstehen natürlicher Sprache. >> Ja, ja, und ich habe das Gefühl, dass es neben der Websuche vielleicht auch die ganz große ist, selbst wenn wir auf eine Online-Shopping-Website oder eine Film-Website gehen und dort eingeben, was wir wollen, und danach eine Website-Suche durchführen kleinere Website als die großen Suchmaschinen. Dabei werden zunehmend auch ausgefeilte NLP-Algorithmen eingesetzt und es wird ebenfalls ein großer Mehrwert geschaffen. Vielleicht ist es für Sie nicht das echte NLP, aber es scheint dennoch sehr wertvoll zu sein. >> Ich stimme zu, es ist sehr wertvoll. Und es gibt viele interessante Probleme auf jeder E-Commerce-Website, wobei die Suche tatsächlich sehr schwierig ist, wenn Leute die Art von Waren beschreiben, die sie wollen. Und Sie müssen versuchen, es mit den verfügbaren Produkten in Einklang zu bringen. Das ist, wie sich herausstellt, überhaupt kein einfaches Problem. >> Ja, das stimmt, ja. In den letzten, ich weiß nicht, ein paar Jahrzehnten hat NLP also einen großen Wandel durchgemacht, weg von den eher regelbasierten Techniken, auf die Sie gerade angespielt haben, hin zum weitaus umfassenderen Einsatz von wirklich maschinellem Lernen. Und Sie waren einer der Leute, die Teile dieses Angriffs anführten und jeden Schritt des Weges beobachteten. Sie erschufen einige der Stämme, während sie geschahen. Können Sie etwas über diesen Prozess und das, was Sie gesehen haben, sagen? >> Sicher, absolut. Ja, als ich als Student im Grundstudium anfing, wurde der Großteil der Verarbeitung natürlicher Sprache durch handgebaute Systeme durchgeführt, die auf verschiedene Weise Regeln und Inferenzverfahren nutzten, um sozusagen einen Pfad und ein Verständnis für einen Textabschnitt aufzubauen . >> Was ist ein Beispiel für ein Regel- oder Inferenzsystem? Eine Regel könnte also Teil der Struktur der menschlichen Sprache sein. Wie im Englischen besteht ein Satz normalerweise aus einer Subjekt-Nominalphrase, gefolgt von einem Verb und einer Objekt-Nominalphrase. Und das gibt Ihnen eine Vorstellung davon, wie Sie die Bedeutung des Satzes verstehen können. Aber es könnte auch etwas darüber aussagen, wie ein Wort zu interpretieren ist, sodass viele Wörter im Englischen sehr mehrdeutig sind. Aber wenn Sie so etwas wie das Wort „Stern“ haben und es im Zusammenhang mit einem Film steht, dann bezieht es sich wahrscheinlich auf einen Menschen auf diesem astronomischen Objekt. Und man hat damals versucht, solche Dinge mit solchen Regeln zu bewältigen. Heutzutage scheint es für uns nicht sehr wahrscheinlich, dass das funktioniert, aber früher war das ziemlich normal. Und erst als viel digitaler Text und Sprache verfügbar wurde, schien es wirklich so, als gäbe es eine andere Möglichkeit, stattdessen Statistiken über menschliche Sprache und Material zu berechnen und Modelle für maschinelles Lernen zu erstellen. Und das war das Erste, worauf ich mich etwa Mitte bis Ende der

1990er Jahre einließ. Der erste Bereich, in dem ich anfing, viel zu recherchieren, Artikel zu veröffentlichen und bekannt zu werden, war der Aufbau dessen, was wir früher oft als statistische Verarbeitung natürlicher Sprache bezeichneten. Später verschmolz es jedoch mit allgemein problematischen Ansätzen für künstliche Intelligenz und maschinelles Lernen. Und das hat uns, sagen wir mal, ungefähr bis ins Jahr 2010 geführt. Und ungefähr zu diesem Zeitpunkt begann das neue Interesse am Deep Learning mithilfe großer künstlicher neuronaler Netze zu beginnen. Für mein Interesse daran muss ich mich wirklich bei Andrew bedanken, denn zu diesem Zeitpunkt ist Andrew immer noch Vollzeit an der Stanford University und er war im Büro neben mir und er war wirklich begeistert von den neuen Dingen, die in diesem Bereich passierten Deep Learning, schätze ich. Jedem, der sein Büro betrat, sagte er, es sei irgendwie aufregend für das, was jetzt passiert, und ich bin in einem neuronalen Netzwerk, man muss anfangen, sich das anzuschauen. Und das war wirklich der Anstoß, der mich schon ziemlich früh dazu brachte, mich mit der Betrachtung von Dingen in neuronalen Netzen zu befassen. Ich hatte tatsächlich schon einiges davon gesehen, und als ich hier Student war, war Dave Rummelhardt an der Stanford University in Psychiatrie und ich hatte an seinem Kurs über neuronale Netze teilgenommen. Ich hatte also einiges davon gesehen, aber es war eigentlich nicht das, worauf ich mich im Rahmen meiner eigenen Forschung eingelassen hatte. Also- >> Das wusste ich nicht, danke, ja. >> Ja. >> Und dann haben wir schließlich gemeinsam einige Schüler betreut. >> Ja, absolut. >> Ich würde gerne den Aufstieg von Deep Learning und NLP hören. Was haben Sie gesehen, seit Sie in diesem Bereich tätig sind? >> Ja, ab etwa 2010 begannen die Studenten damit, die ersten Aufsätze und Deep Learning für NLP-Konferenzen zu verfassen. Es ist immer schwer, wenn man versucht, etwas Neues zu machen. Wir haben genau die gleichen Erfahrungen gemacht, die Menschen vor etwa 15 Jahren gemacht haben, als sie versuchten, statistisches NLP durchzuführen: Wenn es eine etablierte Vorgehensweise gibt, ist es wirklich schwierig, neue Ideen voranzutreiben. Einige unserer ersten Beiträge wurden also von Konferenzen abgelehnt und erschienen stattdessen auf Konferenzen zum maschinellen Lernen oder Deep-Learning-Workshops, aber sehr schnell begann sich das zu ändern und die Leute interessierten sich sehr für Ideen für neuronale Netze. Aber ich habe das Gefühl, dass sich die Periode der neuronalen Netzwerke, die praktisch um 2010 begann, in zwei Teile teilt, weil die erste Periode, sagen wir mal, im Grunde genommen bis 2018 dauert. Wir haben große Erfolge beim Aufbau neuer Netzwerke für alle gezeigt Arten von Aufgaben. Wir haben sie für syntaktisches Parsen und Stimmungsanalyse erstellt. Und was sonst, Alter? >> Beantwortung der Frage. Aber es war so, als hätten wir das Gleiche gemacht wie früher mit anderen Arten von Modellen für maschinelles Lernen, nur dass wir jetzt über ein besseres Modell für maschinelles Lernen verfügten. Und anstatt eine logistische Regression oder eine Support-Vektor-Maschine zu trainieren, haben wir immer noch die gleiche Art von Sentiment-Analyse-Aufgabe durchgeführt, aber jetzt machen wir es mit einem neuronalen Netzwerk. Wenn ich jetzt zurückblicke, denke ich, dass die größere Veränderung um das Jahr 2018 herum stattfand. Denn damals kam die Idee auf, nun ja, wir könnten einfach mit einer großen Menge menschlichen Sprachmaterials beginnen und große, selbstüberwachte Modelle bauen. Das waren damals Modelle wie [UNVERSTÄNDLICH] und GPTs und Nachfolgemodelle dazu. Und sie konnten sich durch die Wortvorhersage in einer riesigen Textmenge gewissermaßen dieses erstaunliche Wissen über menschliche Sprachen aneignen. Und ich denke wirklich, dass man das im Nachhinein als einen größeren Wendepunkt betrachten wird, an dem sich die Art und Weise, wie Dinge gemacht wurden, wirklich verändert hat. >> Ja, ich denke, es gibt diesen Trend dahingehend, dass große Sprachmodelle aus riesigen Datenmengen lernen. Ich glaube, schon im Vorfeld gab es eine Ihrer Forschungsarbeiten, die mich wirklich ein wenig umgehauen hat, nämlich ein Handschuhpapier. Also mit Worteinbettungen, bei denen man die Vektorzahlen lernt, um ein Wort mithilfe eines neuronalen Netzwerks darzustellen. Das war für mich ziemlich überwältigend. Und dann hat die Arbeit, die Sie geleistet haben, die Mathematik wirklich aufgeräumt und es so viel einfacher gemacht. Und dann erinnere ich mich, dass ich gesagt habe: Das ist alles, was ich tun muss. Und dann können Sie diese

wirklich überraschend detaillierten Darstellungen lernen. Der Computer lernt die Nuancen dessen, was Wörter bedeuten. >> Absolut. Ja, also sollte ich anderen ein wenig Anerkennung zollen. Andere Leute arbeiteten ebenfalls an ähnlichen Ideen, darunter Ja Weston und Kollegen bei Google. Aber die Handschuhwortvektoren sind eines der bekanntesten Wortvektorsysteme. Das haben diese Wortvektoren bereits getan. Ja, Sie haben Recht. Veranschaulichen Sie die Idee des selbstüberwachten Lernens dadurch, dass wir einfach riesige Textmengen genommen haben. Und dann könnten wir diese Modelle bauen, die enorm viel über die Bedeutung von Wörtern wissen. Es ist immer noch etwas, was ich den Leuten jedes Jahr in der ersten Vorlesung meines NLP-Kurses zeige. Weil es etwas Einfaches ist, aber tatsächlich so überraschend gut funktioniert. Sie können eine solche einfache Modellierung durchführen, bei der Sie versuchen, ein Wort anhand der Wörter im Kontext vorherzusagen, und indem Sie einfach die Mathematik des Lernens ausführen, um diese Vorhersagen zu treffen. Nun, Sie lernen all diese Dinge über Wortbedeutungen und können diese wirklich schönen Muster ähnlicher Wortbedeutungen oder Analogien von etwas erstellen. Bleistift ist die Zeichnung, ebenso wie Pinsel, und da steht „Malerei“, oder? Dass es sozusagen schon eine Menge erfolgreiches Lernen zeigt. Das war also der Vorläufer dessen, was dann mit Dingen wie Burton GPT zur nächsten Stufe weiterentwickelt wurde, wo es nicht nur um die Bedeutung einzelner Wörter ging. Sondern Bedeutungen ganzer Textteile und Kontexte. >> Ja, also fand ich es erstaunlich, dass man ein kleines neuronales Netzwerk oder ein Modell nehmen und ihm dann viele englische Sätze oder eine andere Sprache geben und das Wort verstecken kann. Bitten Sie es , das Wort vorherzusagen, das ich gerade getroffen habe, und so diese Analogien zu lernen . Und diese sehr tiefen, Ihrer Meinung nach wirklich tiefen Dinge hinter der Bedeutung des Wortes. Und dann 2018, vielleicht dieser andere Infektionspunkt, was geschah danach? >> Ja. Das war also im Jahr 2018 der Punkt, an dem eigentlich zwei Dinge passierten. Eine Sache ist, dass die Menschen, oder eigentlich im Jahr 2017, diese neue Architektur entwickelt haben. Das war viel besser auf moderne parallele GPUs skalierbar. Und das war die Transformatorarchitektur. Der zweite Teil davon war jedoch, dass die Leute es vielleicht wiederentdecken, weil ich den gleichen Trick wie beim Handschuhmodell verwendet habe, nämlich wenn man die Aufgabe hat, einfach ein Wort in einem gegebenen Kontext vorherzusagen. Entweder ein Kontext auf beiden Seiten oder die vorangehenden Wörter, die sich einfach als erstaunliche Lernaufgabe herausstellen. Und das überrascht viele Menschen. Und oft sieht man Diskussionen, in denen Leute sagen, dass es nichts Interessantes gibt, wenn man Dinge herabwürdigt. Dabei werden lediglich Statistiken erstellt, um vorherzusagen, welches Wort am wahrscheinlichsten nach den vorhergehenden Wörtern steht. Und ich denke, das wirklich Interessante ist, dass das wahr ist, aber es ist nicht wahr. Denn ja, die Aufgabe besteht darin, dass Sie das nächste Wort anhand der vorhergehenden Wörter vorhersagen. Aber das wirklich Interessante ist, wenn man diese Aufgabe wirklich so gut wie möglich erledigen möchte. Dann ist es tatsächlich hilfreich, den gesamten Rest des Satzes zu verstehen und zu wissen, wer was mit wem macht und was im Satz steht. Aber darüber hinaus hilft es auch, die Welt zu verstehen, denn wenn Ihr Text etwas in die Richtung der auf Fidschi verwendeten Währung geht, ist das so. Nun, Sie müssen über etwas Weltwissen verfügen, um zu wissen, wie die richtige Antwort darauf lautet. Und so lernen gute Modelle dabei, sowohl die Struktur von Sätzen und ihrer Bedeutung zu verfolgen als auch Fakten über die Welt zu kennen, damit sie Vorhersagen treffen können. Und deshalb wird daraus etwas, das manchmal als „KI-Abschlussaufgabe“ bezeichnet wird, oder? Das brauchst du wirklich. Es gibt nichts, was bei der Beantwortung der Frage „Welches Wort kommt als nächstes“ nicht wirklich nützlich sein kann, oder? Man kann im WM-Halbfinale gegen die Mannschaften antreten und man muss etwas über Fußball wissen [LACHEN], um die richtige Antwort zu geben. >> Ich vervollständige dieses lustige Konzept, oder? Ist die Idee, dass man dieses eine Problem lösen kann, man alles in der KI lösen kann oder eine Art Analogie zu NP-vollständigen Problemen aus der Computertheorie herstellen kann ? Was denken Sie? Glauben Sie, dass die Vorhersage des Knicks-Worts KI-vollständig ist? Ich selbst habe diesbezüglich sehr gemischte Gefühle . Ich kann sagen, ich glaube nicht, dass es wahr ist. Ich bin

