Überwachtes vs. unbeaufsichtigtes Lernen

Was ist maschinelles Lernen?

Was ist maschinelles Lernen? Sie verwenden es wahrscheinlich viele Male am Tag, ohne es zu wissen. Möchten Sie jederzeit wissen, wie ich eine Sushi-Rolle mache? Um dies herauszufinden, können Sie eine Websuche bei Google, Bing oder Baidu durchführen. Und das funktioniert so gut, weil ihre Software für maschinelles Lernen herausgefunden hat, wie Webseiten bewertet werden. Oder wenn Sie Bilder auf Instagram oder Snapchat hochladen und sich denken: „Ich möchte meine Freunde markieren, damit sie ihre Bilder sehen können.“ Nun, diese Apps können Ihre Freunde auf Ihren Bildern erkennen und sie auch beschriften. Das ist auch maschinelles Lernen. Oder wenn Sie gerade einen Star Wars-Film auf dem Video-Streaming-Dienst gesehen haben und sich fragen, welche anderen ähnlichen Filme ich mir ansehen kann? Nun, der Streaming-Dienst wird wahrscheinlich maschinelles Lernen nutzen, um etwas zu empfehlen, das Ihnen gefallen könnte. Jedes Mal, wenn Sie Voice-to-Text auf Ihrem Telefon verwenden, um eine Textnachricht zu schreiben. >> Hey Andrew, wie geht es? >> Oder sagen Sie es Ihrem Telefon. Hey Siri, spiele ein Lied von Rihanna oder frage dein anderes Telefon: Okay, Google zeigt mir indische Restaurants in meiner Nähe.

Das ist auch maschinelles Lernen. Jedes Mal, wenn Sie eine E-Mail mit dem Titel „Herzlichen Glückwunsch!“ erhalten. Du hast eine Million Dollar gewonnen. Nun, vielleicht bist du reich, Glückwunsch. Oder es ist wahrscheinlicher, dass Ihr E-Mail-Dienst die E-Mail als Spam markiert. Auch das ist eine Anwendung des maschinellen Lernens. Über die Verbraucheranwendungen hinaus, die Sie möglicherweise verwenden, dringt KI auch schnell in große Unternehmen und in industrielle Anwendungen vor. Ich mache mir zum Beispiel große Sorgen über den Klimawandel und freue mich, dass maschinelles Lernen bereits darauf abzielt, die Stromerzeugung durch Windkraftanlagen zu optimieren. Oder im Gesundheitswesen hält es zunehmend Einzug in Krankenhäuser, um Ärzten bei der Erstellung genauer Diagnosen zu helfen. Oder kürzlich hat Landing AI viel Arbeit geleistet und Computer Vision in Fabriken eingebaut, um zu prüfen, ob etwas, das vom Fließband kommt, Mängel aufweist. Das ist maschinelles Lernen, die **Wissenschaft, Computer zum Lernen zu bringen, ohne explizit programmiert zu werden**. In diesem Kurs lernen Sie maschinelles Lernen kennen und können selbst maschinelles Lernen implementieren und programmieren.

Hier ist eine Definition dessen, was maschinelles Lernen ist, das Arthur Samuel zugeschrieben wird. Sie definierte maschinelles Lernen als den Studienbereich, der Computern die Fähigkeit verleiht, zu lernen, ohne explizit programmiert zu werden. Samuel erlangte Berühmtheit dadurch, dass er in den 1950er Jahren ein Dame-Spielprogramm schrieb. Das Erstaunliche an diesem Programm war, dass Arthur Samuel selbst kein sehr guter Damespieler war. Sie hatte den Computer so programmiert, dass er vielleicht Zehntausende Spiele gegen sich selbst spielte. Durch die Beobachtung, welche sozialen Unterstützungspositionen gewinnen und welche Positionen verlieren, lernte das **Checkers-Plane-Programm** im Laufe der Zeit, was gute oder schlechte Unterstützungspositionen sind, indem es versuchte, gute und schlechte Positionen zu vermeiden, und lernte, immer besser

zu werden beim Damespielen, weil der Computer die Geduld hatte, Zehntausende Partien gegen sich selbst zu spielen. Er konnte so viel Erfahrung im Damespiel sammeln, dass er schließlich ein besserer Damespieler wurde als Samuel selbst.

Im Allgemeinen gilt: **Je mehr Gelegenheiten Sie einem Lernalgorithmus zum Lernen geben, desto besser ist seine Leistung**.

Die beiden Hauptarten des maschinellen Lernens sind

- überwachtes Lernen und
- unüberwachtes Lernen.

Die heute mit Abstand am häufigsten verwendeten Arten von Lernalgorithmen sind **überwachtes Lernen, unüberwachtes Lernen und Empfehlungssysteme**. Das andere Thema, dem wir in dieser Spezialisierung viel Zeit widmen werden, sind praktische Ratschläge zur Anwendung von Lernalgorithmen. Das ist es, was Sie in diesem Kurs erhalten: die Werkzeuge sowie die Fähigkeiten, diese effektiv anzuwenden.

Eines der relativ einzigartigen Dinge, die Sie in diesem Kurs lernen, ist, dass Sie viel über die **Best Practices** lernen, wie man tatsächlich ein praktisches, wertvolles System für maschinelles Lernen entwickelt. Auf diese Weise ist es weniger wahrscheinlich, dass Sie in einem dieser Teams landen, die am Ende sechs Monate verlieren, wenn sie in die falsche Richtung gehen.

Anwendungen des maschinellen Lernens

Überwachtes Lernen Teil 1

Überwachtes maschinelles Lernen oder häufiger **überwachtes Lernen bezieht sich auf Algorithmen, die x-zu-y- oder Eingabe-Ausgabe-Zuordnungen lernen**. Das Hauptmerkmal des überwachten Lernens besteht darin, dass Sie Ihrem Lernalgorithmus Beispiele geben, aus denen Sie lernen können. Dazu gehören die richtigen Antworten, wobei die richtige Antwort, ich meine, die richtige Bezeichnung y für eine gegebene Eingabe x ist, und **durch das Erkennen korrekter Paare von Eingabe x und gewünschter Ausgabebezeichnung y lernt der Lernalgorithmus** schließlich, nur die Eingabe allein zu nehmen und für die Ausgabebezeichnung eine einigermaßen genaue Vorhersage oder Schätzung der Ausgabe zu liefern.

Schauen wir uns einige Beispiele an. Wenn die Eingabe x eine E-Mail und die Ausgabe y diese E-Mail ist, Spam oder nicht, erhalten Sie Ihren Spam-Filter. Oder wenn es sich bei der Eingabe um einen Audioclip handelt und die Aufgabe des Algorithmus darin besteht, das Texttranskript auszugeben , dann handelt es sich um Spracherkennung. Oder wenn Sie Englisch eingeben und es in die entsprechende Spanisch-, Arabisch-, Hindi-, Chinesisch-, Japanische- oder eine andere Übersetzung ausgeben lassen möchten, dann ist das maschinelle Übersetzung.

Oder die **lukrativste Form des überwachten Lernens wird heute wahrscheinlich in der Online-Werbung** eingesetzt. Fast alle großen Online-Werbeplattformen verfügen über einen

Lernalgorithmus, der einige Informationen zu einer Anzeige und einige Informationen über Sie eingibt und dann versucht herauszufinden, ob Sie auf diese Anzeige klicken oder nicht. Denn durch die Anzeige von Anzeigen ist die Wahrscheinlichkeit, dass sie darauf klicken, für diese großen Online-Werbeplattformen nur geringfügig höher. Jeder Klick stellt einen Umsatz dar, was diesen Unternehmen tatsächlich eine Menge Umsatz beschert.

Oder nehmen Sie die Fertigung. Ich habe in diesem Bereich tatsächlich viel Arbeit beim Erlernen von KI geleistet. Sie können einen Lernalgorithmus als Eingabe ein Bild eines hergestellten Produkts verwenden lassen, beispielsweise eines Mobiltelefons, das gerade vom Band gelaufen ist, und den Lernalgorithmus ausgeben lassen, ob das Produkt einen Kratzer, eine Delle oder einen anderen Defekt aufweist oder nicht. Dies wird als Sichtprüfung bezeichnet und hilft Herstellern, Fehler an ihren Produkten zu reduzieren oder zu verhindern. In all diesen Anwendungen trainieren Sie Ihr Modell zunächst mit Beispielen für Eingaben x und den richtigen Antworten, also den Beschriftungen y. Nachdem das Modell aus diesen Eingabe-, Ausgabe- oder x- und y-Paaren gelernt hat, kann es eine brandneue Eingabe x, etwas, das es noch nie zuvor gesehen hat, verwenden und versuchen, die entsprechende entsprechende Ausgabe y zu erzeugen. Lassen Sie uns näher auf ein konkretes Beispiel eingehen.

Angenommen, Sie möchten die Immobilienpreise basierend auf der Größe des Hauses vorhersagen. Sie haben einige Daten gesammelt und sagen, Sie zeichnen die Daten auf und es sieht so aus. Hier auf der horizontalen Achse ist die Größe des Hauses in Quadratfuß angegeben.

Wie kann Ihnen der Lernalgorithmus helfen? Eine Sache, die ein lernender Algorithmus möglicherweise tun könnte, ist zu sagen, dass es für die direkte Verbindung zu den Daten und das Ablesen der geraden Linie so aussieht, als könnte das Haus Ihres Freundes für etwa, ich weiß nicht, 150.000 US-Dollar verkauft werden. Das Anpassen einer geraden Linie ist jedoch nicht der einzige Lernalgorithmus, den Sie verwenden können. Es gibt andere, die für diese Anwendung besser funktionieren könnten. Wenn Sie beispielsweise eine gerade Linie verlegen und anpassen, entscheiden Sie möglicherweise, dass es besser ist, eine Kurve anzupassen, **eine Funktion, die etwas komplizierter oder komplexer ist als eine gerade Linie**.

Wenn Sie das tun und hier eine Vorhersage treffen, sieht es so aus, als ob das Haus Ihres Freundes für etwa 200.000 US-Dollar verkauft werden könnte.

Nun scheint es nicht angemessen zu sein, diejenige auszuwählen, die Ihrem Freund den besten Preis bietet, aber Sie sehen, wie man einen Algorithmus dazu bringt, systematisch die am besten geeignete Linie, Kurve oder andere Sache auszuwählen, die zu diesen Daten passt. Was Sie auf dieser Folie gesehen haben, ist ein Beispiel für überwachtes Lernen. Denn wir haben dem Algorithmus einen Datensatz gegeben, in dem für jedes Haus auf dem Grundstück die sogenannte richtige Antwort, also die Beschriftung bzw. der richtige Preis y angegeben ist. Die Aufgabe des Lernalgorithmus besteht darin, mehr dieser richtigen Antworten zu generieren, insbesondere den wahrscheinlichen Preis für andere Häuser wie das Haus Ihres Freundes vorherzusagen. Deshalb handelt es sich hier um **überwachtes Lernen**.

Um die Terminologie etwas genauer zu definieren: Diese Immobilienpreisvorhersage ist eine besondere Art des überwachten Lernens, die als Regression bezeichnet wird. Mit Regression meine ich, dass wir versuchen, eine Zahl aus unendlich vielen möglichen Zahlen vorherzusagen, wie zum Beispiel die Hauspreise in unserem Beispiel, die 150.000 oder 70.000 oder 183.000 oder jede andere Zahl dazwischen sein könnten. Dabei handelt es sich um überwachtes Lernen, Lerneingaben, -ausgaben oder X- zu-Y- Zuordnungen. In diesem Video haben Sie ein Beispiel für eine Regression gesehen, bei der die Aufgabe darin besteht, eine Zahl vorherzusagen. Es gibt aber auch eine zweite große Art von überwachtem Lernproblem, **die Klassifizierung**. Schauen wir uns im nächsten Video an, was das bedeutet.