gespannt, was Sie denken. Ich denke, dass das nicht ganz stimmt, weil ich glaube, dass es noch andere Dinge gibt, die Menschen schaffen, sich zurechtzufinden. Es gibt Menschen, die kluge Einblicke in die Mathematik haben, oder es gibt Menschen, die sich mit etwas beschäftigen, das viel mehr ist. Ein dreidimensionales Rätsel aus der realen Welt, bei dem es darum geht, herauszufinden, wie man etwas Mechanisches oder ähnliches macht. Und das ist einfach kein Sprachproblem. Aber andererseits denke ich auch, dass Sprache der Universalität näher kommt, als manche Leute denken, weil wir in dieser 3D-Welt leben. Und darin mit unserem Körper, unseren Gefühlen und anderen Lebewesen und Artefakten um uns herum agieren. Und man könnte denken, dass davon überhaupt nicht viel in der Sprache steht. Aber eigentlich ist fast alles, worüber wir nachdenken, reden und schreiben, in Sprache. Wir können die Positionen von Dingen zueinander in der Sprache beschreiben. So werden überraschend viele andere Teile der Welt in der sprachlichen Reflexion gesehen. Und deshalb lernen Sie auch alles über sie kennen. Wenn Sie etwas über den Sprachgebrauch lernen. >> Man lernt einen Aspekt von vielen Dingen kennen, auch wenn Dinge nicht wirklich möglich sind . >> Man lernt nicht wirklich, wie man Fahrrad fährt, [LACHEN], aber man lernt einige Aspekte dessen, was dazugehört und die man ausbalancieren muss. Und man muss die Füße auf den Pedalen haben und sie treten und so weiter. Ja. >> Und angesichts dieses Trends in LOP waren die letzten Sprachmodelle in den letzten Jahren sehr aufregend. Was denken Sie darüber, wohin das alles führen wird? >> Nun ja, es war einfach unglaublich. Erfolgreich und aufregend, oder? Wir haben also nicht wirklich alle Details erklärt, oder? In der ersten Phase des Erlernens dieser großen Sprachmodelle besteht die Aufgabe lediglich darin, das nächste Wort vorherzusagen. Und das machen Sie milliardenfach bei einem sehr großen Textstück. Und siehe da, Sie erhalten dieses große neuronale Netzwerk, das einfach ein wirklich nützliches Artefakt für alle Arten von Aufgaben zur Verarbeitung natürlicher Sprache ist. Aber dann muss man tatsächlich noch etwas damit machen, wenn man eine bestimmte Aufgabe erledigen möchte, sei es das Beantworten oder Zusammenfassen von Fragen oder das Erkennen toxischer Inhalte in sozialen Medien oder so etwas in der Art. Und an diesem Punkt gibt es eine Auswahl an Dingen, die Sie damit machen können. Die traditionelle Antwort war dann, dass Sie eine bestimmte Aufgabe hatten, sagen wir, es ginge darum, toxische Kommentare in sozialen Medien zu erkennen. Und Sie würden dafür einige überwachte Daten nehmen und dann das Sprachmodell verfeinern, um diese Klassifizierungsaufgabe zu beantworten. Aber diese Basis dieses großen selbstüberwachten Modells hat Ihnen enorm geholfen, denn sie bedeutete, dass das Modell über enorme Sprachkenntnisse verfügte und sehr schnell verallgemeinern konnte. Anders als in den alten Zeiten des überwachten Lernens, wo es so war, wenn Sie mir 10.000 beschriftete Beispielbeispiele geben, kann ich vielleicht ein einigermaßen anständiges Modell für Sie erstellen. Aber wenn Sie mir 50.000 beschriftete Beispiele geben, wird es viel besser sein. Es hat es irgendwie in diese Welt verwandelt. Nun, wenn Sie mir 100 beschriftete Beispiele geben und ich ein großes Sprachmodell verfeinere, kann ich viel besser abschneiden, als ich es mit den 50.000 Beispielen in der alten Welt geschafft hätte. Einige der neueren spannenden Arbeiten gehen jetzt sogar darüber hinaus, nun ja, vielleicht muss man das Modell überhaupt nicht mehr verfeinern. Daher haben die Leute viel mit Methoden gearbeitet, die manchmal auch als Aufforderungen oder Anweisungen bezeichnet werden und bei denen man einfach in natürlicher Sprache, vielleicht mit Beispielen, vielleicht mit expliziten Anweisungen, dem Modell einfach sagen kann, was es tun soll, und es tut, was selbst als jemand, der seit 30 Jahren in der Verarbeitung natürlicher Sprache arbeitet. Ich meine, es ist wirklich umwerfend, wie gut das funktioniert. Ich glaube, vor einem Jahrzehnt hätte ich nicht gedacht, dass wir dem Modell jetzt einfach sagen könnten, ich möchte, dass Sie diesen Text hier zusammenfassen sie werden es dann zusammenfassen. Ich finde das unglaublich. Ja, wir befinden uns in dieser sehr aufregenden Zeit, in der sich viele neue Fähigkeiten der natürlichen Sprache entfalten. Ich denke, es besteht überhaupt kein Zweifel daran, dass die Zukunft in den nächsten Jahren äußerst rosig ist, da die Menschen unterschiedliche Dinge und unterschiedliche Vorgehensweisen erarbeiten und beginnen, sich in unterschiedlichen Anwendungsbereichen zu

bewerben. Die Art von Fähigkeiten, die durch die jüngsten technologischen Entwicklungen freigeschaltet wurden. Es stellt sich in der Technik immer die Frage, ob die Kurve weiterhin steil nach oben geht oder ob es dann neue Dinge gibt, wir müssen herausfinden, wie das geht. >> Es geht schon eine ganze Weile bergauf. Hoffentlich ist eine Extrapolation immer gefährlich. Aber, aber wir werden sehen, ich bin nur neugierig, wissen Sie, Sie haben das Schreiben von Eingabeaufforderungen für das MRP-System, das große Sprachmodell, erwähnt, was Sie wollen, und es scheint es auf magische Weise zu tun. Ich bin neugierig, glauben Sie, dass Prompt Engineering der Weg der Zukunft ist? Wenn ich diese Prompts schreibe, merke ich manchmal, dass es auf wundersame Weise funktioniert, und manchmal ist es frustrierend, meine Anweisungen umzuformulieren, um den Wortlaut so anzupassen, dass er genau richtig ist das gewünschte Ergebnis erzeugen. Glauben Sie also, dass Problem-Engineering der Weg der Zukunft ist, oder glauben Sie, dass es ein Zwischen-Hack ist, bis jemand einen besseren Weg erfindet, diese zu kontrollieren, die Ausgänge dieser Systeme zu kontrollieren? >> Ich denke, es ist beides, ich denke, es wird der Weg der Zukunft sein, aber ich denke auch, dass die Leute im Moment viel herumhacken und umformulieren, um zu versuchen, die Dinge besser zum Laufen zu bringen, und mit etwas Glück mit einem noch ein paar Jahre der Entwicklung, die allmählich verschwinden werden. Ich meine, eine Möglichkeit, über den Unterschied nachzudenken, ist der Vergleich mit der Art von Sprachunterstützung oder virtueller Unterstützung, die heutzutage auf Telefonlautsprechern wie Amazon oder Alexa verfügbar ist, oder? Ich meine, ich denke, wir alle haben die Erfahrung gemacht, dass diese Geräte nicht immer großartig sind, aber wenn man weiß, wie man Dinge richtig ausdrückt, wird es etwas bewirken. Wenn Sie jedoch die falsche Formulierung verwenden, wird dies nicht der Fall sein, und der Unterschied zu Menschen besteht im Großen und Ganzen darin, dass Sie darüber nicht nachdenken müssen. Sie können sagen, was Sie wollen, und es spielt keine Rolle, welche Worte Sie wählen, sie werden den anderen Menschen ansprechen, vorausgesetzt, dass jemand, der dieselbe Sprache spricht usw., Sie versteht und tut, was Sie wollen. Und ich denke und hoffe, dass wir bei diesen Modellen die gleichen Fortschritte sehen werden, dass das Herumspielen mit der jeweiligen Formulierung, die Sie verwenden, einen großen Unterschied in der Funktionsweise machen kann. Aber hoffentlich wird das in ein paar Jahren nicht mehr wahr sein, Sie können andere Formulierungen verwenden und es wird immer noch funktionieren. Aber die Grundidee ist, dass wir in ein Zeitalter eintreten, in dem tatsächlich die menschliche Sprache als Befehlssprache verwendet werden kann, um Ihrem Computer zu sagen, was er tun soll. Anstatt also Menüs und Optionsfelder usw. verwenden oder Python-Code schreiben zu müssen, können Sie anstelle dieser Dinge sagen, was Sie wollen, und der Computer erledigt es. Ich denke, dass sich vor uns ein Zeitalter auftut, das weiter wachsen und enorme Veränderungen bewirken wird. >> Es fühlt sich an, als hätte man einen langen Weg zurückgelegt, aber es liegt noch viel vor uns und noch viel mehr vor uns. >> Ja, absolut. >> Was die Entwicklung der NLP-Technologie betrifft, ist das Einzige, was ich Sie fragen möchte, und ich vermute, dass Sie und ich diesbezüglich möglicherweise unterschiedliche Ansichten haben. In den letzten Jahrzehnten ging der Trend jedoch dahin, sich weniger auf regelbasiertes Engineering und mehr auf maschinelles Lernen auf Datenbasis zu verlassen. Manchmal blicken viele Daten in die Zukunft. Was halten Sie von dieser Mischung aus handcodierten Einschränkungen oder anderen Einschränkungen, expliziten Einschränkungen im Vergleich dazu, dass wir ein neues Netzwerk einrichten und viele Daten darauf werfen? Wo wird sich Ihrer Meinung nach das Gleichgewicht verschlechtern? >> Ich denke, es besteht kein Zweifel daran, dass das Lernen aus Daten der Weg in die Zukunft ist und was wir auch weiterhin tun werden. Aber ich denke, es gibt immer noch Raum für Modelle mit mehr Struktur und mehr induktiver Voreingenommenheit, die eine Art Grundlage für die Ausnutzung der Natur der Sprache haben. Das Modell, das in den letzten Jahren enorm erfolgreich war, ist das Transformer Neural Network und die Transformer Neural Networks, im Wesentlichen diese riesige Assoziationsmaschine. Es werden also Assoziationen von überall her aufgesaugt. >> Und schauen Sie sich zwei Wörter an und finden Sie heraus, welche Wörter zu welchem anderen Wort