Überwachtes Lernen Teil 2

Überwachte Lernalgorithmen lernen also, Eingaben, Ausgaben oder X-zu-Y-Zuordnungen vorherzusagen. Und im letzten Video haben Sie gesehen, dass Regressionsalgorithmen, eine Art überwachter Lernalgorithmus, lernen, Zahlen aus unendlich vielen möglichen Zahlen vorherzusagen. Es gibt einen zweiten Haupttyp von überwachten Lernalgorithmen, den sogenannten **Klassifizierungsalgorithmus**. Werfen wir einen Blick darauf, was das bedeutet. Nehmen Sie die Brustkrebserkennung als Beispiel für ein Klassifizierungsproblem.

Anhand der Krankenakten eines Patienten versucht Ihr maschinelles Lernsystem herauszufinden, ob ein Tumor, bei dem es sich um einen Knoten handelt, bösartig, also krebsartig oder gefährlich, ist. Oder wenn dieser Tumor, dieser Knoten gutartig ist, was bedeutet, dass es sich nur um einen Knoten handelt, der nicht krebsartig und nicht so gefährlich ist? Einige meiner Freunde haben tatsächlich an diesem speziellen Problem gearbeitet. Vielleicht enthält Ihr Datensatz also Tumoren unterschiedlicher Größe. Und diese Tumoren werden entweder als gutartig gekennzeichnet, was ich in diesem Beispiel mit einer 0 bezeichne, oder als bösartig, was in diesem Beispiel mit einer 1 gekennzeichnet werde. Anschließend können Sie Ihre Daten in einem Diagramm wie diesem darstellen, wobei die horizontale Achse die Größe darstellt des Tumors und die vertikale Achse nimmt nur zwei Werte an: 0 oder 1, je nachdem, ob der Tumor gutartig ist, 0 oder bösartig 1.

Ein Grund dafür, **dass sich dies von der Regression unterscheidet, besteht darin, dass wir versuchen, nur eine kleine Anzahl möglicher Ausgaben oder Kategorien vorherzusagen**. In diesem Fall gibt es zwei mögliche Ausgänge: 0 oder 1, gutartig oder bösartig. Dies unterscheidet sich von der Regression, die versucht, eine beliebige Zahl vorherzusagen, und zwar alle der unendlich vielen möglichen Zahlen. Ausschlaggebend für diese Klassifizierung ist also die Tatsache, dass es nur zwei mögliche Ergebnisse gibt. Da es in diesem Beispiel nur zwei mögliche Ausgaben bzw. zwei mögliche Kategorien gibt, können Sie diesen Datensatz auch auf einer Linie wie dieser darstellen. Im Moment werde ich zwei verschiedene Symbole verwenden, um die Kategorie zu kennzeichnen: einen Kreis und ein O, um die gutartigen Beispiele zu kennzeichnen, und ein Kreuz, um die bösartigen Beispiele zu kennzeichnen. Und wenn neue Patienten zur Diagnose kommen und einen Knoten dieser Größe haben, dann stellt sich die Frage: Wird Ihr System diesen Tumor als gutartig oder bösartig einstufen? Es stellt sich heraus, dass es bei Klassifizierungsproblemen auch mehr als zwei mögliche Ausgabekategorien geben kann. Vielleicht lernen Sie, dass ein Algorithmus mehrere Arten

von Krebsdiagnosen ausgeben kann, wenn sich herausstellt, dass er bösartig ist. Nennen wir also zwei verschiedene Krebsarten Typ 1 und Typ 2.

In diesem Fall hätte der Durchschnitt **drei mögliche Ausgabekategorien**, die er vorhersagen könnte. Zusammenfassend lässt sich sagen, dass Klassifizierungsalgorithmen Kategorien vorhersagen. Kategorien müssen keine Zahlen sein. Es könnte beispielsweise nicht numerisch sein und vorhersagen, ob es sich bei einem Bild um das einer Katze oder eines Hundes handelt. Und es kann vorhersagen, ob ein Tumor gutartig oder bösartig ist. Kategorien können auch Zahlen wie 0, 1 oder 0, 1, 2 sein.

Was die Klassifizierung bei der Interpretation der Zahlen jedoch von der Regression unterscheidet, ist, **dass die Klassifizierung einen kleinen endlichen begrenzten Satz möglicher Ausgabekategorien wie 0, 1 und 2 vorhersagt** aber nicht alle möglichen Zahlen dazwischen wie 0,5 oder 1,7. In dem Beispiel des überwachten Lernens, das wir uns angesehen haben, hatten wir nur einen Eingabewert mit der Größe des Tumors. Sie können aber auch mehr als einen Eingabewert verwenden, um eine Ausgabe vorherzusagen. Hier ist ein Beispiel: Anstatt nur die Tumorgröße zu kennen , sagen Sie, dass Sie auch das Alter jedes Patienten in Jahren kennen. Ihr neuer Datensatz verfügt nun über zwei Eingaben: Alter und Tumorgröße. In diesem neuen Datensatz verwenden wir Kreise, um Patienten anzuzeigen, deren Tumoren gutartig sind, und Kreuze, um Patienten mit einem bösartigen Tumor anzuzeigen.

Wenn also ein neuer Patient eintrifft, kann der Arzt die Tumorgröße des Patienten messen und auch das Alter des Patienten erfassen. Wie können wir vor diesem Hintergrund vorhersagen, ob der Tumor dieses Patienten gutartig oder bösartig ist? Nun, angesichts der heutigen Situation könnte der Lernalgorithmus eine Grenze finden, die die bösartigen von den gutartigen Tumoren trennt. Der Lernalgorithmus muss also entscheiden, wie eine Grenzlinie durch diese Daten angepasst werden soll. Die vom Lernalgorithmus gefundene Grenzlinie würde dem Arzt bei der Diagnose helfen. In diesem Fall ist es wahrscheinlicher, dass der Tumor gutartig ist.

Anhand dieses Beispiels haben wir gesehen, wie Eingaben zum Alter und zur Tumorgröße des Patienten verwendet werden können. Bei anderen maschinellen Lernproblemen sind oft viel mehr Eingabewerte erforderlich. Meine Freunde, die an der Brustkrebserkennung gearbeitet haben, verwenden viele zusätzliche Eingaben, wie die Dicke des Tumorklumpens, die Gleichmäßigkeit der Zellgröße, die Gleichmäßigkeit der Zellform und so weiter. Um es noch einmal zusammenzufassen: Das überwachte Lernen ordnet den Eingang x dem Ausgang y zu, wobei der Lernalgorithmus aus dem Zitat die richtigen Antworten lernt.

Die beiden wichtigsten Arten des überwachten Lernens sind Regression und Klassifizierung. In einer Regressionsanwendung wie der Vorhersage von Hauspreisen muss der Lernalgorithmus Zahlen aus unendlich vielen möglichen Ausgabezahlen vorhersagen. Während bei der Klassifizierung der Lernalgorithmus eine Vorhersage einer Kategorie treffen muss, also aller einer kleinen Menge möglicher Ausgaben.

Unüberwachtes Lernen Teil 1

Nach dem überwachten Lernen ist das unüberwachte Lernen die am weitesten verbreitete Form des maschinellen Lernens.

Jedem Beispiel war eine Ausgabebezeichnung y zugeordnet, beispielsweise „gutartig“ oder „bösartig“, die beim unüberwachten Lernen durch die Pole und Kreuze gekennzeichnet wurde. Es wurden Daten bereitgestellt, die nicht mit Ausgabebezeichnungen verknüpft sind. Nehmen wir an, Sie erhalten Daten zu Patienten, ihrer Tumorgröße und dem Alter des Patienten. Allerdings nicht, ob der Tumor gutartig oder bösartig war, daher sieht der Datensatz rechts so aus. Unsere Aufgabe besteht stattdessen darin, **im Datensatz eine Struktur oder ein Muster zu finden.** Das ist unüberwachtes Lernen.

Ein unbeaufsichtigter Lernalgorithmus könnte entscheiden, dass die Daten zwei verschiedenen Gruppen oder zwei verschiedenen Clustern zugeordnet werden können. Und so könnte es entscheiden, dass es hier einen Cluster oder eine Gruppe gibt und dass es hier einen anderen Cluster oder eine andere Gruppe gibt. Hierbei handelt es sich um eine besondere Art des unbeaufsichtigten Lernens, die als **Clustering-Algorithmus** bezeichnet wird. Weil es die unbeschrifteten Daten in verschiedene Cluster einordnet und sich herausstellt, dass dies in vielen Anwendungen verwendet wird.

Hier ist zum Beispiel ein Beispiel aus Google News, wo die Überschrift des Top-Artikels lautet: Großer Panda bringt in Japans ältestem Zoo zwei Zwillinge zur Welt. Dieser Artikel ist mir tatsächlich aufgefallen, weil meine Tochter Pandas liebt und es daher jede Menge Panda-Spielzeuge gibt. Wenn Sie sich bei mir zu Hause Panda-Videos ansehen und sich das ansehen, werden Sie vielleicht feststellen, dass sich darunter weitere verwandte Artikel befinden. Vielleicht können Sie allein anhand der Schlagzeilen erahnen, was Clustering bewirken könnte. Beachten Sie, dass das Wort Panda hier, hier, hier und hier vorkommt und dass das Wort Zwilling auch in allen fünf Artikeln vorkommt. Und das Wort Zoo kommt auch in all diesen Artikeln vor, sodass der Clustering-Algorithmus Artikel findet. Alle Hunderttausende Nachrichtenartikel im Internet an diesem Tag, die Artikel finden, in denen ähnliche Wörter vorkommen, und sie in Cluster gruppieren.

Das Coole ist, dass dieser Clustering-Algorithmus selbstständig herausfindet, welche Wörter darauf hindeuten, dass bestimmte Artikel zur selben Gruppe gehören. Ich meine damit, dass es keinen Mitarbeiter bei Google News gibt, der dem Algorithmus sagt, er solle Artikel finden, die das Wort „Panda“ enthalten. Und Zwillinge und Zoo, um sie in denselben Cluster einzuordnen, ändern sich die Nachrichtenthemen jeden Tag. Und es gibt so viele Nachrichten, dass es einfach nicht möglich ist, dies jeden Tag für alle Themen zu tun, die Cover verwenden.

Stattdessen muss der Algorithmus selbstständig und ohne Aufsicht herausfinden, welche Cluster von Nachrichtenartikeln es heute gibt. Aus diesem Grund handelt es sich bei diesem Clustering-Algorithmus um eine Art unbeaufsichtigten Lernalgorithmus. Schauen wir uns das zweite Beispiel für unbeaufsichtigtes Lernen an, das auf die Clusterung genetischer oder DNA-Daten angewendet wird. Dieses Bild zeigt ein Bild von DNA-Mikroarray-Daten, diese

sehen aus wie winzige Raster einer Tabellenkalkulation. Und jede winzige Spalte repräsentiert die genetische oder DNA-Aktivität einer Person.

So stammt zum Beispiel diese gesamte Kolumne hier aus der DNA einer Person. Und diese andere Spalte ist von einer anderen Person, jede Zeile repräsentiert ein bestimmtes Gen. Nur als Beispiel: Vielleicht repräsentiert diese Rolle hier ein Gen, das die Augenfarbe beeinflusst , oder diese Rolle hier ist ein Gen, das beeinflusst, wie groß jemand ist. Forscher haben sogar einen genetischen Zusammenhang dafür gefunden, ob jemand bestimmte Gemüsesorten wie Brokkoli, Rosenkohl oder Spargel nicht mag. Wenn dich also das nächste Mal jemand fragt, warum du deinen Salat nicht aufgegessen hast, kannst du ihm sagen, vielleicht liegt es an der DNA-Mikrorasse.

Die Idee besteht darin, zu messen, wie stark bestimmte Gene bei jeder einzelnen Person exprimiert werden. Diese Farben Rot, Grün, Grau usw. zeigen also den Grad an, in dem verschiedene Individuen ein bestimmtes Gen aktiv haben oder nicht. Anschließend können Sie einen Clustering-Algorithmus ausführen, um Personen in verschiedene Kategorien einzuteilen. Oder verschiedene Arten von Menschen, wie vielleicht diese Individuen, die sich zusammenschließen, und nennen wir diesen Typ einfach einen. Und diese Menschen werden in Typ zwei eingeteilt, und diese Menschen werden in Typ drei eingeteilt. Dabei handelt es sich um unbeaufsichtigtes Lernen, da wir dem Algorithmus nicht im Voraus mitteilen, dass es eine Person vom Typ 1 mit bestimmten Eigenschaften gibt.

Oder eine Person vom Typ 2 mit bestimmten Eigenschaften. Stattdessen sagen wir hier eine Menge Daten. Ich kenne die verschiedenen Arten von Menschen nicht, aber kann man Daten automatisch strukturieren? Und finden Sie automatisch heraus, ob es sich um die wichtigsten Personentypen handelt, **da wir dem Algorithmus nicht im Voraus die richtige Antwort für die Beispiele geben**. Dies ist unbeaufsichtigtes Lernen. Hier ist das dritte Beispiel. Viele Unternehmen verfügen aufgrund dieser Daten über riesige Datenbanken mit Kundeninformationen. Können Sie Ihre Kunden automatisch in verschiedene Marktsegmente gruppieren, damit Sie Ihre Kunden effizienter bedienen können? Konkret hat das Deep-Learning-Dot-AI-Team einige Nachforschungen angestellt, um die Deep-Learning-Dot-AI-Community besser zu verstehen. Und warum verschiedene Personen an diesen Kursen teilnehmen, den wöchentlichen Batch-Newsletter abonnieren oder an unseren KI-Veranstaltungen teilnehmen.

Unüberwachtes Lernen Teil 2

Im letzten Video haben Sie gesehen, was unbeaufsichtigtes Lernen ist und eine Art **unbeaufsichtigten Lernens namens Clustering**. Lassen Sie uns eine etwas formalere Definition des unbeaufsichtigten Lernens geben und einen kurzen Blick auf einige andere Arten des unbeaufsichtigten Lernens außer Clustering werfen.

Während beim überwachten Lernen die Daten sowohl Eingaben x als auch Ausgabebezeichnungen y enthalten, enthalten die Daten beim unüberwachten Lernen **nur Eingaben x, aber keine Ausgabebezeichnungen y**, und der Algorithmus muss eine Struktur, ein Muster oder etwas Interessantes darin finden. Wir sehen gerade ein Beispiel für

unbeaufsichtigtes Lernen, einen sogenannten Clustering-Algorithmus, der ähnliche Datenpunkte gruppiert.

In dieser Spezialisierung lernen Sie Clustering sowie zwei weitere Arten des unbeaufsichtigten Lernens kennen. Eine davon heißt **Anomalieerkennung** und dient der Erkennung ungewöhnlicher Ereignisse. Dies erweist sich als sehr wichtig für die Betrugserkennung im Finanzsystem, wo ungewöhnliche Ereignisse und ungewöhnliche Transaktionen Anzeichen von Betrug sein können, und für viele andere Anwendungen. Sie lernen auch etwas über Dimensionsreduktion. Auf diese Weise können Sie einen großen Datensatz nehmen und ihn auf fast magische Weise zu einem viel kleineren Datensatz komprimieren und dabei so wenig Informationen wie möglich verlieren.

Für den Fall, dass **Anomalieerkennung und Dimensionsreduzierung** für Sie noch nicht allzu sinnvoll erscheinen.

Vielleicht erinnern Sie sich an das Problem mit der Spam-Filterung. Wenn Sie Daten gekennzeichnet haben, die Sie jetzt als Spam- oder Nicht-Spam-E-Mail kennzeichnen, können Sie dies als überwachtes Lernproblem behandeln. Das zweite Beispiel, das Nachrichtenbeispiel. Das ist genau das Google News- und konkrete Beispiel, das Sie im letzten Video gesehen haben. Sie können dies erreichen, indem Sie einen Clustering-Algorithmus verwenden, um Nachrichtenartikel zu gruppieren.

Dass wir unüberwachtes Lernen nutzen werden. Das Beispiel der Marktsegmentierung, über das ich vorhin gesprochen habe. Sie können dies auch als unbeaufsichtigtes Lernproblem tun, da Sie Ihrem Algorithmus einige Daten geben und ihn auffordern können, Marktsegmente automatisch zu entdecken. Das letzte Beispiel zur Diagnose von Diabetes. Nun ja, eigentlich ist das unserem Brustkrebs-Beispiel aus den betreuten Lernvideos sehr ähnlich. Nur haben wir anstelle von gutartigen oder bösartigen Tumoren Diabetes oder keinen Diabetes.

Sie können dies als überwachtes Lernproblem betrachten, genau wie wir es für das Brusttumor- Klassifizierungsproblem getan haben. Auch wenn wir im letzten Video hauptsächlich über Clustering gesprochen haben, werden wir in späteren Videos in dieser Spezialisierung auch viel tiefer auf die Erkennung von Anomalien und die Reduzierung der Dimensionalität eingehen. Das ist unbeaufsichtigtes Lernen

Jupyter- Notizbücher

Das heute von Praktikern des maschinellen Lernens und der **Datenwissenschaft am häufigsten verwendete Tool ist das Jupyter Notebook**. Dies sind die Standardumgebungen, die viele von uns zum Programmieren, Experimentieren und Ausprobieren verwenden. In diesem Kurs erstellen Sie direkt hier in Ihrem Webbrowser eine Jupyter Notebook-Benutzerumgebung, um einige dieser Ideen auch selbst zu testen. Dabei handelt es sich nicht um eine erfundene vereinfachte Umgebung, sondern um genau dieselben Umgebungen, genau dasselbe Tool, das Jupyter Notebook, das Entwickler derzeit in vielen großen Ländern verwenden.

Regressionsmodell

Lineare Regression Modell Teil 1

In diesem Video schauen wir uns den Gesamtprozess des überwachten Lernens an. Konkret sehen Sie das erste Modell dieses Kurses, das lineare Regressionsmodell. Das bedeutet lediglich, eine gerade Linie an Ihre Daten anzupassen. Es ist heute wahrscheinlich der am weitesten verbreitete Lernalgorithmus der Welt.

Hier haben wir ein Diagramm, in dem die horizontale Achse die Größe des Hauses in Quadratfuß und die vertikale Achse den Preis eines Hauses in Tausend Dollar darstellt. Lassen Sie uns fortfahren und die Datenpunkte für verschiedene Häuser im Datensatz grafisch darstellen.

Hier ist jeder Datenpunkt, jedes dieser kleinen Kreuze ein Haus mit der Größe und dem Preis, für den es zuletzt verkauft wurde. Nehmen wir an, Sie sind Immobilienmakler in Portland und helfen einer Kundin, ihr Haus zu verkaufen. Sie fragt Sie: Wie viel kann ich Ihrer Meinung nach für dieses Haus bekommen? Dieser Datensatz könnte Ihnen dabei helfen, den Preis abzuschätzen, den sie dafür erhalten könnte. Sie beginnen damit, die Größe des Hauses zu messen, und es stellt sich heraus, dass das Haus 1250 Quadratmeter groß ist. Für wie viel könnte dieses Haus Ihrer Meinung nach verkauft werden? Eine Möglichkeit besteht darin, aus diesem Datensatz ein lineares Regressionsmodell zu erstellen. Ihr Modell passt eine gerade Linie an die Daten an, die so aussehen könnte. Basierend auf dieser geraden Linienanpassung an die Daten können Sie erkennen, dass das Haus 1250 Quadratfuß groß ist. Es schneidet hier die beste Anpassungslinie, und wenn Sie dies auf der vertikalen Achse auf der linken Seite verfolgen, können Sie den Preis erkennen hier vielleicht etwa 220.000 US-Dollar.

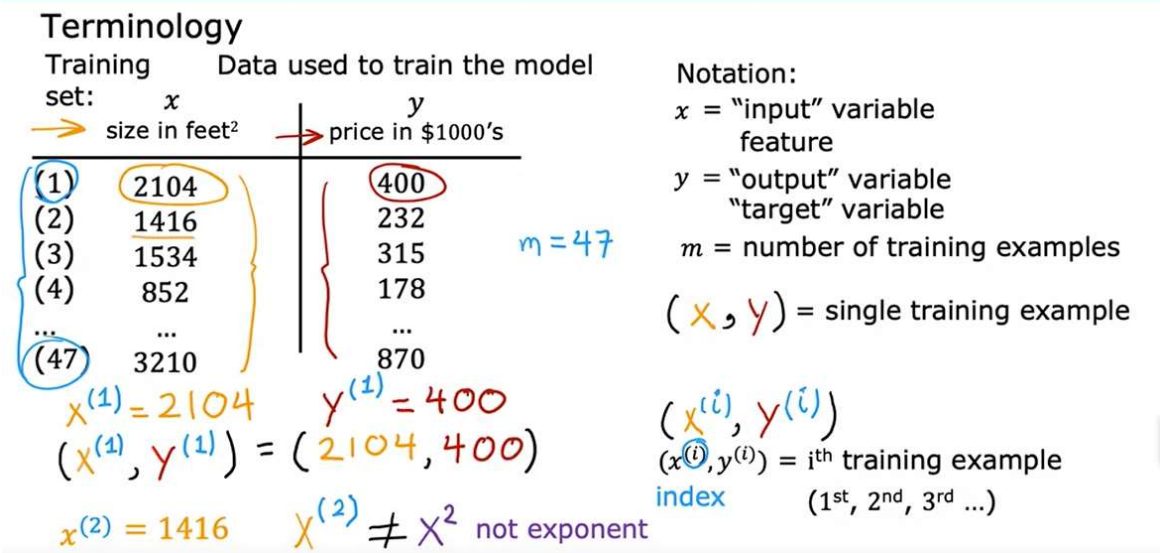
Dies ist ein Beispiel für ein sogenanntes überwachtes Lernmodell. Wir nennen dieses überwachtes Lernen, **weil Sie zunächst ein Modell trainieren**, indem Sie Daten angeben, die die richtigen Antworten liefern, weil Sie die Modellbeispiele von Häusern sowohl mit der Größe des Hauses als auch mit dem Preis erhalten, den das Modell für jedes Haus vorhersagen sollte.

Schauen wir uns nun einige Notationen zur Beschreibung der Daten an. Dies ist eine Notation, die Sie auf Ihrer Reise zum maschinellen Lernen als nützlich erachten. Wenn Sie sich zunehmend mit der Terminologie des maschinellen Lernens vertraut machen, können Sie diese Terminologie auch verwenden, um mit anderen über Konzepte des maschinellen Lernens zu sprechen, da vieles davon in der gesamten KI ziemlich Standard ist und Sie diese Notation in dieser Spezialisierung mehrmals sehen werden Es ist also in Ordnung, wenn Sie sich nicht alles merken, um es zuzuordnen. Mit der Zeit wird es Ihnen natürlich immer vertrauter.

Der Datensatz, den Sie gerade gesehen haben und der zum Trainieren des Modells verwendet wird, wird als Trainingssatz bezeichnet. Beachten Sie, dass das Haus Ihres Kunden nicht in diesem Datensatz enthalten ist, da es noch nicht verkauft ist und daher niemand weiß, wie hoch der Preis ist. Um den Preis des Hauses Ihres Kunden vorherzusagen,

trainieren Sie zunächst Ihr Modell, um aus dem Trainingssatz zu lernen, und dieses Modell kann dann den Hauspreis Ihres Kunden vorhersagen.