gehören. >> Ja. Man nutzt also alles, um alles vorherzusagen, macht es immer wieder und bekommt alles, was man will. Und wissen Sie, das war unglaublich, unglaublich erfolgreich, aber es war unglaublich erfolgreich in dem Bereich, in dem es riesige Datenmengen gibt. Richtig, so dass diese Transformatormodelle für diese großen Sprachmodelle jetzt auf zig Milliarden Textwörtern trainiert werden. Als ich mit der statistischen Verarbeitung natürlicher Sprache begann. Und einige der traditionellen Linguisten beschwerten sich darüber, dass ich Statistiken aus 30 Millionen Wörtern von Newswire sammelte. Und ich habe ein Vorhersagemodell entwickelt und dachte, das sei einfach nicht das, worum es in der Linguistik geht. Ich hatte das Gefühl, eine vollkommen gute Antwort zu haben, nämlich dass ein menschliches Kind eine Sprache lernt. Sie sind tatsächlich mehr als 30 Millionen Wörtern an Daten ausgesetzt. Aber diese Art von Datenmenge, also die Art von Datenmenge, die wir verwendeten, waren vollkommen vernünftige Datenmengen, die es zu verwenden galt. Es geht nicht gerade darum, den menschlichen Spracherwerb zu modellieren. Aber darüber nachzudenken, wie wir für viele Daten etwas über Sprache lernen können. Aber diese modernen Transformatoren nutzen jetzt bereits mindestens zwei Größenordnungen mehr Daten. Und die meisten Leute denken, dass man die Dinge auf die nächste Ebene bringen kann, indem man noch mehr verwendet und es auf drei Größenordnungen anhebt. Und in gewisser Hinsicht war diese Expansionsstrategie äußerst effektiv. Ich kann es also niemandem verübeln, wenn er sagt: „Lasst uns die Automatisierung um eine weitere Größenordnung erweitern und sehen, was für erstaunliche Dinge wir tun können.“ Aber es zeigt auch, dass menschliches Lernen viel, viel besser ist, wenn es darum geht, viel mehr Informationen aus einer recht begrenzten Datenmenge zu extrahieren. Und an diesem Punkt kann man verschiedene Hypothesen aufstellen. Aber ich denke, es ist vernünftig anzunehmen, dass das menschliche Lernen in gewisser Weise auf die Struktur der Welt ausgerichtet ist. Und Dinge, die es in der Welt sieht und die es ihm ermöglichen, schneller aus weniger Daten zu lernen. >> Richtig, da stimme ich dir zu. Ich denke, dass das bessere Lernen unserer Räume, unser derzeitiges maschinelles Lernen, viel weniger effizient ist oder Daten viel weniger effizient nutzt. Es gibt also viel mehr Daten als jedes Kind. Und dann denke ich darüber nach, ob die verbesserten Lernalgorithmen auf sprachlichen Regeln basieren oder ob es nur Ingenieure sein werden. Entwicklung viel effizienterer Versionen des Transformators oder was auch immer danach kommt. Ich denke, das wird so sein. >> Das wird traditionell sein. Ich glaube nicht, dass es daran liegen wird, dass Leute explizit traditionelle Sprachregeln in das System integrieren. Ich glaube nicht, dass das der richtige Weg ist. Andererseits denke ich, dass wir allmählich beobachten, dass Modelle wie diese Transformatormodelle tatsächlich die Struktur der Sprache selbst entdecken, oder? Die weitreichenden Auswirkungen der menschlichen Sprache sind also, dass im Englischen das Subjekt vor dem Verb und das Objekt danach steht. Im Japanischen hingegen stehen die Verben am Ende des Satzes und Subjekt und Objekt normalerweise in dieser Reihenfolge davor. Aber es könnte in der anderen Reihenfolge tatsächlich sein, dass Transformatormodelle diese Fakten lernen. Sie können sie befragen und feststellen, dass sie diese Vorstellungen kennen, auch wenn ihnen nie explizit etwas über Subjekte und Objekte gesagt wurde. Ich denke, sie entdecken auch noch viel mehr über den Sprachgebrauch und -kontext, die Bedeutungen und Bedeutungen von Wörtern und darüber, was unangenehme Sprache ist und was nicht. Aber ein Teil dessen, was sie lernen, ist die gleiche Art von Struktur, die Linguisten als Struktur verschiedener menschlicher Sprachen beschrieben haben. >> Es ist also, als hätten Linguisten über viele Jahrzehnte hinweg bestimmte Dinge entdeckt. Und durch das Training mit Milliarden von Wörtern entdecken Transformatoren die gleichen Dinge, die Linguisten bei Menschen entdeckt haben. Das ist cool. Das alles sind also wirklich aufregende Fortschritte im NLP, die durch maschinelles Lernen und andere Dinge vorangetrieben werden. Für jemanden, der das Feld betritt, maschinelles Lernen oder KI oder NLP betritt. Es ist einfach viel los. Welchen Rat würden Sie jemandem geben, der in maschinelles Lernen einsteigen möchte? >> Ja. Nun, es ist ein guter Zeitpunkt, um einzusteigen. Ich denke, es besteht überhaupt kein Zweifel daran, dass wir uns noch in einem frühen Stadium befinden, in dem wir die Auswirkungen dieses neuen Ansatzes sehen,

bei dem die Software-Informatik auf der Grundlage einer viel stärkeren Nutzung effektiv neu erfunden wird maschinelles Lernen. Und die verschiedenen anderen Dinge, die sich daraus ergeben. Und ganz allgemein gibt es branchenübergreifend jede Menge Möglichkeiten für mehr Automatisierung. Ich nutze für mich oder in anderen Bereichen wie Vision und Robotik, den gleichen Dingen, mehr die Interpretation von menschlichem Sprachmaterial. Also viele Möglichkeiten. An diesem Punkt gibt es also offensichtlich viel zu tun, denn man möchte eine gute Grundlage schaffen, richtig. Sie kennen also einige der wichtigsten technischen Methoden des maschinellen Lernens und verstehen Ideen zur Erstellung von Modellen aus Daten. Schauen Sie sich Verluste an, führen Sie Schulungen durch, diagnostizieren Sie Fehler, all diese Kernthemen. Das ist insbesondere für die Verarbeitung natürlicher Sprache auf jeden Fall nützlich, einige dieser Fähigkeiten sind durchaus relevant. Aber dann gibt es bestimmte Arten von Modellen, die häufig verwendet werden, einschließlich des Transformators, über den wir heute viel gesprochen haben. Sie sollten auf jeden Fall über Transformatoren Bescheid wissen, denn tatsächlich werden sie zunehmend in allen anderen Bereichen des maschinellen Lernens sowie in der visuellen Bioinformatik eingesetzt, sogar in der Robotik werden mittlerweile Transformatoren eingesetzt. Aber darüber hinaus denke ich, dass es auch nützlich ist, etwas über die menschliche Sprache und die Natur der damit verbundenen Probleme zu lernen. Denn auch wenn die Menschen die Regeln der menschlichen Sprache nicht direkt in ihr Computersystem kodieren werden. Ein Gespür dafür, was in der Sprache passiert, worauf man achten muss und was man modellieren möchte, ist immer noch eine nützliche Fähigkeit. >> Und dann geht es um das Erlernen der Grundlagen und um das Erlernen dieser Konzepte. Sie sind mit einem sprachwissenschaftlichen Hintergrund in die KI eingestiegen und wir sehen jetzt Menschen aus allen Gesellschaftsschichten, die mit der Arbeit in der KI beginnen möchten. Was denken Sie über die Vorbereitung, die man haben sollte, oder darüber, wie man mit etwas anderem als Informatik oder KI beginnen kann? Es gibt also viele Orte, von denen Sie kommen und die Sie auf unterschiedliche Weise durchqueren können. Und wir sehen eine Menge Leute, die das tun, sie sind Leute, die in verschiedenen Bereichen angefangen haben, sei es Chemie, Physik oder noch viel weiter entfernte Bereiche. Und die Menschen haben in der Vergangenheit begonnen, sich mit maschinellem Lernen zu befassen. Ich denke, dass es da sozusagen zwei Ebenen der Antwort gibt. Eine der erstaunlichen Veränderungen besteht darin, dass es jetzt sehr gute Softwarepakete gibt, mit denen Sie Ihre Netzwerkmodelle bearbeiten können. Diese Software ist wirklich einfach zu bedienen. Sie müssen eigentlich nicht viele hochtechnische Dinge verstehen. Sie müssen eine fundierte Vorstellung davon haben, was die Idee des maschinellen Lernens ist. Und wie trainiere ich ein Modell und worauf sollte ich achten und auf die Zahlen, die ausgedruckt werden, um zu sehen, ob es funktioniert? Man muss aber eigentlich keinen höheren Abschluss haben, um diese Modelle bauen zu können. Ich meine, und was wir tatsächlich sehen, ist, dass sich viele Oberstufenschüler damit beschäftigen, weil es tatsächlich etwas ist, das man sich aneignen und umsetzen kann, wenn man über grundlegende Computerkenntnisse und ein wenig Programmierkenntnisse verfügt. Es ist einfach viel zugänglicher als viele Dinge, die sich mit KI befasst haben, sei es außerhalb der KI oder in anderen Bereichen wie Betriebssystemen oder Sicherheit. Aber wenn Sie auf eine tiefere Ebene vordringen und tatsächlich mehr darüber verstehen möchten, was vor sich geht. Ich denke, man kann nicht wirklich dorthin gelangen, wenn man nicht über bestimmte mathematische Grundlagen verfügt, zum Beispiel weil DeepLearning letzten Endes auf Infinitesimalrechnung basiert und man Funktionen optimieren muss. Und wenn man in diesem Bereich keinerlei Hintergrundwissen hat, endet das meiner Meinung nach irgendwann in einem Krieg. Also >> Das maschinelle Lernen in der Datenwissenschaft. Es ist für einige unserer Arbeiten praktisch. >> Ja. Ich glaube also, dass ich auf einer gewissen Ebene, wenn man sich im Hauptfach Geschichte oder nicht-mathematische Teile der Psychologie befindet, tatsächlich einen guten Freund habe, der, ja, er hat in der Graduiertenschule Infinitesimalrechnung gelernt, weil er Psychologe war, und das hatte er noch nie getan es vorher. Und beschloss, dass er anfangen wollte, etwas über diese neuen Arten von Modellen zu lernen, und dass es noch nicht zu spät war, an einem