Beim maschinellen Lernen ist die Standardschreibweise zur Bezeichnung der Eingabe hier der Kleinbuchstabe x, und wir nennen dies die Eingabevariable, die auch als Feature oder Eingabemerkmal bezeichnet wird. Für das erste Haus in Ihrem Trainingssatz ist x beispielsweise die Größe des Hauses, also entspricht x 2.104. Die Standardschreibweise zur Bezeichnung der Ausgabevariablen, die Sie vorhersagen möchten, die manchmal auch als Zielvariable bezeichnet wird, ist der Kleinbuchstabe y. Hier ist y der Preis des Hauses, und für das erste Trainingsbeispiel ist dieser gleich 400, also ist y gleich 400. Der Datensatz hat eine Zeile für jedes Haus und in diesem Trainingssatz gibt es 47 Zeilen, wobei jede Zeile repräsentiert ein anderes Trainingsbeispiel. Wir verwenden das Kleinbuchstabe „m“, um es auf die Gesamtzahl der Trainingsbeispiele zu verweisen, sodass m hier gleich 47 ist.



Um das einzelne Trainingsbeispiel zu kennzeichnen, verwenden wir die Notationsklammern x, y. Für das erste Trainingsbeispiel (x, y) ist dieses Zahlenpaar (2104, 400). Jetzt haben wir viele verschiedene Trainingsbeispiele. Tatsächlich haben wir 47 davon. Um auf ein bestimmtes Trainingsbeispiel zu verweisen, entspricht dies einer bestimmten Zeile in dieser Tabelle links. Ich verwende die Notation x hochgestellt in Klammern, i, y hochgestellt in Klammern i.

Der hochgestellte Index sagt uns, dass dies das i-te Trainingsbeispiel ist, also das erste, zweite oder dritte bis hin zum 47. Trainingsbeispiel. Ich beziehe mich hier auf eine bestimmte Zeile in der Tabelle. Hier ist zum Beispiel das erste Beispiel, wenn i im Trainingssatz gleich 1 ist und x hochgestellt 1 also gleich 2.104 und y hochgestellt 1 gleich 400 ist. Fügen wir diese hochgestellte 1 auch hier hinzu. Bitte beachten Sie, dass es sich bei diesem hochgestellten i in Klammern nicht um eine Potenzierung handelt. Wenn ich das schreibe, ist das nicht x im Quadrat. Dies ist nicht x hoch 2. Es bezieht sich nur auf das zweite Trainingsbeispiel. Dieses i ist nur ein Index in den Trainingssatz und bezieht sich auf Zeile i in der Tabelle. In diesem Video haben Sie gesehen, wie ein Trainingssatz aussieht, sowie eine Standardnotation zur Beschreibung dieses Trainingssatzes. Schauen wir uns im nächsten Video an, wie Sie diesen Trainingssatz, den Sie gerade gesehen haben, rotieren und ihn dem

Lernalgorithmus zuführen, damit der Algorithmus aus diesen Daten lernen kann. Das sehen wir uns im nächsten Video an.

Lineares Regressionsmodell Teil 2

Um das Modell zu trainieren, geben Sie den Trainingssatz, sowohl die Eingabemerkmale als auch die Ausgabeziele, an Ihren Lernalgorithmus weiter. Dann erzeugt Ihr überwachter Lernalgorithmus eine Funktion. Wir schreiben diese Funktion als Kleinbuchstabe f, wobei f für Funktion steht. Historisch gesehen wurde diese Funktion als Hypothese bezeichnet, aber ich werde sie in dieser Klasse einfach als Funktion f bezeichnen.

Die Aufgabe von f besteht darin, eine neue Eingabe x zu nehmen und eine Schätzung oder eine Vorhersage auszugeben, die ich ŷ nennen werde, und sie wird wie die Variable y mit diesem kleinen Hutsymbol oben geschrieben. Beim maschinellen Lernen gilt die Konvention, dass ŷ die Schätzung oder Vorhersage für y ist. Die Funktion f heißt Modell. X wird als Eingabe oder Eingabemerkmal bezeichnet, und die Ausgabe des Modells ist die Vorhersage, y-hat. Die Vorhersage des Modells ist der geschätzte Wert von y.

Wenn das Symbol nur der Buchstabe y ist, bezieht sich das auf das Ziel, das der tatsächliche wahre Wert im Trainingssatz ist. Im Gegensatz dazu handelt es sich bei ŷ um eine Schätzung. Es kann der tatsächliche wahre Wert sein oder auch nicht. Nun, wenn Sie Ihrem Kunden helfen, das Haus zu verkaufen, ist der wahre Preis des Hauses unbekannt, bis er es verkauft. Ihr Modell f gibt bei gegebener Größe den Preis aus, der der Schätzer ist, also die Vorhersage des wahren Preises. Wenn wir nun einen Lernalgorithmus entwerfen, ist eine Schlüsselfrage: Wie stellen wir die Funktion f dar? Oder mit anderen Worten: **Welche mathematische Formel werden wir verwenden, um f zu berechnen?**

Bleiben wir zunächst bei f als gerader Linie. Ihre Funktion kann als f_w geschrieben werden, b von x ist gleich, ich werde w mal x plus b verwenden. Ich werde w und b bald definieren.

Aber vorerst müssen Sie nur wissen, dass w und b Zahlen sind und die für w und b **gewählten Werte die Vorhersage ŷ** basierend auf dem Eingabemerkmal x bestimmen. Dieses f w b von x bedeutet, dass f eine Funktion ist, die x als Eingabe verwendet, und abhängig von den Werten von w und b gibt f einen Wert einer Vorhersage ŷ aus. Als Alternative dazu, f_w , b von x zu schreiben, schreibe ich manchmal einfach f von x, ohne w und b explizit in den Index aufzunehmen. Ist nur eine einfachere Notation, die genau dasselbe bedeutet wie f w, b von x.

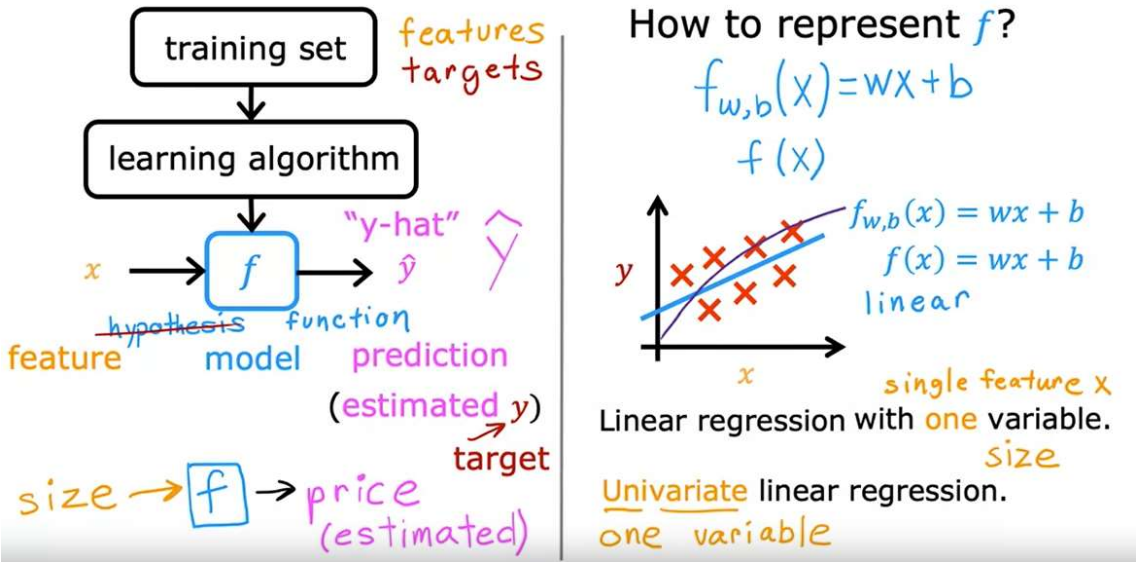
Stellen wir den Trainingssatz im Diagramm dar, wobei sich das Eingabemerkmal x auf der horizontalen Achse und das Ausgabeziel y auf der vertikalen Achse befindet. Denken Sie daran, dass der Algorithmus aus diesen Daten lernt und die am besten passende Linie generiert, wie vielleicht diese hier. Diese Gerade ist die lineare Funktion f_w b von x gleich w mal x plus b. Oder einfacher gesagt, wir können w und b weglassen und einfach schreiben, dass $f(x) = wx + b$ ist.

Hier sehen Sie, was diese Funktion tut: Sie macht Vorhersagen für den Wert von y mithilfe einer Streamline-Funktion von x. Sie fragen sich vielleicht, warum wir eine lineare Funktion wählen, wenn die lineare Funktion nur ein schicker Begriff für eine gerade Linie ist und nicht

für eine nichtlineare Funktion wie eine Kurve oder eine Parabel? Nun, manchmal möchte man auch komplexere nichtlineare Funktionen anpassen, wie zum Beispiel eine Kurve wie diese.

Da diese lineare Funktion jedoch relativ einfach und leicht zu handhaben ist, verwenden wir eine Linie als Grundlage, die Ihnen letztendlich dabei helfen wird, zu komplexeren, nichtlinearen Modellen zu gelangen. **Dieses spezielle Modell hat einen Namen, es heißt lineare Regression**. Genauer gesagt handelt es sich um eine lineare Regression mit einer Variablen, wobei der Ausdruck „eine Variable“ bedeutet, dass es eine einzelne Eingabevariable oder ein einzelnes Merkmal x gibt, nämlich die Größe des Hauses.

Ein anderer Name für ein lineares Modell mit einer Eingabevariablen ist **univariate lineare Regression**, wobei „ uni“ auf Lateinisch „eins“ und „variate“ „Variable“ bedeutet.



Im Labor können Sie die Werte von w und b auswählen, um zu versuchen, sie an die Trainingsdaten anzupassen.. Das ist eine lineare Regression. Damit dies funktioniert, müssen Sie vor allem eine **Kostenfunktion konstruieren**.

Die Idee einer Kostenfunktion ist eine der universellsten und wichtigsten Ideen im maschinellen Lernen und wird sowohl in der linearen Regression als auch beim Training vieler der fortschrittlichsten KI-Modelle der Welt verwendet. Fahren wir mit dem nächsten Video fort und schauen wir uns an, wie Sie eine Kostenfunktion erstellen können.

Kostenfunktionsformel

Um eine lineare Regression zu implementieren, besteht der erste wichtige Schritt darin, zunächst eine sogenannte **Kostenfunktion zu definieren**. Dies ist etwas, was wir in diesem Video einbauen werden, und die Kostenfunktion wird uns sagen, **wie gut das Modell funktioniert**, sodass wir versuchen können, es besser zu machen.

Denken Sie daran, dass Sie über einen Trainingssatz verfügen, der Eingabemerkmale x und Ausgabeziele y enthält. Das Modell, das Sie zur Anpassung dieses Trainingssatzes verwenden werden, ist diese lineare Funktion $f_{w,b}$ von x = w*x + b. Um etwas mehr Terminologie

einzuführen, werden **w und b als Parameter des Modells** bezeichnet. Beim maschinellen Lernen sind **Modellparameter die Variablen, die Sie während des Trainings anpassen können, um das Modell zu verbessern**. Manchmal werden die Parameter w und b auch als **Koeffizienten oder Gewichte** bezeichnet. Schauen wir uns nun an, was diese Parameter w und b bewirken. Abhängig von den Werten, die Sie für w und b gewählt haben, erhalten Sie eine andere Funktion f von x, die eine andere Linie im Diagramm erzeugt.

Denken Sie daran, dass wir **f(x) als Abkürzung für** $f_{w, b}(x)$ **schreiben können**. Wir werden uns einige Diagramme von f(x) in einem Diagramm ansehen. Vielleicht sind Sie bereits mit dem Zeichnen von Linien in Diagrammen vertraut, aber auch wenn dies eine Rezension für Sie ist, hoffe ich, dass dies Ihnen dabei hilft, ein Verständnis dafür zu entwickeln, wie die Parameter w und b von f bestimmen.

Wenn w gleich 0 und b gleich 1,5 ist, dann sieht f wie diese horizontale Linie aus. In diesem Fall ist die Funktion f(x) 0 mal x plus 1,5, also ist f immer ein konstanter Wert. Für den geschätzten Wert von y wird immer 1,5 vorhergesagt. Ŷ ist immer gleich b und hier wird b auch y-Achsenabschnitt genannt, weil es dort die vertikale Achse oder die y-Achse in diesem Diagramm schneidet.