Kuhkurs teilzunehmen. Und das tat er, richtig. Man muss also einiges davon wissen, aber für viele Leute, wenn sie etwas davon schon einmal gesehen haben, auch wenn man etwas eingerostet ist. Ich denke, man kann irgendwie wieder in die richtige Stimmung kommen, und es spielt keine Rolle, dass man sich als Student nicht mit KI oder maschinellem Lernen und solchen Dingen beschäftigt hat, man kann wirklich anfangen zu lernen, wie man diese Modelle baut und Dinge tun und wirklich, das ist meine eigene Geschichte, oder? Dass ich trotz der Tatsache, dass ich heutzutage an der Fakultät für Ingenieurwissenschaften in Stanford studieren darf, kein Ingenieur bin. Ich habe einen Doktortitel in Linguistik, den ich größtenteils von Kenntnissen in Mathematik und Linguistik und Programmierkenntnissen auf die Entwicklung von KI-Modellen umgestellt habe. >> Ich über etwas. Denken Sie, dass die verbesserten Bibliotheken und Abstraktionen, die jetzt verfügbar sind, wie Codierungs-Frameworks, wie der Tensorfluss von P Torch, aussehen? Glauben Sie, dass dadurch die Notwendigkeit verringert wird, die Analysis zu verstehen? Denn Junge, es ist schon eine Weile her, dass ich aufgrund der automatischen Differenzierung tatsächlich ein Derivat nehmen musste, um überhaupt eine neue Neuronetzwerkarchitektur zu implementieren oder zu erstellen >> Ja, ich meine, absolut. Ich meine, also in den frühen Tagen, als wir die Dinge sozusagen von 2010 bis 2015 machten, oder? Für jedes Modell, das wir gebaut haben, haben wir die Ableitungen von Hand ausgearbeitet und dann Code und was auch immer geschrieben. Manchmal war es Python, aber manchmal war es vielleicht Java oder C [LACHEN], um diese Ableitungen zu berechnen und zu überprüfen, ob wir sie richtig gemacht haben und so weiter, wobei man heutzutage eigentlich nichts davon mehr wissen muss, um DeepLearning-Modelle zu erstellen. Ich meine, das ist tatsächlich etwas, worüber ich nachdenke, sogar in Bezug auf meine eigene Verarbeitung natürlicher Sprache mit dem DeepLearning-Kurs, den ich unterrichte. Am Anfang beschäftigen wir uns noch mit der Matrizenrechnung und stellen sicher, dass die Leute etwas über Jacobian und ähnliches wissen, damit sie verstehen, was bei der Backpropagation, DeepLearning, gemacht wird. Aber es gibt so etwas, bei dem das bedeutet, dass wir ihnen einfach zwei Wochen lang die Hölle heiß machen. Es ist so etwas wie ein Bootcamp oder so etwas, das sie leiden lässt. Und dann sagen wir: „Aber Sie machen den Rest des Unterrichts mit Pytorch und sie müssen sozusagen nie wieder etwas davon wissen, oder?“ Es stellt sich immer die Frage, wie tief man in die technischen Grundlagen einsteigen möchte, oder? Du kannst doch weitermachen, oder? Zum Beispiel muss ein Informatiker im Jahr 2020 verstehen, was Elektronik und Transistoren sind oder was in Ihrer CPU passiert. Nun, es ist kompliziert, ich meine, in vielerlei Hinsicht ist es hilfreich, einiges davon zu wissen. Ich meine, ich weiß, Andrew, Sie waren einer der Pioniere, die maschinelles Lernen auf die GPU übertragen haben, und das bedeutet in gewisser Weise, dass Sie ein Gefühl dafür haben mussten, dass es da draußen diese neue Hardware gibt. Und es weist einige Eigenschaften der Parallelität auf, was bedeutet, dass es wahrscheinlich möglich ist, etwas Aufregendes zu tun. Daher ist es nützlich, etwas umfassenderes Wissen und Verständnis zu haben, und manchmal geht etwas kaputt, und wenn man tiefergehende Kenntnisse hat, kann man verstehen, warum es kaputt gegangen ist. Aber es gibt noch einen anderen Sinn: Die meisten Menschen müssen sich auf manche Dinge verlassen, und heutzutage können Sie das meiste, was Sie in der Modellierung neuronaler Netze tun möchten, auch ohne Kenntnisse in Analysis machen. >> Ja, ich denke, das ist ein toller Punkt. Ich habe das Gefühl, dass manchmal die Zuverlässigkeit der Abstraktion darüber entscheidet, wie oft man etwas reparieren muss, das kaputt ist. Tatsächlich ist mein Verständnis der Quantenphysik sehr schwach. Ich verstehe es kaum. Man könnte also argumentieren: Ich verstehe nicht, wie Computer funktionieren, weil Transistoren in der Quantenphysik gebaut sind. Aber zum Glück musste ich, wenn mit dem Transistor etwas schief ging, nie große Anstrengungen unternehmen, um es zu reparieren, [KREUZSPRECHEN], glaube ich. Und so denke ich, oder ein anderes Beispiel, die Sortierfunktion, die Bibliothek zum Sortieren von Dingen, und manchmal funktionieren sie tatsächlich nicht, richtig, Swap im Speicher oder was auch immer. Und wenn Sie wirklich verstehen, wie die Sortierfunktion funktioniert, können Sie das Problem beheben. Aber manchmal, wenn wir Abstraktionen, Bibliotheken und APIs haben, die zuverlässig

genug sind, verringert die Tatsache, dass diese Abstraktionen gut sind, die Notwendigkeit, einige der Dinge zu verstehen, die bei mir passieren. Es ist also eine aufregende Welt. Es fühlt sich an, als hätten wir Giganten, die auf den Schultern von Giganten aufbauen, und all diese Dinge werden von Monat zu Monat komplexer und aufregender. >> Ja, absolut. >> Vielen Dank, Chris, das war wirklich interessant und inspirierend und ich hoffe, dass alle, die das sehen, Chris seine eigene Reise zum Informatiker hören können. Und ein führender, vielleicht der führende NFP-Informatiker zu werden, sowie all diese aufregende Arbeit, die derzeit in NFP stattfindet. Ich hoffe, das inspiriert auch Sie, ins Wasser zu springen und es auszuprobieren. Es gibt einfach noch viel mehr Arbeit, die unsere Gemeinschaft gemeinsam erledigen muss. Ich denke, je mehr von uns daran arbeiten, desto besser wird es der Welt gehen. Vielen Dank, Chris. Es war wirklich toll, Sie hier zu haben. >> Vielen Dank, Andrew. Es hat Spaß gemacht, zu plaudern. [MUSIK][MUSIK] Hallo, ich freue mich, mit meinem alten Freund und Mitarbeiter, Professor Chris Manning, hier zu sein. Chris hat eine sehr lange und beeindruckende Biografie, aber um es kurz zu machen: Er ist Professor für Informatik an der Stanford University und außerdem Direktor des Stanford AI Lab. Und er gilt auch als der am häufigsten zitierte Forscher im Bereich NLP oder Verarbeitung natürlicher Sprache. Also, wirklich schön, hier bei dir zu sein, Chris. >> Schön, die Gelegenheit zu haben, mit Andrew zu plaudern. >> Wir kennen uns also seit vielen Jahren und haben zusammengearbeitet. Ein interessanter Teil Ihres Hintergrunds war für mich immer, dass Sie, obwohl Sie heute ein angesehener Forscher im Bereich maschinelles Lernen im NLP sind, eigentlich in einem ganz anderen Bereich angefangen haben. Wenn ich mich recht erinnere, haben Sie in Linguistik promoviert und sich mit der Syntax von Sprachen beschäftigt. Wie sind Sie vom Studium der Syntax zum NLP-Forscher gekommen? >> Das kann ich Ihnen durchaus sagen, aber ich möchte auch darauf hinweisen, dass ich eigentlich immer noch Professor für Linguistik bin. Ich habe einen gemeinsamen Termin in Stanford. Und ab und zu unterrichte ich tatsächlich immer noch echte Linguistik und computergestützte Verarbeitung natürlicher Sprache. Von Anfang an interessierte ich mich sehr für menschliche Sprachen und dafür, wie sie funktionieren, wie Menschen sie verstehen und wie sie erlernt werden. Ich hatte also diese Art von Anziehungskraft, ich sah diese Anziehungskraft in menschlichen Sprachen. Aber das hat mich auch dazu gebracht, über Ideen nachzudenken, die wir heute eher als maschinelles Lernen oder rechnerische Ideen betrachten. Zwei der zentralen Ideen der menschlichen Sprache: Wie erlernen kleine Kinder die menschliche Sprache? Und was die Erwachsenen betrifft, nun ja, wir reden jetzt einfach miteinander und verstehen uns ziemlich gut. Und es ist tatsächlich erstaunlich, wie wir das schaffen. Welche Art der Verarbeitung ermöglicht das? Und so wurde mein Interesse schon früh geweckt, mich mit maschinellem Lernen zu befassen. Tatsächlich habe ich, noch bevor ich es in die Graduiertenschule geschafft hatte, in kleinen Schritten damit begonnen, maschinelles Lernen zu erlernen, das aus diesen Interessen resultierte. >> Dass die gesamte menschliche Sprache gelernt wurde, wir irgendwann in unserem Leben gelernt hatten, Englisch zu sprechen, und wenn wir an einem anderen Ort aufgewachsen wären, hätten wir eine völlig andere Sprache gelernt. Es ist erstaunlich, wie Menschen das tun, und jetzt lernen vielleicht auch Maschinen Sprache. Aber erzählen Sie uns einfach mehr über Ihre Reise. Sie hatten also einen Doktortitel in Linguistik und wie kam es dann dazu? >> Davor gibt es also auch einiges. Ich meine, als ich noch im Grundstudium war, habe ich offiziell drei Hauptfächer belegt. Dies war in Australien, einer in Mathematik, einer in Informatik und einer in Linguistik. Jetzt bekommen die Leute ein etwas übertriebenes Gefühl dafür, was das bedeutet, wenn man sich in einem amerikanischen Kontext befindet, weil es meiner Meinung nach unmöglich wäre, drei Hauptfächer und den Bachelor in Stanford zu absolvieren. Aber eigentlich habe ich als Student ein Kunststudium gemacht, damit ich machen konnte, was ich wollte, zum Beispiel Linguistik. Man musste zwei Hauptfächer absolvieren, um das Kunststudium abzuschließen. Für die USA war es also eher ein Doppelstudium. >> Sie wissen das wahrscheinlich nicht über mich, aber Mellon, ich habe eigentlich ein Dreifachstudium absolviert, das war einmal Statistik und Wirtschaft. Okay, wir sind beide Triple-Major-Kollegen. >> Ja, jedenfalls hatte ich Hintergrundwissen und Interesse daran, etwas mit