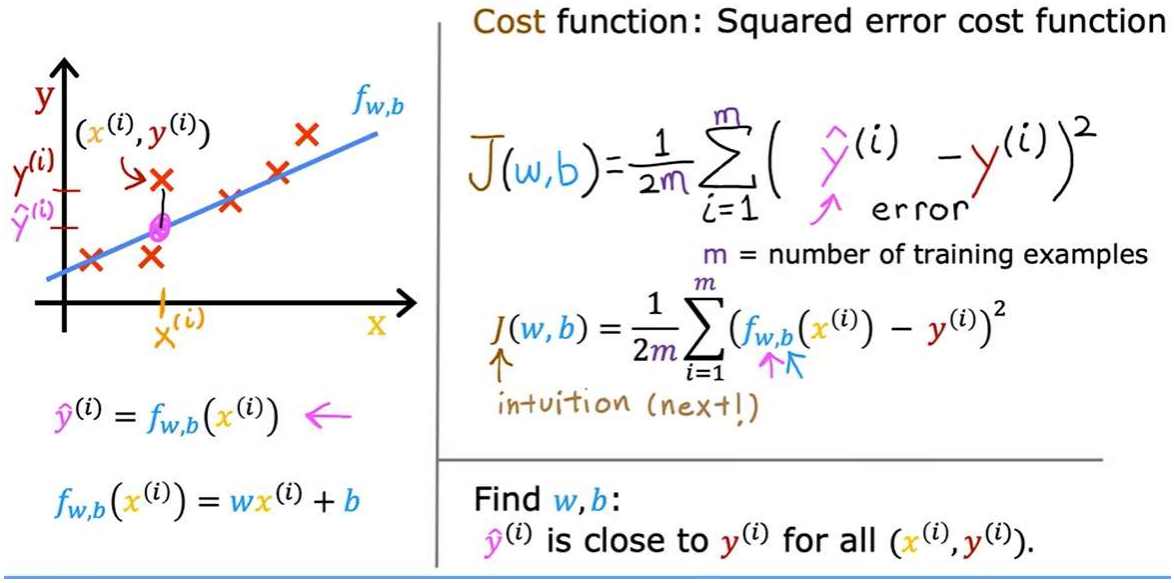
Als zweites Beispiel: Wenn w 0,5 und b gleich 0 ist, dann ist f(x) 0,5 mal x. Wenn x 0 ist, ist die Vorhersage auch 0, und wenn x 2 ist, beträgt die Vorhersage 0,5 mal 2, also 1. Sie erhalten eine Linie, die so aussieht, und stellen fest, dass die Steigung 0,5 geteilt durch 1 beträgt. Der Wert von w gibt Ihnen die Steigung der Linie, die 0,5 beträgt. Wenn schließlich w gleich 0,5 und b gleich 1 ist, dann ist f(x) 0,5 mal x plus 1, und wenn x 0 ist, dann ist f(x) gleich b, was 1 ist, sodass die Linie die vertikale Achse bei b, dem y-Achsenabschnitt, schneidet . Auch wenn x 2 ist, dann ist f(x) 2, sodass die Linie so aussieht. Auch diese Steigung beträgt 0,5 dividiert durch 1, sodass der Wert von w die Steigung von 0,5 ergibt. Denken Sie daran, dass Sie über ein Trainingsset wie das hier gezeigte verfügen.

Bei der linearen Regression möchten Sie Werte für die Parameter w und b so wählen, dass die gerade Linie, die Sie aus der Funktion f erhalten, irgendwie gut zu den Daten passt. Wie vielleicht diese hier gezeigte Zeile.

Wenn ich sehe, dass die Linie visuell zu den Daten passt, kann man sich das so vorstellen, dass die durch f definierte Linie ungefähr durch oder irgendwo in der Nähe der Trainingsbeispiele verläuft, im Vergleich zu anderen möglichen Linien, die nicht so nah an diesen Punkten liegen. Um Sie an die Notation zu erinnern: Ein Trainingsbeispiel wie dieses hier wird durch x_i und y_i definiert , wobei y das Ziel ist. Für eine gegebene Eingabe x^i erstellt die Funktion f auch einen Vorhersagewert für y und einen Wert, den sie für y vorhersagt, wie hier gezeigt. Für unsere Wahl eines Modells ist f von x^i w mal x^i plus b.

Anders ausgedrückt ist die Vorhersage ŷ i f von wb von x^i , wobei f von x^i für das von uns verwendete Modell gleich wx^i plus b ist. Die Frage ist nun, wie man Werte für w und b findet, damit die Vorhersage ŷ i für viele oder vielleicht alle Trainingsbeispiele x ^i , y^i nahe am wahren Ziel y^i liegt . Um diese Frage zu beantworten, werfen wir zunächst einen Blick darauf, wie man misst, wie gut eine Linie zu den Trainingsdaten passt.

Dazu erstellen wir eine Kostenfunktion. Die Kostenfunktion nimmt das vorhergesagte ŷ und vergleicht es mit dem Ziel y, indem sie ŷ minus y nimmt. Dieser Unterschied wird als Fehler bezeichnet. **Wir messen, wie weit die Vorhersage vom Ziel entfernt ist**.



Als nächstes berechnen wir das **Quadrat dieses Fehlers**. Außerdem möchten wir diesen Term für verschiedene Trainingsbeispiele i im Trainingssatz berechnen. Wenn wir den Fehler messen, zum Beispiel i, berechnen wir diesen quadrierten Fehlerterm. Schließlich wollen wir den Fehler über den gesamten Trainingssatz messen. Fassen wir insbesondere die quadratischen Fehler so zusammen. Wir summieren von i gleich 1,2, 3 bis hin zu m und denken daran, dass **m die Anzahl der Trainingsbeispiele ist**, die für diesen Datensatz 47 beträgt. Beachten Sie, dass m größer ist und Ihre Kostenfunktion eine größere Zahl berechnet, wenn wir mehr Trainingsbeispiele haben. Dies ist eine Zusammenfassung mehrerer Beispiele. Um eine Kostenfunktion zu erstellen, die nicht automatisch größer wird, wenn die Größe des Trainingssatzes vereinbarungsgemäß größer wird, berechnen wir den **durchschnittlichen quadratischen Fehler** anstelle des gesamten quadratischen Fehlers und tun dies, indem wir wie folgt durch m dividieren. Wir sind fast da. Nur noch eine letzte Sache. Konventionell teilt sich die Kostenfunktion, die maschinelles Lernen verwendet, tatsächlich durch das **Zweifache von m**.

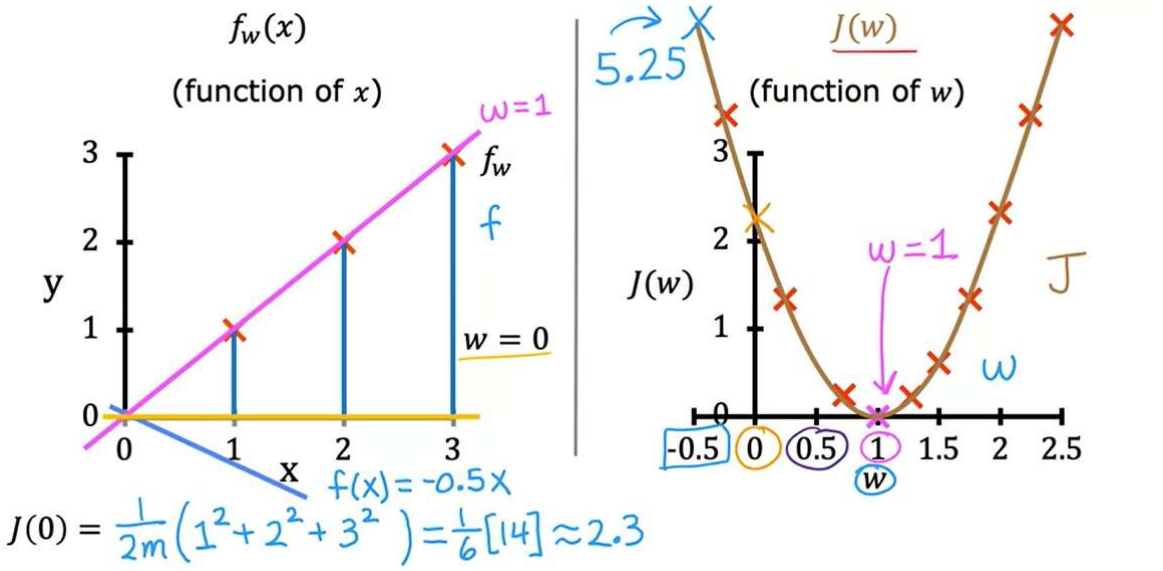
Die zusätzliche Division durch 2 soll nur dazu dienen, dass einige unserer späteren Berechnungen übersichtlicher aussehen, aber die Kostenfunktion funktioniert immer noch, unabhängig davon, ob Sie diese Division durch 2 einbeziehen oder nicht. Dieser Ausdruck hier ist die Kostenfunktion und wir werden J(wb) schreiben, um auf die Kostenfunktion zu verweisen. Dies wird auch als **quadrierte Fehlerkostenfunktion** bezeichnet, und zwar deshalb, weil Sie das Quadrat dieser Fehlerterme nehmen. Beim maschinellen Lernen verwenden verschiedene Personen unterschiedliche Kostenfunktionen für unterschiedliche Anwendungen, aber die Kostenfunktion mit quadratischem Fehler wird bei weitem am häufigsten für die lineare Regression und im Übrigen für alle Regressionsprobleme verwendet, bei denen sie für viele Anwendungen gute Ergebnisse zu liefern scheint . Zur Erinnerung: Die Vorhersage y ist gleich den Ausgaben des Modells f bei x.

Wir können die Kostenfunktion J(wb) als 1 über 2m mal die Summe von i gleich 1 bis m von f von x^i minus y^i der quadrierten Menge umschreiben. Schließlich wollen wir Werte für w und b finden, die die Kostenfunktion klein machen. Aber bevor wir dorthin gehen, wollen wir uns zunächst ein besseres Bild davon machen, was J(wb) wirklich berechnet. An diesem Punkt denken Sie vielleicht, dass wir eine Menge Mathematik betrieben haben, um die Kostenfunktion zu definieren. Aber was genau macht es? Fahren wir mit dem nächsten Video fort, in dem wir Schritt für Schritt durch ein Beispiel dafür gehen, was die Kostenfunktion wirklich berechnet. Ich hoffe, dass es Ihnen dabei hilft, eine Vorstellung davon zu entwickeln, was es bedeutet, wenn J(wb) groß ist und wenn die Kosten j klein sind. Kommen wir zum nächsten Video.

Kostenfunktion Verständnis

Wir sehen die mathematische Definition der Kostenfunktion. Lassen Sie uns nun eine Vorstellung davon entwickeln, was die Kostenfunktion wirklich tut. In diesem Video gehen wir ein Beispiel durch, um zu sehen, wie die Kostenfunktion verwendet werden kann, um die besten Parameter für Ihr Modell zu finden.