Informatik zu tun. Meine Interessen waren also irgendwie gemischt, und ich meine, als ich mich an Graduiertenschulen beworben habe, war einer der Orte, an denen ich mich beworben habe, Carnegie Mellon, weil sie stark in Computerlinguistik waren . Und wenn ich dorthin gegangen wäre, wäre ich als Informatik-Student eingeschrieben worden, aber am Ende bin ich als Linguistik-Student gelandet, weil es zu dieser Zeit in der Informatik-Abteilung noch keine Verarbeitung natürlicher Sprache gab. Aber ich war immer noch daran interessiert, Ideen zur Verarbeitung natürlicher Sprache zu verfolgen. Doch zu diesem Zeitpunkt, Anfang der Neunzigerjahre, begannen sich die Dinge gerade zu ändern. Der Großteil der Verarbeitung natürlicher Sprache bestand jedoch aus regelbasierten logischen deklarativen Systemen. Aber es war auch in jenen Jahren, Anfang der 1990er Jahre, als man begann, große Mengen menschlicher Sprachmaterialien, Texte und Sprache digital verfügbar zu machen. Das war also eigentlich kurz bevor das World Wide Web explodierte. Aber es handelte sich bereits um Dinge wie juristische Materialien, Zeitungsartikel und parlamentarisches SARS, wo man zuletzt Millionen von Wörtern der menschlichen Sprache in die Hände bekommen konnte. Und es schien einfach ganz klar, dass es spannende Dinge geben musste, die man machen konnte, indem man empirisch mit viel menschlicher Sprache arbeitete. Und das hat mich wirklich dazu gebracht, mich mit einer neuen Art der Verarbeitung natürlicher Sprache zu beschäftigen, die dann zu meiner späteren Karriere geführt hat. >> Es hört sich so an, als ob Ihre Karriere ursprünglich eher in der Linguistik verlief und sich mit dem Aufkommen von Daten, maschinellem Lernen und empirischen Methoden zu NLP, maschinellem Lernen und NLP verlagerte. >> Ja, ich meine, es hat sich auf jeden Fall verändert, und ich habe mich auf jeden Fall viel mehr auf die Verarbeitung natürlicher Sprache und Modelle für maschinelles Lernen verlagert. Aber bis zu einem gewissen Grad hat sich das Gleichgewicht verändert. Aber damit beschäftige ich mich schon seit einiger Zeit, eigentlich ging es als Student meiner Abschlussarbeit darum, die Formen von Wörtern zu lernen. Das ist zu einem bekannten Problem beim Erlernen früherer englischer Verben und der frühen Verbindungsliteratur geworden. Und ich habe versucht, Paradigmen für Verbformen zu lernen. Und ich lernte Regeln für die verschiedenen Formen mithilfe des C 4.5-Entscheidungsbaum-Lernalgorithmus. [LACHEN] Wenn Sie sich daran erinnern. >> Ja, richtig, gute Zeiten. Ja, und es ist überraschenderweise nicht intuitiv, oder? Ich weiß nicht, wie der Übergang vom Präsens zum Präteritum und allen anderen Sonderfällen sein kann. >> Ja. >> Ja, also haben wir viel über die NLP-Verarbeitung natürlicher Sprache gesprochen. Können Sie für einige der Lernenden, die zum ersten Mal mit maschinellem Lernen beginnen, sagen, was NLP ist ? >> Sicher, absolut, ja. NLP steht also für die Verarbeitung natürlicher Sprache. Ein anderes Wort oder ein Begriff, der manchmal dafür verwendet wird, ist Computerlinguistik, es ist dasselbe. Ich meine, die Verarbeitung natürlicher Sprache ist eigentlich ein seltsamer Begriff, oder? Das bedeutet also, dass wir Dinge mit menschlichen Sprachen machen. Sie müssen also davon ausgehen, dass Sie ein ausreichender Informatiker sind, sodass Sie, wenn Sie Sprache sagen, in der Programmiersprache Ihres Gehirns denken. Und deshalb muss man „natürliche Sprache“ sagen, um zu meinen, dass man über die Sprachbilder spricht, die Menschen verwenden. Insgesamt bedeutet die Verarbeitung natürlicher Sprache also, alles Intelligente mit menschlichen Sprachen zu tun. In gewisser Hinsicht lässt sich das also aufteilen in das Verstehen menschlicher Sprachen, die Produktion menschlicher Sprachen und den Erwerb menschlicher Sprachen, obwohl die Menschen oft auch im Hinblick auf verschiedene Anwendungen darüber nachdenken. Und dann denken Sie vielleicht über Dinge wie maschinelle Übersetzung oder die Beantwortung von Fragen oder die Erstellung von Werbetexten oder Zusammenfassungen nach. Es gibt so viele verschiedene Aufgaben, an denen Menschen mit bestimmten Zielen arbeiten und bei denen man Dinge mit menschlicher Sprache erledigt. Und es gibt viel Verarbeitung natürlicher Sprache, weil so viel von dem, was die Welt auf unsere menschliche Welt einwirkt, in Form von menschlichem Sprachmaterial verarbeitet und übertragen wird. Also, wegen all dieser Anwendungen oder sogar der Websuche, oder? Die meisten von uns nutzen NLP. >> Ja. >> Viele, viele Male [UNVERSTÄNDLICH] >> Sie haben Recht, in gewisser Weise ist die Websuche die größte Anwendung natürlicher Sprache, oder?

[LACHEN] Und das ist wirklich das große Problem, ich meine, traditionell war es eher ein einfaches, oder? In den guten alten Zeiten gab es zwar verschiedene Wartefaktoren und so weiter, aber es ging hauptsächlich um die Art der passenden Schlüsselwörter, dann um Ihre Suchbegriffe und dann um einige Faktoren für die Qualität der Seite. Es fühlte sich nicht wirklich wie ein Sprachverständnis an, aber das hat sich im Laufe der Jahre wirklich verändert. Wenn Sie heutzutage also einer Suchmaschine eine Frage stellen, wird Ihnen oft ein Antwortfeld angezeigt, in dem sie einen Text extrahiert hat und die ihrer Meinung nach richtige Antwort fett, farbig oder so ähnlich darstellt . Das ist dann diese Aufgabe des Beantwortens von Fragen und dann ist es wirklich eine Aufgabe zum Verstehen natürlicher Sprache. >> Ja, ja, und ich habe das Gefühl, dass es neben der Websuche vielleicht auch die ganz große ist, selbst wenn wir auf eine Online-Shopping-Website oder eine Film-Website gehen und dort eingeben, was wir wollen, und danach eine Website-Suche durchführen kleinere Website als die großen Suchmaschinen. Dabei werden zunehmend auch ausgefeilte NLP-Algorithmen eingesetzt und es wird ebenfalls ein großer Mehrwert geschaffen. Vielleicht ist es für Sie nicht das echte NLP, aber es scheint dennoch sehr wertvoll zu sein. >> Ich stimme zu, es ist sehr wertvoll. Und es gibt viele interessante Probleme auf jeder E-Commerce-Website, wobei die Suche tatsächlich sehr schwierig ist, wenn Leute die Art von Waren beschreiben, die sie wollen. Und Sie müssen versuchen, es mit den verfügbaren Produkten in Einklang zu bringen. Das ist, wie sich herausstellt, überhaupt kein einfaches Problem. >> Ja, das stimmt, ja. In den letzten, ich weiß nicht, ein paar Jahrzehnten hat NLP also einen großen Wandel durchgemacht, weg von den eher regelbasierten Techniken, auf die Sie gerade angespielt haben, hin zum weitaus umfassenderen Einsatz von wirklich maschinellem Lernen. Und Sie waren einer der Leute, die Teile dieses Angriffs anführten und jeden Schritt des Weges beobachteten. Sie erschufen einige der Stämme, während sie geschahen. Können Sie etwas über diesen Prozess und das, was Sie gesehen haben, sagen? >> Sicher, absolut. Ja, als ich als Student im Grundstudium anfing, wurde der Großteil der Verarbeitung natürlicher Sprache durch handgebaute Systeme durchgeführt, die auf verschiedene Weise Regeln und Inferenzverfahren nutzten, um sozusagen einen Pfad und ein Verständnis für einen Textabschnitt aufzubauen . >> Was ist ein Beispiel für ein Regel- oder Inferenzsystem? Eine Regel könnte also Teil der Struktur der menschlichen Sprache sein. Wie im Englischen besteht ein Satz normalerweise aus einer Subjekt-Nominalphrase, gefolgt von einem Verb und einer Objekt-Nominalphrase. Und das gibt Ihnen eine Vorstellung davon, wie Sie die Bedeutung des Satzes verstehen können. Aber es könnte auch etwas darüber aussagen, wie ein Wort zu interpretieren ist, sodass viele Wörter im Englischen sehr mehrdeutig sind. Aber wenn Sie so etwas wie das Wort „Stern“ haben und es im Zusammenhang mit einem Film steht, dann bezieht es sich wahrscheinlich auf einen Menschen auf diesem astronomischen Objekt. Und man hat damals versucht, solche Dinge mit solchen Regeln zu bewältigen. Heutzutage scheint es für uns nicht sehr wahrscheinlich, dass das funktioniert, aber früher war das ziemlich normal. Und erst als viel digitaler Text und Sprache verfügbar wurde, schien es wirklich so, als gäbe es eine andere Möglichkeit, stattdessen Statistiken über menschliche Sprache und Material zu berechnen und Modelle für maschinelles Lernen zu erstellen. Und das war das Erste, worauf ich mich etwa Mitte bis Ende der 1990er Jahre einließ. Der erste Bereich, in dem ich anfing, viel zu recherchieren, Artikel zu veröffentlichen und bekannt zu werden, war der Aufbau dessen, was wir früher oft als statistische Verarbeitung natürlicher Sprache bezeichneten. Später verschmolz es jedoch mit allgemein problematischen Ansätzen für künstliche Intelligenz und maschinelles Lernen. Und das hat uns, sagen wir mal, ungefähr bis ins Jahr 2010 geführt. Und ungefähr zu diesem Zeitpunkt begann das neue Interesse am Deep Learning mithilfe großer künstlicher neuronaler Netze zu beginnen. Für mein Interesse daran muss ich mich wirklich bei Andrew bedanken, denn zu diesem Zeitpunkt ist Andrew immer noch Vollzeit an der Stanford University und er war im Büro neben mir und er war wirklich begeistert von den neuen Dingen, die in diesem Bereich passierten Deep Learning, schätze ich. Jedem, der sein Büro betrat, sagte er, es sei irgendwie aufregend für das, was jetzt passiert, und ich bin in einem neuronalen Netzwerk, man