Die Kostenfunktion J misst die **Differenz zwischen den Vorhersagen des Modells und den tatsächlichen wahren Werten für y**. Was Sie später sehen, ist, dass die lineare Regression versuchen würde, Werte für w und b zu finden und dann a J von w so klein wie möglich zu machen. In der Mathematik schreiben wir es so. Wir wollen J als Funktion von w und b minimieren. Damit wir die Kostenfunktion J besser visualisieren können, verwenden wir nun eine vereinfachte Version des linearen Regressionsmodells. Wir werden das Modell fw von x verwenden, das w mal x ist. => Kostenfunktion des Parameters w, der„Steigung innerhalb der Messwerte“



Man kann sich das so vorstellen, als würde man das ursprüngliche Modell auf der linken Seite nehmen und den Parameter b entfernen oder den Parameter b gleich 0 setzen. Es verlässt einfach die Gleichung, sodass f jetzt nur noch w mal x ist. Sie haben jetzt nur noch einen Parameter w und Ihre Kostenfunktion J sieht ähnlich aus wie vorher. Nehmen Sie die

Differenz und quadrieren Sie sie, außer jetzt ist f gleich w mal xi, und J ist jetzt eine Funktion von nur w. Auch das Ziel wird etwas anders, da Sie nur einen Parameter haben, w, nicht w und b. Ziel dieses vereinfachten Modells ist es, den Wert für w zu finden, der J von w minimiert. Um dies visuell zu sehen, bedeutet dies, dass, wenn b auf 0 gesetzt ist, f eine Linie definiert, die so aussieht. Sie sehen, dass die Linie hier durch den Ursprung verläuft, denn wenn x 0 ist, ist auch f(x) 0. Lassen Sie uns nun anhand dieses vereinfachten Modells sehen, wie sich die Kostenfunktion ändert, wenn Sie unterschiedliche Werte für den Parameter w wählen.

Schauen wir uns insbesondere die Diagramme des Modells f(x) und der Kostenfunktion J an. Ich werde diese nebeneinander darstellen, und Sie werden sehen, wie die beiden zusammenhängen. Beachten Sie zunächst, dass fw für f subscript w, wenn der Parameter w fest ist, also immer ein konstanter Wert ist, fw nur eine Funktion von x ist, was bedeutet, dass der geschätzte Wert von y vom Wert der Eingabe x abhängt . Im Gegensatz dazu ist die Kostenfunktion J, wenn man nach rechts blickt, eine Funktion von w, wobei w die Steigung der durch f w definierten Linie steuert. Die durch J definierten Kosten hängen von einem Parameter ab, in diesem Fall dem Parameter w. Lassen Sie uns fortfahren und diese Funktionen fw von x und J von w nebeneinander darstellen, damit Sie sehen können, wie sie zusammenhängen. Wir beginnen mit dem Modell, das ist die Funktion fw von x auf der linken Seite. Hier befindet sich das Eingabemerkmal x auf der horizontalen Achse und der Ausgabewert y auf der vertikalen Achse.

Hier sind die Diagramme von drei Punkten, die den Trainingssatz an den Positionen 1, 1, 2, 2 und 3,3 darstellen. Wählen wir einen Wert für w. Angenommen, w ist 1. Für diese Wahl von w, der Funktion fw , sagt man diese gerade Linie mit einer Steigung von 1. Als Nächstes können Sie die Kosten J berechnen, wenn w gleich 1 ist. Sie erinnern sich vielleicht daran, dass die Die Kostenfunktion ist wie folgt definiert und ist die quadratische Fehlerkostenfunktion. Wenn Sie fw (X^i) durch w mal X^i ersetzen , sieht die Kostenfunktion folgendermaßen aus. Wobei dieser Ausdruck jetzt w mal X^i minus Y^i ist . Für diesen Wert von w stellt sich heraus, dass der Fehlerterm innerhalb der Kostenfunktion, also w mal X^i minus Y^i , für jeden der drei Datenpunkte gleich 0 ist. Denn für diesen Datensatz gilt: Wenn x 1 ist, ist y 1. Wenn w ebenfalls 1 ist, ist f(x) gleich 1, also ist f(x) für dieses erste Trainingsbeispiel gleich y und die Differenz ist 0. Setzt man dies in die Kostenfunktion J ein, erhält man 0 im Quadrat. Wenn x gleich 2 ist, dann ist y gleich 2 und f(x) ist ebenfalls 2. Auch hier ist f(x) gleich y, für das zweite Trainingsbeispiel. In der Kostenfunktion beträgt der quadratische Fehler für das zweite Beispiel ebenfalls 0 im Quadrat. Wenn schließlich x 3 ist, dann ist y 3 und f(3) ist ebenfalls 3. In einer Kostenfunktion ist der dritte quadrierte Fehlerterm ebenfalls 0 im Quadrat. Für alle drei Beispiele in diesem Trainingssatz ist f (_ , dann sind die Kosten J gleich 0. Jetzt können Sie rechts die Kostenfunktion J zeichnen.

mit w und nicht mit x bezeichnet wird und die vertikale Achse jetzt mit J und nicht mit y bezeichnet ist, da die Kostenfunktion eine Funktion des Parameters w ist. Sie haben J(1) gleich 0. Mit anderen Worten, wenn w gleich 1 ist, ist J(w) 0, also lassen Sie mich fortfahren und das grafisch darstellen. Schauen wir uns nun an, wie sich F und J für verschiedene Werte von w ändern. W kann einen Wertebereich annehmen, also kann w negative Werte

annehmen, w kann 0 sein und es kann auch positive Werte annehmen. Was wäre, wenn w gleich 0,5 statt 1 wäre, wie würden diese Diagramme dann aussehen? Machen wir weiter und planen das. Setzen wir w gleich 0,5, und in diesem Fall sieht die Funktion f(x) nun so aus, dass sie eine Gerade mit einer Steigung gleich 0,5 ist. Berechnen wir auch die Kosten J, wenn w 0,5 ist. Denken Sie daran, dass die Kostenfunktion den quadratischen Fehler oder die Differenz zwischen dem Schätzwert, also ŷ I, also F(X^i), und dem wahren Wert, also Y^i , für jedes Beispiel i misst . Visuell können Sie sehen, dass der Fehler oder die Differenz hier gleich der Höhe dieser vertikalen Linie ist, wenn x gleich 1 ist. Denn diese untere Linie ist die Lücke zwischen dem tatsächlichen Wert von y und dem Wert, den die Funktion f vorhergesagt hat etwas weiter unten hier. Für dieses erste Beispiel ist f(x) 0,5, wenn x 1 ist.

Der quadratische Fehler im ersten Beispiel beträgt 0,5 minus 1 zum Quadrat. Denken Sie an die Kostenfunktion. Wir summieren alle Trainingsbeispiele im Trainingssatz. Kommen wir zum zweiten Trainingsbeispiel. Wenn x 2 ist, sagt das Modell voraus, dass f(x) 1 ist und der tatsächliche Wert von y 2 ist. Der Fehler für das zweite Beispiel ist gleich der Höhe dieses kleinen Liniensegments hier, und der quadrierte Fehler ist das Quadrat von Bestimmen Sie die Länge dieses Liniensegments, sodass Sie 1 minus 2 zum Quadrat erhalten. Machen wir das dritte Beispiel. Bei Wiederholung dieses Vorgangs beträgt der Fehler, der auch durch dieses Liniensegment dargestellt wird, 1,5 minus 3 zum Quadrat. Als nächstes summieren wir alle diese Terme, was 3,5 ergibt. Dann multiplizieren wir diesen Term mit 1 über 2m, wobei m die Anzahl der Trainingsbeispiele ist. Da es drei Trainingsbeispiele gibt, ist m gleich 3, also ist dies gleich 1 über 2 mal 3, wobei dieses m hier 3 ist.

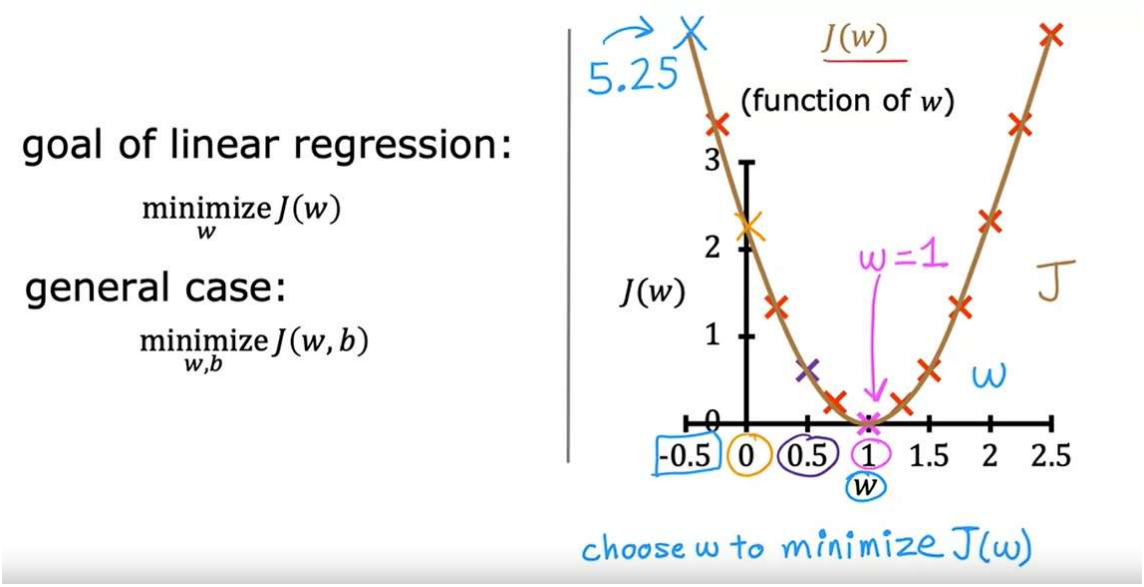
Wenn wir die Rechnung durchrechnen, ergibt sich ein Wert von 3,5 dividiert durch 6. Die Kosten J betragen etwa 0,58. Lassen Sie uns fortfahren und das dort rechts einzeichnen. Versuchen wir nun einen weiteren Wert für w. Wie wäre es, wenn w gleich 0 wäre? Wie sehen die Graphen für f und J aus, wenn w gleich 0 ist? Es stellt sich heraus, dass, wenn w gleich 0 ist, f(x) genau diese horizontale Linie ist, die genau auf der x-Achse liegt. Der Fehler für jedes Beispiel ist eine Linie, die von jedem Punkt bis zur horizontalen Linie reicht, die f(x) gleich 0 darstellt. Die Kosten J, wenn w gleich 0 ist, betragen 1 über 2m mal der Menge, 1^2 plus 2^2 plus 3 ^2, und das entspricht 1 über 6 mal 14, also etwa 2,33. Zeichnen wir hier diesen Punkt ein, an dem w 0 ist und J von 0 2,33 beträgt. Sie können dies auch für andere Werte von w tun. Da w eine beliebige Zahl sein kann, kann es auch ein negativer Wert sein. Wenn w negativ 0,5 ist, dann ist die Linie f eine absteigende Linie wie diese. Es stellt sich heraus, dass, wenn w minus 0,5 ist, die Kosten sogar noch höher sind, etwa 5,25, was hier oben der Fall ist. Sie können die Kostenfunktion für verschiedene Werte von w usw. weiter berechnen und diese grafisch darstellen. Es stellt sich heraus, dass Sie durch die Berechnung einer Reihe von Werten langsam herausfinden können, wie die Kostenfunktion J aussieht, und genau das ist J.

Um es noch einmal zusammenzufassen: Jeder Wert des Parameters w entspricht einer unterschiedlichen geraden Linienanpassung f(x) im Diagramm links. Für den gegebenen Trainingssatz entspricht diese Auswahl für einen Wert von w einem einzelnen Punkt im Diagramm rechts, da Sie für jeden Wert von w die Kosten J von w berechnen können. Wenn w beispielsweise gleich 1 ist, entspricht dies dieser geraden Linienanpassung durch die Daten und entspricht auch diesem Punkt im Diagramm von J, wobei w gleich 1 und die Kosten J von

1 gleich 0 sind. Wenn w hingegen gleich 0,5 ist, Dadurch erhält man diese Linie, die eine geringere Steigung hat. Diese Linie entspricht in Kombination mit dem Trainingssatz diesem Punkt im Kostenfunktionsdiagramm bei w gleich 0,5. Für jeden Wert von w erhalten Sie eine andere Linie und die entsprechenden Kosten, J von w, und Sie können diese Punkte verwenden, um dieses Diagramm auf der rechten Seite zu zeichnen. Wie können Sie vor diesem Hintergrund den Wert von w auswählen, der die Funktion f ergibt und gut zu den Daten passt? Nun, wie Sie sich vorstellen können, scheint es eine gute Wahl zu sein, einen Wert für w zu wählen, der dazu führt, dass J von w so klein wie möglich ist.

J

ist die Kostenfunktion, die misst, wie groß die quadratischen Fehler sind. Wenn wir also w wählen, das diese quadratischen Fehler minimiert und sie so klein wie möglich macht, erhalten wir ein gutes Modell. Wenn Sie in diesem Beispiel den Wert von w wählen würden, der den kleinstmöglichen Wert von J von w ergibt, würden Sie am Ende w = 1 wählen. Wie Sie sehen, ist das eigentlich eine ziemlich gute Wahl. Dadurch entsteht die Linie, die sehr gut zu den Trainingsdaten passt.

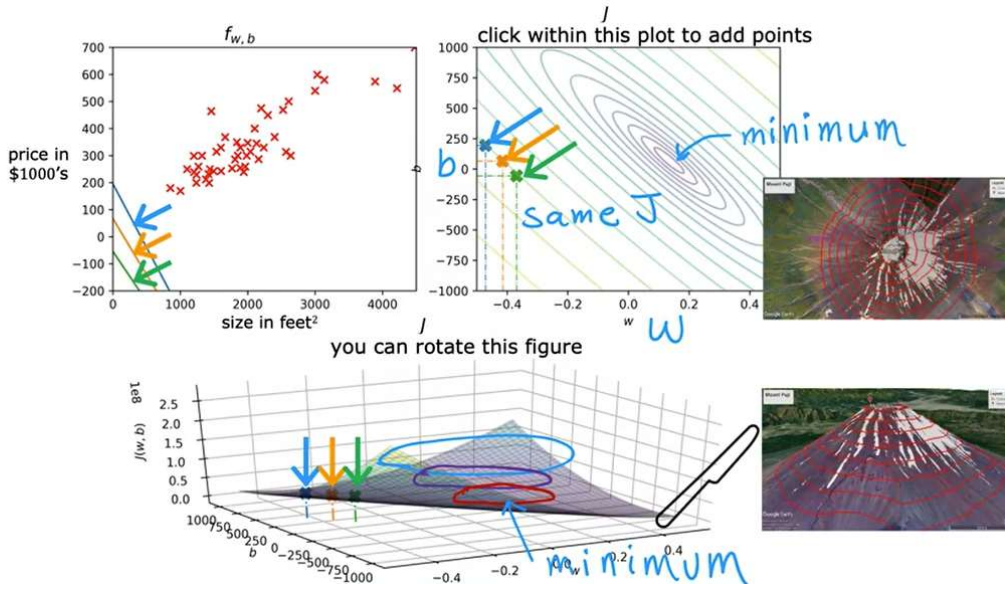


Auf diese Weise verwenden Sie bei der linearen Regression die Kostenfunktion, um den Wert von w zu finden, der J minimiert. Im allgemeineren Fall, in dem wir die Parameter w und b statt nur w hatten, finden Sie die Werte von w und b, die J minimieren.

Zusammenfassend haben Sie Diagramme von f und J gesehen und herausgefunden, wie die beiden zusammenhängen. Wenn Sie w oder w und b variieren, erhalten Sie unterschiedliche gerade Linien, und wenn diese gerade Linie über die Daten verläuft, ist die Ursache J klein. **Das Ziel der linearen Regression besteht darin, die Parameter w oder w und b zu finden, die den kleinstmöglichen Wert für die Kostenfunktion J ergeben**. In diesem Video haben wir nun unser Beispiel mit einem vereinfachten Problem durchgearbeitet, bei dem nur w verwendet wurde. Lassen Sie uns im nächsten Video visualisieren, wie die Kostenfunktion für die Vollversion der linearen Regression unter Verwendung von w und b aussieht. Sie sehen einige coole 3D-Plots. Kommen wir zum nächsten Video.

Visualisierung der Kostenfunktion

Im letzten Video haben Sie eine Visualisierung der Kostenfunktion J von w oder J von w, b gesehen. Schauen wir uns einige weitere umfassendere Visualisierungen an, damit Sie eine noch bessere Vorstellung davon bekommen, was die Kostenfunktion tut. Hier ist, was wir bisher gesehen haben. Es gibt das Modell, die Modellparameter w und b, die Kostenfunktion J von w und b sowie das Ziel der linearen Regression, das darin besteht, die Kostenfunktion J von w und b über die Parameter w und b zu minimieren. Im letzten Video hatten wir b vorübergehend auf Null gesetzt, um die Visualisierungen zu vereinfachen. Kehren wir nun zum ursprünglichen Modell mit den beiden Parametern w und b zurück, ohne b gleich 0 zu setzen. Wie beim letzten Mal möchten wir ein visuelles Verständnis der Modellfunktion f(x) erhalten, die hier links gezeigt wird , und wie es mit der Kostenfunktion J von w, b zusammenhängt, hier rechts gezeigt. Hier ist ein Übungssatz zu Hausgrößen und -preisen. Nehmen wir an, Sie wählen eine mögliche Funktion von x aus, wie diese. Hier habe ich w auf 0,06 und b auf 50 gesetzt. f(x) ist 0,06 mal x plus 50.



Beachten Sie, dass dies kein besonders gutes Modell für diesen Trainingssatz ist, sondern eigentlich ein ziemlich schlechtes Modell. Es scheint, dass die Immobilienpreise ständig unterschätzt werden. Angesichts dieser Werte für w und b schauen wir uns an, wie die Kostenfunktion J von w und b aussehen könnte. Erinnern Sie sich daran, was wir das letzte Mal gesehen haben, als Sie nur w hatten, weil wir b vorübergehend auf Null gesetzt haben, um die Dinge zu vereinfachen, aber dann hatten wir eine Darstellung der Kostenfunktion erstellt, die nur als Funktion von w so aussah. **Als wir nur einen Parameter w hatten, hatte die Kostenfunktion diese U-förmige Kurve**, die ein bisschen wie eine Suppenschüssel geformt war. Das hört sich lecker an. In diesem Immobilienpreisbeispiel, das wir auf dieser Folie haben, haben wir nun **zwei Parameter, w und b**. Die Handlung wird etwas komplexer.

Es stellt sich heraus, dass auch die Kostenfunktion eine ähnliche Form wie eine Suppenschüssel hat, außer dass sie **dreidimensional statt zweidimensioniert ist**. Abhängig von Ihrem Trainingssatz sieht die Kostenfunktion tatsächlich in etwa so aus. Für mich sieht

das wie eine Suppenschüssel aus, vielleicht weil ich ein bisschen hungrig bin, oder vielleicht sieht es für dich wie ein gebogener Essteller oder eine Hängematte aus. Eigentlich klingt das auch entspannend, und da ist Ihr Kokosnussgetränk. Wenn Sie mit diesem Kurs fertig sind, sollten Sie sich vielleicht einen Urlaub gönnen und in einer Hängematte wie dieser entspannen. Was Sie hier sehen, ist ein 3D-Oberflächendiagramm, bei dem die Achsen mit w und b beschriftet sind Wenn Sie w und b, die beiden Parameter des Modells, variieren, erhalten Sie unterschiedliche Werte für die Kostenfunktion J von w und b.

Dies ist der U-förmigen Kurve, die Sie im letzten Video gesehen haben, sehr ähnlich, außer dass Sie statt eines Parameters w als Eingabe für j jetzt zwei Parameter, w und b, als Eingaben in diese Suppenschüssel oder diese Hängemattenform haben Funktion J. Ich möchte nur darauf hinweisen, dass jeder einzelne Punkt auf dieser Oberfläche eine bestimmte Auswahl von w und b darstellt. Wenn zum Beispiel w minus 10 und b minus 15 wäre, dann ist die Höhe der Oberfläche über diesem Punkt der Wert von j, wenn w minus 10 und b minus 15 ist. Nun, um das Spezifische noch genauer zu betrachten Punkte, es gibt eine andere Möglichkeit, die Kostenfunktion J darzustellen, die für die Visualisierung nützlich wäre. Anstatt diese 3D-Oberflächendiagramme zu verwenden, verwende ich lieber genau dieselbe Funktion J.

Ich ändere die Funktion J überhaupt nicht und zeichne sie mithilfe eines sogenannten Konturdiagramms. Wenn Sie jemals eine topografische Karte gesehen haben, die zeigt, wie hoch verschiedene Berge sind, sind die Konturen in einer topografischen Karte im Wesentlichen horizontale Schnitte der Landschaft beispielsweise eines Berges. Dieses Bild zeigt den Berg Fuji in Japan. Ich erinnere mich noch daran, wie meine Familie als Teenager den Berg Fuji besuchte. Es sind wunderschöne Sehenswürdigkeiten. Wenn Sie direkt über dem Berg fliegen, sieht diese Höhenlinienkarte so aus. Es zeigt alle Punkte, sie liegen für verschiedene Höhen auf der gleichen Höhe. Unten auf dieser Folie befindet sich ein 3D-Oberflächendiagramm der Kostenfunktion J. Ich weiß, es sieht nicht sehr schüsselförmig aus, aber es ist tatsächlich eine nur sehr ausgedehnte Schüssel, weshalb es so aussieht. In einem optionalen Labor, das in Kürze folgt, können Sie dies in 3D sehen und sich selbst um die Oberfläche drehen, sodass es dort deutlicher wie eine Schüssel aussieht. Als nächstes sehen Sie hier oben rechts ein Konturdiagramm genau derselben Kostenfunktion wie unten gezeigt.

Die beiden Achsen in diesem Konturdiagramm sind b auf der vertikalen Achse und w auf der horizontalen Achse. Was jedes dieser Ovale, auch Ellipsen genannt, zeigt, sind die Mittelpunkte auf der 3D-Oberfläche, die sich auf exakt derselben Höhe befinden. Mit anderen Worten, die Menge der Punkte, die den gleichen Wert für die Kostenfunktion J haben. Um die Konturdiagramme zu erhalten, nehmen Sie die 3D-Oberfläche unten und schneiden sie mit einem Messer horizontal in Scheiben. Sie nehmen horizontale Schnitte dieser 3D-Oberfläche und erhalten alle Punkte, sie liegen auf derselben Höhe. Daher wird jeder horizontale Schnitt letztendlich als eine dieser Ellipsen oder eines dieser Ovale dargestellt. Konkret: Wenn Sie diesen Punkt und diesen Punkt und diesen Punkt nehmen, haben alle diese drei Punkte denselben Wert für die Kostenfunktion J, obwohl sie unterschiedliche Werte für w und b haben. In der Abbildung oben links sehen Sie auch, dass diese drei Punkte unterschiedlichen Funktionen f entsprechen, die in diesem Fall alle drei eigentlich ziemlich schlecht für die Vorhersage von Immobilienpreisen sind. Nun, der Boden

der Schüssel, wo die Kostenfunktion J minimal ist, ist dieser Punkt genau hier, in der Mitte dieser konzentrischen Ovale.

Wenn Sie Konturdiagramme noch nicht oft gesehen haben, möchte ich Sie bitten, sich vorzustellen, dass Sie in einem Flugzeug oder in einer Rakete hoch über der Schüssel fliegen und direkt darauf hinunterblicken. Das ist so, als ob Sie Ihren Computermonitor flach mit der Vorderseite nach oben auf Ihren Schreibtisch stellen und die Schüsselform direkt aus Ihrem Bildschirm herausragt und sich über Ihren Schreibtisch erhebt. Stellen Sie sich vor, dass die Schüsselform aus Ihrem flach liegenden Computerbildschirm herauswächst, sodass jedes dieser Ovale die gleiche Höhe über Ihrem Bildschirm hat und das Minimum der Schüssel genau dort unten in der Mitte des kleinsten Ovals liegt. Es stellt sich heraus, dass die Konturdiagramme eine praktische Möglichkeit sind, die 3D-Kostenfunktion J zu visualisieren, aber in gewisser Weise erfolgt die Darstellung nur in 2D. In diesem Video haben Sie gesehen, wie das 3D-Schalen-Oberflächendiagramm auch als Konturdiagramm visualisiert werden kann. Lassen Sie uns im nächsten Video auch anhand dieser Visualisierung einige spezifische Auswahlmöglichkeiten von w und b im linearen Regressionsmodell visualisieren, damit Sie sehen können, wie sich diese unterschiedlichen Auswahlmöglichkeiten auf die gerade Linie auswirken, die Sie an die Daten anpassen. Kommen wir zum nächsten Video.