muss anfangen, sich das anzuschauen. Und das war wirklich der Anstoß, der mich schon ziemlich früh dazu brachte, mich mit der Betrachtung von Dingen in neuronalen Netzen zu befassen. Ich hatte tatsächlich schon einiges davon gesehen, und als ich hier Student war, war Dave Rummelhardt an der Stanford University in Psychiatrie und ich hatte an seinem Kurs über neuronale Netze teilgenommen. Ich hatte also einiges davon gesehen, aber es war eigentlich nicht das, worauf ich mich im Rahmen meiner eigenen Forschung eingelassen hatte. Also- >> Das wusste ich nicht, danke, ja. >> Ja. >> Und dann haben wir schließlich gemeinsam einige Schüler betreut. >> Ja, absolut. >> Ich würde gerne den Aufstieg von Deep Learning und NLP hören. Was haben Sie gesehen, seit Sie in diesem Bereich tätig sind? >> Ja, ab etwa 2010 begannen die Studenten damit, die ersten Aufsätze und Deep Learning für NLP-Konferenzen zu verfassen. Es ist immer schwer, wenn man versucht, etwas Neues zu machen. Wir haben genau die gleichen Erfahrungen gemacht, die Menschen vor etwa 15 Jahren gemacht haben, als sie versuchten, statistisches NLP durchzuführen: Wenn es eine etablierte Vorgehensweise gibt, ist es wirklich schwierig, neue Ideen voranzutreiben. Einige unserer ersten Beiträge wurden also von Konferenzen abgelehnt und erschienen stattdessen auf Konferenzen zum maschinellen Lernen oder Deep-Learning-Workshops, aber sehr schnell begann sich das zu ändern und die Leute interessierten sich sehr für Ideen für neuronale Netze. Aber ich habe das Gefühl, dass sich die Periode der neuronalen Netzwerke, die praktisch um 2010 begann, in zwei Teile teilt, weil die erste Periode, sagen wir mal, im Grunde genommen bis 2018 dauert. Wir haben große Erfolge beim Aufbau neuer Netzwerke für alle gezeigt Arten von Aufgaben. Wir haben sie für syntaktisches Parsen und Stimmungsanalyse erstellt. Und was sonst, Alter? >> Beantwortung der Frage. Aber es war so, als hätten wir das Gleiche gemacht wie früher mit anderen Arten von Modellen für maschinelles Lernen, nur dass wir jetzt über ein besseres Modell für maschinelles Lernen verfügten. Und anstatt eine logistische Regression oder eine Support-Vektor-Maschine zu trainieren, haben wir immer noch die gleiche Art von Sentiment-Analyse-Aufgabe durchgeführt, aber jetzt machen wir es mit einem neuronalen Netzwerk. Wenn ich jetzt zurückblicke, denke ich, dass die größere Veränderung um das Jahr 2018 herum stattfand. Denn damals kam die Idee auf, nun ja, wir könnten einfach mit einer großen Menge menschlichen Sprachmaterials beginnen und große, selbstüberwachte Modelle bauen. Das waren damals Modelle wie [UNVERSTÄNDLICH] und GPTs und Nachfolgemodelle dazu. Und sie konnten sich durch die Wortvorhersage in einer riesigen Textmenge gewissermaßen dieses erstaunliche Wissen über menschliche Sprachen aneignen. Und ich denke wirklich, dass man das im Nachhinein als einen größeren Wendepunkt betrachten wird, an dem sich die Art und Weise, wie Dinge gemacht wurden, wirklich verändert hat. >> Ja, ich denke, es gibt diesen Trend dahingehend, dass große Sprachmodelle aus riesigen Datenmengen lernen. Ich glaube, schon im Vorfeld gab es eine Ihrer Forschungsarbeiten, die mich wirklich ein wenig umgehauen hat, nämlich ein Handschuhpapier. Also mit Worteinbettungen, bei denen man die Vektorzahlen lernt, um ein Wort mithilfe eines neuronalen Netzwerks darzustellen. Das war für mich ziemlich überwältigend. Und dann hat die Arbeit, die Sie geleistet haben, die Mathematik wirklich aufgeräumt und es so viel einfacher gemacht. Und dann erinnere ich mich, dass ich gesagt habe: Das ist alles, was ich tun muss. Und dann können Sie diese wirklich überraschend detaillierten Darstellungen lernen. Der Computer lernt die Nuancen dessen, was Wörter bedeuten. >> Absolut. Ja, also sollte ich anderen ein wenig Anerkennung zollen. Andere Leute arbeiteten ebenfalls an ähnlichen Ideen, darunter Ja Weston und Kollegen bei Google. Aber die Handschuhwortvektoren sind eines der bekanntesten Wortvektorsysteme. Das haben diese Wortvektoren bereits getan. Ja, Sie haben Recht. Veranschaulichen Sie die Idee des selbstüberwachten Lernens dadurch, dass wir einfach riesige Textmengen genommen haben. Und dann könnten wir diese Modelle bauen, die enorm viel über die Bedeutung von Wörtern wissen. Es ist immer noch etwas, was ich den Leuten jedes Jahr in der ersten Vorlesung meines NLP-Kurses zeige. Weil es etwas Einfaches ist, aber tatsächlich so überraschend gut funktioniert. Sie können eine solche einfache Modellierung durchführen, bei der Sie versuchen, ein Wort anhand der Wörter im Kontext vorherzusagen, und indem Sie einfach die Mathematik des

Lernens ausführen, um diese Vorhersagen zu treffen. Nun, Sie lernen all diese Dinge über Wortbedeutungen und können diese wirklich schönen Muster ähnlicher Wortbedeutungen oder Analogien von etwas erstellen. Bleistift ist die Zeichnung, ebenso wie Pinsel, und da steht „Malerei“, oder? Dass es sozusagen schon eine Menge erfolgreiches Lernen zeigt. Das war also der Vorläufer dessen, was dann mit Dingen wie Burton GPT zur nächsten Stufe weiterentwickelt wurde, wo es nicht nur um die Bedeutung einzelner Wörter ging. Sondern Bedeutungen ganzer Textteile und Kontexte. >> Ja, also fand ich es erstaunlich, dass man ein kleines neuronales Netzwerk oder ein Modell nehmen und ihm dann viele englische Sätze oder eine andere Sprache geben und das Wort verstecken kann. Bitten Sie es, das Wort vorherzusagen , das ich gerade getroffen habe, und so diese Analogien zu lernen . Und diese sehr tiefen, Ihrer Meinung nach wirklich tiefen Dinge hinter der Bedeutung des Wortes. Und dann 2018, vielleicht dieser andere Infektionspunkt, was geschah danach? >> Ja. Das war also im Jahr 2018 der Punkt, an dem eigentlich zwei Dinge passierten. Eine Sache ist, dass die Menschen, oder eigentlich im Jahr 2017, diese neue Architektur entwickelt haben. Das war viel besser auf moderne parallele GPUs skalierbar. Und das war die Transformatorarchitektur. Der zweite Teil davon war jedoch, dass die Leute es vielleicht wiederentdecken, weil ich den gleichen Trick wie beim Handschuhmodell verwendet habe, nämlich wenn man die Aufgabe hat, einfach ein Wort in einem gegebenen Kontext vorherzusagen. Entweder ein Kontext auf beiden Seiten oder die vorangehenden Wörter, die sich einfach als erstaunliche Lernaufgabe herausstellen. Und das überrascht viele Menschen. Und oft sieht man Diskussionen, in denen Leute sagen, dass es nichts Interessantes gibt, wenn man Dinge herabwürdigt. Dabei werden lediglich Statistiken erstellt, um vorherzusagen, welches Wort am wahrscheinlichsten nach den vorhergehenden Wörtern steht. Und ich denke, das wirklich Interessante ist, dass das wahr ist, aber es ist nicht wahr. Denn ja, die Aufgabe besteht darin, dass Sie das nächste Wort anhand der vorhergehenden Wörter vorhersagen. Aber das wirklich Interessante ist, wenn man diese Aufgabe wirklich so gut wie möglich erledigen möchte. Dann ist es tatsächlich hilfreich, den gesamten Rest des Satzes zu verstehen und zu wissen, wer was mit wem macht und was im Satz steht. Aber darüber hinaus hilft es auch, die Welt zu verstehen, denn wenn Ihr Text etwas in die Richtung der auf Fidschi verwendeten Währung geht, ist das so. Nun, Sie müssen über etwas Weltwissen verfügen, um zu wissen, wie die richtige Antwort darauf lautet. Und so lernen gute Modelle dabei, sowohl die Struktur von Sätzen und ihrer Bedeutung zu verfolgen als auch Fakten über die Welt zu kennen, damit sie Vorhersagen treffen können. Und deshalb wird daraus etwas, das manchmal als „KI-Abschlussaufgabe“ bezeichnet wird, oder? Das brauchst du wirklich. Es gibt nichts, was bei der Beantwortung der Frage „Welches Wort kommt als nächstes“ nicht wirklich nützlich sein kann, oder? Man kann im WM-Halbfinale gegen die Mannschaften antreten und man muss etwas über Fußball wissen [LACHEN], um die richtige Antwort zu geben. >> Ich vervollständige dieses lustige Konzept, oder? Ist die Idee, dass man dieses eine Problem lösen kann, man alles in der KI lösen kann oder eine Art Analogie zu NP-vollständigen Problemen aus der Computertheorie herstellen kann ? Was denken Sie? Glauben Sie, dass die Vorhersage des Knicks-Worts KI-vollständig ist? Ich selbst habe diesbezüglich sehr gemischte Gefühle . Ich kann sagen, ich glaube nicht, dass es wahr ist. Ich bin gespannt, was Sie denken. Ich denke, dass das nicht ganz stimmt, weil ich glaube, dass es noch andere Dinge gibt, die Menschen schaffen, sich zurechtzufinden. Es gibt Menschen, die kluge Einblicke in die Mathematik haben, oder es gibt Menschen, die sich mit etwas beschäftigen, das viel mehr ist. Ein dreidimensionales Rätsel aus der realen Welt, bei dem es darum geht, herauszufinden, wie man etwas Mechanisches oder ähnliches macht. Und das ist einfach kein Sprachproblem. Aber andererseits denke ich auch, dass Sprache der Universalität näher kommt, als manche Leute denken, weil wir in dieser 3D-Welt leben. Und darin mit unserem Körper, unseren Gefühlen und anderen Lebewesen und Artefakten um uns herum agieren. Und man könnte denken, dass davon überhaupt nicht viel in der Sprache steht. Aber eigentlich ist fast alles, worüber wir nachdenken, reden und schreiben, in Sprache. Wir können die Positionen von Dingen zueinander in der Sprache beschreiben.

So werden überraschend viele andere Teile der Welt in der sprachlichen Reflexion gesehen. Und deshalb lernen Sie auch alles über sie kennen. Wenn Sie etwas über den Sprachgebrauch lernen. >> Man lernt einen Aspekt von vielen Dingen kennen, auch wenn Dinge nicht wirklich möglich sind . >> Man lernt nicht wirklich, wie man Fahrrad fährt, [LACHEN], aber man lernt einige Aspekte dessen, was dazugehört und die man ausbalancieren muss. Und man muss die Füße auf den Pedalen haben und sie treten und so weiter. Ja. >> Und angesichts dieses Trends in LOP waren die letzten Sprachmodelle in den letzten Jahren sehr aufregend. Was denken Sie darüber, wohin das alles führen wird? >> Nun ja, es war einfach unglaublich. Erfolgreich und aufregend, oder? Wir haben also nicht wirklich alle Details erklärt, oder? In der ersten Phase des Erlernens dieser großen Sprachmodelle besteht die Aufgabe lediglich darin, das nächste Wort vorherzusagen. Und das machen Sie milliardenfach bei einem sehr großen Textstück. Und siehe da, Sie erhalten dieses große neuronale Netzwerk, das einfach ein wirklich nützliches Artefakt für alle Arten von Aufgaben zur Verarbeitung natürlicher Sprache ist. Aber dann muss man tatsächlich noch etwas damit machen, wenn man eine bestimmte Aufgabe erledigen möchte, sei es das Beantworten oder Zusammenfassen von Fragen oder das Erkennen toxischer Inhalte in sozialen Medien oder so etwas in der Art. Und an diesem Punkt gibt es eine Auswahl an Dingen, die Sie damit machen können. Die traditionelle Antwort war dann, dass Sie eine bestimmte Aufgabe hatten, sagen wir, es ginge darum, toxische Kommentare in sozialen Medien zu erkennen. Und Sie würden dafür einige überwachte Daten nehmen und dann das Sprachmodell verfeinern, um diese Klassifizierungsaufgabe zu beantworten. Aber diese Basis dieses großen selbstüberwachten Modells hat Ihnen enorm geholfen, denn sie bedeutete, dass das Modell über enorme Sprachkenntnisse verfügte und sehr schnell verallgemeinern konnte. Anders als in den alten Zeiten des überwachten Lernens, wo es so war, wenn Sie mir 10.000 beschriftete Beispielbeispiele geben, kann ich vielleicht ein einigermaßen anständiges Modell für Sie erstellen. Aber wenn Sie mir 50.000 beschriftete Beispiele geben, wird es viel besser sein. Es hat es irgendwie in diese Welt verwandelt. Nun, wenn Sie mir 100 beschriftete Beispiele geben und ich ein großes Sprachmodell verfeinere, kann ich viel besser abschneiden, als ich es mit den 50.000 Beispielen in der alten Welt geschafft hätte. Einige der neueren spannenden Arbeiten gehen jetzt sogar darüber hinaus, nun ja, vielleicht muss man das Modell überhaupt nicht mehr verfeinern. Daher haben die Leute viel mit Methoden gearbeitet, die manchmal auch als Aufforderungen oder Anweisungen bezeichnet werden und bei denen man einfach in natürlicher Sprache, vielleicht mit Beispielen, vielleicht mit expliziten Anweisungen, dem Modell einfach sagen kann, was es tun soll, und es tut, was selbst als jemand, der seit 30 Jahren in der Verarbeitung natürlicher Sprache arbeitet. Ich meine, es ist wirklich umwerfend, wie gut das funktioniert. Ich glaube, vor einem Jahrzehnt hätte ich nicht gedacht, dass wir dem Modell jetzt einfach sagen könnten, ich möchte, dass Sie diesen Text hier zusammenfassen sie werden es dann zusammenfassen. Ich finde das unglaublich. Ja, wir befinden uns in dieser sehr aufregenden Zeit, in der sich viele neue Fähigkeiten der natürlichen Sprache entfalten. Ich denke, es besteht überhaupt kein Zweifel daran, dass die Zukunft in den nächsten Jahren äußerst rosig ist, da die Menschen unterschiedliche Dinge und unterschiedliche Vorgehensweisen erarbeiten und beginnen, sich in unterschiedlichen Anwendungsbereichen zu bewerben. Die Art von Fähigkeiten, die durch die jüngsten technologischen Entwicklungen freigeschaltet wurden. Es stellt sich in der Technik immer die Frage, ob die Kurve weiterhin steil nach oben geht oder ob es dann neue Dinge gibt, wir müssen herausfinden, wie das geht. >> Es geht schon eine ganze Weile bergauf. Hoffentlich ist eine Extrapolation immer gefährlich. Aber, aber wir werden sehen, ich bin nur neugierig, wissen Sie, Sie haben das Schreiben von Eingabeaufforderungen für das MRP-System, das große Sprachmodell, erwähnt, was Sie wollen, und es scheint es auf magische Weise zu tun. Ich bin neugierig, glauben Sie, dass Prompt Engineering der Weg der Zukunft ist? Wenn ich diese Prompts schreibe, merke ich manchmal, dass es auf wundersame Weise funktioniert, und manchmal ist es frustrierend, meine Anweisungen umzuformulieren, um den Wortlaut so anzupassen, dass er genau richtig ist das gewünschte Ergebnis erzeugen. Glauben Sie also, dass

Problem-Engineering der Weg der Zukunft ist, oder glauben Sie, dass es ein Zwischen-Hack ist, bis jemand einen besseren Weg erfindet, diese zu kontrollieren, die Ausgänge dieser Systeme zu kontrollieren? >> Ich denke, es ist beides, ich denke, es wird der Weg der Zukunft sein, aber ich denke auch, dass die Leute im Moment viel herumhacken und umformulieren, um zu versuchen, die Dinge besser zum Laufen zu bringen, und mit etwas Glück mit einem noch ein paar Jahre der Entwicklung, die allmählich verschwinden werden. Ich meine, eine Möglichkeit, über den Unterschied nachzudenken, ist der Vergleich mit der Art von Sprachunterstützung oder virtueller Unterstützung, die heutzutage auf Telefonlautsprechern wie Amazon oder Alexa verfügbar ist, oder? Ich meine, ich denke, wir alle haben die Erfahrung gemacht, dass diese Geräte nicht immer großartig sind, aber wenn man weiß, wie man Dinge richtig ausdrückt, wird es etwas bewirken. Wenn Sie jedoch die falsche Formulierung verwenden, wird dies nicht der Fall sein, und der Unterschied zu Menschen besteht im Großen und Ganzen darin, dass Sie darüber nicht nachdenken müssen. Sie können sagen, was Sie wollen, und es spielt keine Rolle, welche Worte Sie wählen, sie werden den anderen Menschen ansprechen, vorausgesetzt, dass jemand, der dieselbe Sprache spricht usw., Sie versteht und tut, was Sie wollen. Und ich denke und hoffe, dass wir bei diesen Modellen die gleichen Fortschritte sehen werden, dass das Herumspielen mit der jeweiligen Formulierung, die Sie verwenden, einen großen Unterschied in der Funktionsweise machen kann. Aber hoffentlich wird das in ein paar Jahren nicht mehr wahr sein, Sie können andere Formulierungen verwenden und es wird immer noch funktionieren. Aber die Grundidee ist, dass wir in ein Zeitalter eintreten, in dem tatsächlich die menschliche Sprache als Befehlssprache verwendet werden kann, um Ihrem Computer zu sagen, was er tun soll. Anstatt also Menüs und Optionsfelder usw. verwenden oder Python-Code schreiben zu müssen, können Sie anstelle dieser Dinge sagen, was Sie wollen, und der Computer erledigt es. Ich denke, dass sich vor uns ein Zeitalter auftut, das weiter wachsen und enorme Veränderungen bewirken wird. >> Es fühlt sich an, als hätte man einen langen Weg zurückgelegt, aber es liegt noch viel vor uns und noch viel mehr vor uns. >> Ja, absolut. >> Was die Entwicklung der NLP-Technologie betrifft, ist das Einzige, was ich Sie fragen möchte, und ich vermute, dass Sie und ich diesbezüglich möglicherweise unterschiedliche Ansichten haben. In den letzten Jahrzehnten ging der Trend jedoch dahin, sich weniger auf regelbasiertes Engineering und mehr auf maschinelles Lernen auf Datenbasis zu verlassen. Manchmal blicken viele Daten in die Zukunft. Was halten Sie von dieser Mischung aus handcodierten Einschränkungen oder anderen Einschränkungen, expliziten Einschränkungen im Vergleich dazu, dass wir ein neues Netzwerk einrichten und viele Daten darauf werfen? Wo wird sich Ihrer Meinung nach das Gleichgewicht verschlechtern? >> Ich denke, es besteht kein Zweifel daran, dass das Lernen aus Daten der Weg in die Zukunft ist und was wir auch weiterhin tun werden. Aber ich denke, es gibt immer noch Raum für Modelle mit mehr Struktur und mehr induktiver Voreingenommenheit, die eine Art Grundlage für die Ausnutzung der Natur der Sprache haben. Das Modell, das in den letzten Jahren enorm erfolgreich war, ist das Transformer Neural Network und die Transformer Neural Networks, im Wesentlichen diese riesige Assoziationsmaschine. Es werden also Assoziationen von überall her aufgesaugt. >> Und schauen Sie sich zwei Wörter an und finden Sie heraus, welche Wörter zu welchem anderen Wort gehören. >> Ja. Man nutzt also alles, um alles vorherzusagen, macht es immer wieder und bekommt alles, was man will. Und wissen Sie, das war unglaublich, unglaublich erfolgreich, aber es war unglaublich erfolgreich in dem Bereich, in dem es riesige Datenmengen gibt. Richtig, so dass diese Transformatormodelle für diese großen Sprachmodelle jetzt auf zig Milliarden Textwörtern trainiert werden. Als ich mit der statistischen Verarbeitung natürlicher Sprache begann. Und einige der traditionellen Linguisten beschwerten sich darüber, dass ich Statistiken aus 30 Millionen Wörtern von Newswire sammelte. Und ich habe ein Vorhersagemodell entwickelt und dachte, das sei einfach nicht das, worum es in der Linguistik geht. Ich hatte das Gefühl, eine vollkommen gute Antwort zu haben, nämlich dass ein menschliches Kind eine Sprache lernt. Sie sind tatsächlich mehr als 30 Millionen Wörtern an Daten ausgesetzt. Aber diese Art von Datenmenge, also die Art von

Datenmenge, die wir verwendeten, waren vollkommen vernünftige Datenmengen, die es zu verwenden galt. Es geht nicht gerade darum, den menschlichen Spracherwerb zu modellieren. Aber darüber nachzudenken, wie wir für viele Daten etwas über Sprache lernen können. Aber diese modernen Transformatoren nutzen jetzt bereits mindestens zwei Größenordnungen mehr Daten. Und die meisten Leute denken, dass man die Dinge auf die nächste Ebene bringen kann, indem man noch mehr verwendet und es auf drei Größenordnungen anhebt. Und in gewisser Hinsicht war diese Expansionsstrategie äußerst effektiv. Ich kann es also niemandem verübeln, wenn er sagt: „Lasst uns die Automatisierung um eine weitere Größenordnung erweitern und sehen, was für erstaunliche Dinge wir tun können." Aber es zeigt auch, dass menschliches Lernen viel, viel besser ist, wenn es darum geht, viel mehr Informationen aus einer recht begrenzten Datenmenge zu extrahieren. Und an diesem Punkt kann man verschiedene Hypothesen aufstellen. Aber ich denke, es ist vernünftig anzunehmen, dass das menschliche Lernen in gewisser Weise auf die Struktur der Welt ausgerichtet ist. Und Dinge, die es in der Welt sieht und die es ihm ermöglichen, schneller aus weniger Daten zu lernen. >> Richtig, da stimme ich dir zu. Ich denke, dass das bessere Lernen unserer Räume, unser derzeitiges maschinelles Lernen, viel weniger effizient ist oder Daten viel weniger effizient nutzt. Es gibt also viel mehr Daten als jedes Kind. Und dann denke ich darüber nach, ob die verbesserten Lernalgorithmen auf sprachlichen Regeln basieren oder ob es nur Ingenieure sein werden. Entwicklung viel effizienterer Versionen des Transformators oder was auch immer danach kommt. Ich denke, das wird so sein. >> Das wird traditionell sein. Ich glaube nicht, dass es daran liegen wird, dass Leute explizit traditionelle Sprachregeln in das System integrieren. Ich glaube nicht, dass das der richtige Weg ist. Andererseits denke ich, dass wir allmählich beobachten, dass Modelle wie diese Transformatormodelle tatsächlich die Struktur der Sprache selbst entdecken, oder? Die weitreichenden Auswirkungen der menschlichen Sprache sind also, dass im Englischen das Subjekt vor dem Verb und das Objekt danach steht. Im Japanischen hingegen stehen die Verben am Ende des Satzes und Subjekt und Objekt normalerweise in dieser Reihenfolge davor. Aber es könnte in der anderen Reihenfolge tatsächlich sein, dass Transformatormodelle diese Fakten lernen. Sie können sie befragen und feststellen, dass sie diese Vorstellungen kennen, auch wenn ihnen nie explizit etwas über Subjekte und Objekte gesagt wurde. Ich denke, sie entdecken auch noch viel mehr über den Sprachgebrauch und -kontext, die Bedeutungen und Bedeutungen von Wörtern und darüber, was unangenehme Sprache ist und was nicht. Aber ein Teil dessen, was sie lernen, ist die gleiche Art von Struktur, die Linguisten als Struktur verschiedener menschlicher Sprachen beschrieben haben. >> Es ist also, als hätten Linguisten über viele Jahrzehnte hinweg bestimmte Dinge entdeckt. Und durch das Training mit Milliarden von Wörtern entdecken Transformatoren die gleichen Dinge, die Linguisten bei Menschen entdeckt haben. Das ist cool. Das alles sind also wirklich aufregende Fortschritte im NLP, die durch maschinelles Lernen und andere Dinge vorangetrieben werden. Für jemanden, der das Feld betritt, maschinelles Lernen oder KI oder NLP betritt. Es ist einfach viel los. Welchen Rat würden Sie jemandem geben, der in maschinelles Lernen einsteigen möchte? >> Ja. Nun, es ist ein guter Zeitpunkt, um einzusteigen. Ich denke, es besteht überhaupt kein Zweifel daran, dass wir uns noch in einem frühen Stadium befinden, in dem wir die Auswirkungen dieses neuen Ansatzes sehen, bei dem die Software-Informatik auf der Grundlage einer viel stärkeren Nutzung effektiv neu erfunden wird maschinelles Lernen. Und die verschiedenen anderen Dinge, die sich daraus ergeben. Und ganz allgemein gibt es branchenübergreifend jede Menge Möglichkeiten für mehr Automatisierung. Ich nutze für mich oder in anderen Bereichen wie Vision und Robotik, den gleichen Dingen, mehr die Interpretation von menschlichem Sprachmaterial. Also viele Möglichkeiten. An diesem Punkt gibt es also offensichtlich viel zu tun, denn man möchte eine gute Grundlage schaffen, richtig. Sie kennen also einige der wichtigsten technischen Methoden des maschinellen Lernens und verstehen Ideen zur Erstellung von Modellen aus Daten. Schauen Sie sich Verluste an, führen Sie Schulungen durch, diagnostizieren Sie Fehler, all diese Kernthemen. Das ist insbesondere für die Verarbeitung natürlicher Sprache auf jeden Fall nützlich, einige dieser Fähigkeiten sind durchaus

relevant. Aber dann gibt es bestimmte Arten von Modellen, die häufig verwendet werden, einschließlich des Transformators, über den wir heute viel gesprochen haben. Sie sollten auf jeden Fall über Transformatoren Bescheid wissen, denn tatsächlich werden sie zunehmend in allen anderen Bereichen des maschinellen Lernens sowie in der visuellen Bioinformatik eingesetzt, sogar in der Robotik werden mittlerweile Transformatoren eingesetzt. Aber darüber hinaus denke ich, dass es auch nützlich ist, etwas über die menschliche Sprache und die Natur der damit verbundenen Probleme zu lernen. Denn auch wenn die Menschen die Regeln der menschlichen Sprache nicht direkt in ihr Computersystem kodieren werden. Ein Gespür dafür, was in der Sprache passiert, worauf man achten muss und was man modellieren möchte, ist immer noch eine nützliche Fähigkeit. >> Und dann geht es um das Erlernen der Grundlagen und um das Erlernen dieser Konzepte. Sie sind mit einem sprachwissenschaftlichen Hintergrund in die KI eingestiegen und wir sehen jetzt Menschen aus allen Gesellschaftsschichten, die mit der Arbeit in der KI beginnen möchten. Was denken Sie über die Vorbereitung, die man haben sollte, oder darüber, wie man mit etwas anderem als Informatik oder KI beginnen kann? Es gibt also viele Orte, von denen Sie kommen und die Sie auf unterschiedliche Weise durchqueren können. Und wir sehen eine Menge Leute, die das tun, sie sind Leute, die in verschiedenen Bereichen angefangen haben, sei es Chemie, Physik oder noch viel weiter entfernte Bereiche. Und die Menschen haben in der Vergangenheit begonnen, sich mit maschinellem Lernen zu befassen. Ich denke, dass es da sozusagen zwei Ebenen der Antwort gibt. Eine der erstaunlichen Veränderungen besteht darin, dass es jetzt sehr gute Softwarepakete gibt, mit denen Sie Ihre Netzwerkmodelle bearbeiten können. Diese Software ist wirklich einfach zu bedienen. Sie müssen eigentlich nicht viele hochtechnische Dinge verstehen. Sie müssen eine fundierte Vorstellung davon haben, was die Idee des maschinellen Lernens ist. Und wie trainiere ich ein Modell und worauf sollte ich achten und auf die Zahlen, die ausgedruckt werden, um zu sehen, ob es funktioniert? Man muss aber eigentlich keinen höheren Abschluss haben, um diese Modelle bauen zu können. Ich meine, und was wir tatsächlich sehen, ist, dass sich viele Oberstufenschüler damit beschäftigen, weil es tatsächlich etwas ist, das man sich aneignen und umsetzen kann, wenn man über grundlegende Computerkenntnisse und ein wenig Programmierkenntnisse verfügt. Es ist einfach viel zugänglicher als viele Dinge, die sich mit KI befasst haben, sei es außerhalb der KI oder in anderen Bereichen wie Betriebssystemen oder Sicherheit. Aber wenn Sie auf eine tiefere Ebene vordringen und tatsächlich mehr darüber verstehen möchten, was vor sich geht. Ich denke, man kann nicht wirklich dorthin gelangen, wenn man nicht über bestimmte mathematische Grundlagen verfügt, zum Beispiel weil DeepLearning letzten Endes auf Infinitesimalrechnung basiert und man Funktionen optimieren muss. Und wenn man in diesem Bereich keinerlei Hintergrundwissen hat, endet das meiner Meinung nach irgendwann in einem Krieg. Also >> Das maschinelle Lernen in der Datenwissenschaft. Es ist für einige unserer Arbeiten praktisch. >> Ja. Ich glaube also, dass ich auf einer gewissen Ebene, wenn man sich im Hauptfach Geschichte oder nicht-mathematische Teile der Psychologie befindet, tatsächlich einen guten Freund habe, der, ja, er hat in der Graduiertenschule Infinitesimalrechnung gelernt, weil er Psychologe war, und das hatte er noch nie getan es vorher. Und beschloss, dass er anfangen wollte, etwas über diese neuen Arten von Modellen zu lernen, und dass es noch nicht zu spät war, an einem Kuhkurs teilzunehmen. Und das tat er, richtig. Man muss also einiges davon wissen, aber für viele Leute, wenn sie etwas davon schon einmal gesehen haben, auch wenn man etwas eingerostet ist. Ich denke, man kann irgendwie wieder in die richtige Stimmung kommen, und es spielt keine Rolle, dass man sich als Student nicht mit KI oder maschinellem Lernen und solchen Dingen beschäftigt hat, man kann wirklich anfangen zu lernen, wie man diese Modelle baut und Dinge tun und wirklich, das ist meine eigene Geschichte, oder? Dass ich trotz der Tatsache, dass ich heutzutage an der Fakultät für Ingenieurwissenschaften in Stanford studieren darf, kein Ingenieur bin. Ich habe einen Doktortitel in Linguistik, den ich größtenteils von Kenntnissen in Mathematik und Linguistik und Programmierkenntnissen auf die Entwicklung von KI-Modellen umgestellt habe. >> Ich über etwas. Denken Sie, dass die verbesserten Bibliotheken und Abstraktionen, die jetzt verfügbar sind, wie

Codierungs-Frameworks, wie der Tensorfluss von P Torch, aussehen? Glauben Sie, dass dadurch die Notwendigkeit verringert wird, die Analysis zu verstehen? Denn Junge, es ist schon eine Weile her, dass ich aufgrund der automatischen Differenzierung tatsächlich ein Derivat nehmen musste, um überhaupt eine neue Neuronetzwerkarchitektur zu implementieren oder zu erstellen >> Ja, ich meine, absolut. Ich meine, also in den frühen Tagen, als wir die Dinge sozusagen von 2010 bis 2015 machten, oder? Für jedes Modell, das wir gebaut haben, haben wir die Ableitungen von Hand ausgearbeitet und dann Code und was auch immer geschrieben. Manchmal war es Python, aber manchmal war es vielleicht Java oder C [LACHEN], um diese Ableitungen zu berechnen und zu überprüfen, ob wir sie richtig gemacht haben und so weiter, wobei man heutzutage eigentlich nichts davon wissen muss, um DeepLearning-Modelle zu erstellen. Ich meine, das ist tatsächlich etwas, worüber ich nachdenke, sogar in Bezug auf meine eigene Verarbeitung natürlicher Sprache mit dem DeepLearning-Kurs, den ich unterrichte. Am Anfang beschäftigen wir uns noch mit der Matrizenrechnung und stellen sicher, dass die Leute etwas über Jacobian und ähnliches wissen, damit sie verstehen, was bei der Backpropagation, DeepLearning, gemacht wird. Aber es gibt so etwas, bei dem das bedeutet, dass wir ihnen einfach zwei Wochen lang die Hölle heiß machen. Es ist so etwas wie ein Bootcamp oder so etwas, das sie leiden lässt. Und dann sagen wir: „Aber Sie machen den Rest des Unterrichts mit Pytorch und sie müssen sozusagen nie wieder etwas davon wissen, oder?“ Es stellt sich immer die Frage, wie tief man in die technischen Grundlagen einsteigen möchte, oder? Du kannst doch weitermachen, oder? Zum Beispiel muss ein Informatiker im Jahr 2020 verstehen, was Elektronik und Transistoren sind oder was in Ihrer CPU passiert. Nun, es ist kompliziert, ich meine, in vielerlei Hinsicht ist es hilfreich, einiges davon zu wissen. Ich meine, ich weiß, Andrew, Sie waren einer der Pioniere, die maschinelles Lernen auf die GPU übertragen haben, und das bedeutet in gewisser Weise, dass Sie ein Gefühl dafür haben mussten, dass es da draußen diese neue Hardware gibt. Und es weist einige Eigenschaften der Parallelität auf, was bedeutet, dass es wahrscheinlich möglich ist, etwas Aufregendes zu tun. Daher ist es nützlich, etwas umfassenderes Wissen und Verständnis zu haben, und manchmal geht etwas kaputt, und wenn man tiefergehende Kenntnisse hat, kann man verstehen, warum es kaputt gegangen ist. Aber es gibt noch einen anderen Sinn: Die meisten Menschen müssen sich auf manche Dinge verlassen, und heutzutage können Sie das meiste, was Sie in der Modellierung neuronaler Netze tun möchten, auch ohne Kenntnisse in Analysis machen. >> Ja, ich denke, das ist ein toller Punkt. Ich habe das Gefühl, dass manchmal die Zuverlässigkeit der Abstraktion darüber entscheidet, wie oft man etwas reparieren muss, das kaputt ist. Tatsächlich ist mein Verständnis der Quantenphysik sehr schwach. Ich verstehe es kaum. Man könnte also argumentieren: Ich verstehe nicht, wie Computer funktionieren, weil Transistoren in der Quantenphysik gebaut sind. Aber zum Glück musste ich, wenn mit dem Transistor etwas schief ging, nie große Anstrengungen unternehmen, um es zu reparieren, [KREUZSPRECHEN], glaube ich. Und so denke ich, oder ein anderes Beispiel, die Sortierfunktion, die Bibliothek zum Sortieren von Dingen, und manchmal funktionieren sie tatsächlich nicht, richtig, Swap im Speicher oder was auch immer. Und wenn Sie wirklich verstehen, wie die Sortierfunktion funktioniert, können Sie das Problem beheben. Aber manchmal, wenn wir Abstraktionen, Bibliotheken und APIs haben, die zuverlässig genug sind, verringert die Tatsache, dass diese Abstraktionen gut sind, die Notwendigkeit, einige der Dinge zu verstehen, die bei mir passieren. Es ist also eine aufregende Welt. Es fühlt sich an, als hätten wir Giganten, die auf den Schultern von Giganten aufbauen, und all diese Dinge werden von Monat zu Monat komplexer und aufregender. >> Ja, absolut. >> Vielen Dank, Chris, das war wirklich interessant und inspirierend und ich hoffe, dass alle, die das sehen, Chris seine eigene Reise zum Informatiker hören können. Und ein führender, vielleicht der führende NFP-Informatiker zu werden, sowie all diese aufregende Arbeit, die derzeit in NFP stattfindet. Ich hoffe, das inspiriert auch Sie, ins Wasser zu springen und es auszuprobieren. Es gibt einfach noch viel mehr Arbeit, die unsere Gemeinschaft gemeinsam erledigen muss. Ich denke, je mehr von uns daran arbeiten, desto besser

wird es der Welt gehen. Vielen Dank, Chris. Es war wirklich toll, Sie hier zu haben. >> Vielen Dank, Andrew. Es hat Spaß gemacht zu plaudern.[MUSIK]