Visualisierungsbeispiele

Schauen wir uns einige weitere Visualisierungen von w und b an. Hier ist ein Beispiel. Hier haben Sie einen bestimmten Punkt im Diagramm j. Für diesen Punkt entspricht w etwa minus 0,15 und b etwa 800. Dieser Punkt entspricht einem Wertepaar für w und b, das einen bestimmten Kostenwert j verwendet. Tatsächlich entspricht dieses Wertepaar für w und b dieser Funktion f von x, also dieser Linie, die Sie links sehen können. Diese Linie schneidet die vertikale Achse bei 800, weil b gleich 800 ist und die Steigung der Linie negativ 0,15 ist, weil w gleich negativ 0,15 ist.

Wenn Sie sich nun die Datenpunkte im Trainingssatz ansehen, stellen Sie möglicherweise fest, dass diese Linie nicht gut zu den Daten passt. Für diese Funktion f(x) mit diesen Werten von w und b sind viele der Vorhersagen für den Wert von y ziemlich weit vom tatsächlichen Zielwert von y entfernt, der in den Trainingsdaten enthalten ist. Da diese Linie nicht gut passt, liegen die Kosten dieser Linie hier draußen, was ziemlich weit vom Minimum entfernt ist, wenn man sich den Graphen von j ansieht. Die Kosten sind ziemlich hoch, da diese Wahl von W und B einfach nicht so gut zum Trainingssatz passt. Schauen wir uns nun ein weiteres Beispiel mit einer anderen Wahl von w und b an. Hier ist eine weitere Funktion, die immer noch nicht gut zu den Daten passt, aber vielleicht etwas weniger schlecht. Dieser Punkt stellt hier die Kosten für dieses Broschürenpaar aus W und B dar, das diese Linie erstellt. Der Wert von w ist gleich 0 und der Wert von b beträgt etwa 360. Dieses Parameterpaar entspricht dieser Funktion, die eine flache Linie ist, da f(x) gleich 0 mal x plus 360 ist.

Ich hoffe das ergibt Sinn. Schauen wir uns noch ein weiteres Beispiel an. Hier ist eine weitere Auswahl für w und b, und mit diesen Werten erhalten Sie diese Linie f von x. Auch hier ist die

Übereinstimmung nicht besonders gut mit den Daten, im Vergleich zum vorherigen Beispiel tatsächlich weiter vom Minimum entfernt. Denken Sie daran, dass das Minimum in der Mitte dieser kleinsten Ellipse liegt. Letztes Beispiel: Wenn Sie sich f(x) auf der linken Seite ansehen, scheint dies ziemlich gut zum Trainingssatz zu passen. Sie können rechts sehen, dass dieser Punkt, der die Kosten darstellt, sehr nahe an der Mitte der kleineren Ellipse liegt. Er ist nicht ganz genau das Minimum, aber ziemlich nahe dran. Für diesen Wert von w und b erhalten Sie diese Linie f von x. Sie können sehen, dass Sie den Fehler für jeden Datenpunkt erhalten, wenn Sie die vertikalen Abstände zwischen den Datenpunkten und den vorhergesagten Werten auf der geraden Linie messen.

Die Summe der Fehlerquadrate für alle diese Datenpunkte liegt ziemlich nahe an der minimal möglichen Summe der Fehlerquadrate aller möglichen Geradenanpassungen. Ich hoffe, dass Sie durch die Betrachtung dieser Zahlen ein besseres Gefühl dafür bekommen, wie sich unterschiedliche Wahlen der Parameter auf die Linie f(x) auswirken und wie dies unterschiedlichen Werten für die Kosten j entspricht, und hoffentlich können Sie sehen, wie die Linien besser passen entsprechen Punkten im Diagramm von j, die näher an den minimal möglichen Kosten für diese Kostenfunktion j von w und b liegen. In der optionalen Übung, die diesem Video folgt, können Sie einige Codes ausführen und sich dabei merken, dass der gesamte Code angegeben ist. Sie müssen also nur die Umschalt-Eingabetaste drücken, um ihn auszuführen und einen Blick darauf zu werfen, und die Übung zeigt Ihnen, wie das funktioniert Die Kostenfunktion ist im Code implementiert.

Bei einem kleinen Trainingssatz und unterschiedlichen Auswahlmöglichkeiten für die Parameter können Sie sehen, wie die Kosten variieren, je nachdem, wie gut das Modell mit den Daten übereinstimmt. Im optionalen Labor können Sie auch mit der interaktiven Konsolenhandlung spielen. Schauen Sie sich das an. Sie können mit dem Mauszeiger irgendwo auf das Konturdiagramm klicken und sehen die gerade Linie, die durch die Werte definiert wird, die Sie für die Parameter w und b ausgewählt haben. Hier oben auf dem 3D-Oberflächendiagramm sehen Sie auch einen Punkt, der die Kosten anzeigt.

Schließlich verfügt das optionale Labor auch über ein 3D-Oberflächendiagramm, das Sie manuell mit dem Mauszeiger drehen und drehen können, um einen besseren Einblick in die Kostenfunktion zu erhalten. Ich wünsche Ihnen viel Spaß beim Spielen mit dem optionalen Labor. Jetzt in linearer Regression, anstatt manuell versuchen zu müssen, ein Konturdiagramm für den besten Wert für w und b zu lesen, was kein wirklich gutes Verfahren ist und auch nicht funktionieren wird, wenn wir zu komplexeren Modellen für maschinelles Lernen kommen.

Was Sie wirklich wollen, ist ein effizienter Algorithmus, den Sie in Code schreiben können, um automatisch die Werte der Parameter w und b zu finden. Sie liefern Ihnen die beste Anpassungslinie. Das minimiert die Kostenfunktion j. Dafür gibt es einen Algorithmus namens **Gradientenabstieg.** Dieser Algorithmus ist einer der wichtigsten Algorithmen beim maschinellen Lernen. Der Gradientenabstieg und Variationen des Gradientenabstiegs werden zum Trainieren nicht nur der linearen Regression, sondern einiger der größten und komplexesten Modelle in der gesamten KI verwendet. Gehen wir zum nächsten Video, um in diesen wirklich wichtigen Algorithmus namens Gradientenabstieg einzutauchen.

Python Code Kostenfunktion

Computing Cost

The term 'cost' in this assignment might be a little confusing since the data is housing cost. Here, cost is a measure how well our model is predicting the target price of the house. The term 'price' is used for housing data.

The equation for cost with one variable is:

$$J(w,b) = \frac{1}{2m}\sum_{i=0}^{m-1}(f_{w,b}(x^{(i)}) - y^{(i)})^2 \tag{1}$$

where

$$f_{w,b}(x^{(i)}) = wx^{(i)} + b \tag{2}$$

- $f_{w,b}(x^{(i)})$ is our prediction for example i using parameters w, b.
- $(f_{w,b}(x^{(i)}) - y^{(i)})^2$ is the squared difference between the target value and the prediction.
- These differences are summed over all the m examples and divided by `2m` to produce the cost, $J(w,b)$.

> Note, in lecture summation ranges are typically from 1 to m, while code will be from 0 to m-1.

The code below calculates cost by looping over each example. In each loop:

- `f_wb` , a prediction is calculated
- the difference between the target and the prediction is calculated and squared.
- this is added to the total cost.

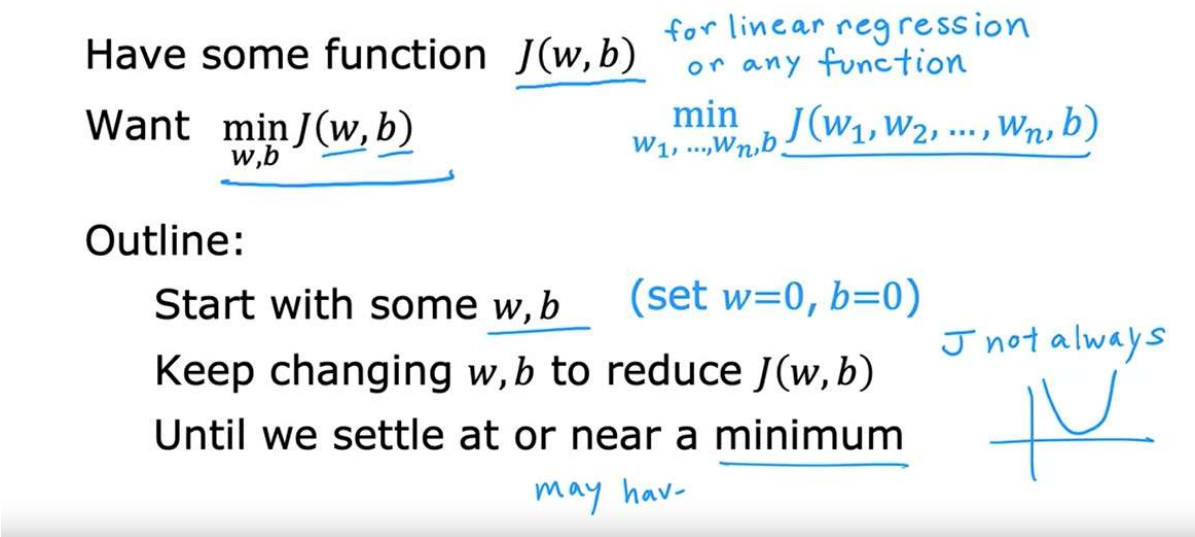
```
def compute_cost(x, y, w, b):
    """
    Computes the cost function for linear regression.

    Args:
      x (ndarray (m,)): Data, m examples
      y (ndarray (m,)): target values
      w,b (scalar)    : model parameters

    Returns
        total_cost (float): The cost of using w,b as the parameters for linear regression
               to fit the data points in x and y
    """
    # number of training examples
    m = x.shape[0]

    cost_sum = 0
    for i in range(m):
        f_wb = w * x[i] + b

        cost = (f_wb - y[i]) ** 2
        cost_sum = cost_sum + cost
    total_cost = (1 / (2 * m)) * cost_sum

    return total_cost
```

Gefälleabstieg (Gradient Decent)

Willkommen zurück. Im letzten Video haben wir Visualisierungen der Kostenfunktion j gesehen und gezeigt, wie Sie verschiedene Optionen für die Parameter w und b ausprobieren und sehen können, welchen Kostenwert sie Ihnen bringen. Es wäre schön, wenn wir eine systematischere Möglichkeit hätten, die Werte von w und b zu ermitteln, was zu den kleinstmöglichen Kosten j von w, b führt. Es stellt sich heraus, dass es einen Algorithmus namens „Gradientenabstieg“ gibt, den Sie dafür verwenden können.

Der Gradientenabstieg wird beim maschinellen Lernen überall eingesetzt, nicht nur für die lineare Regression, sondern beispielsweise zum Trainieren einiger der fortschrittlichsten neuronalen Netzwerkmodelle, auch Deep-Learning-Modelle genannt. Deep-Learning-Modelle haben Sie im zweiten Kurs kennengelernt. Wenn Sie diese beiden Aspekte des Gradientenabstiegs erlernen, werden Sie mit einem der wichtigsten Bausteine des maschinellen Lernens vertraut gemacht.

Hier ist eine Übersicht darüber, was wir mit dem Gefälleabstieg machen werden. Sie haben hier die Kostenfunktion j von w, b, die Sie minimieren möchten. In dem Beispiel, das wir bisher gesehen haben, handelt es sich um eine Kostenfunktion für die lineare Regression, aber es stellt sich heraus, dass der Gradientenabstieg ein Algorithmus ist, mit dem Sie versuchen können, jede Funktion zu minimieren, nicht nur eine Kostenfunktion für die lineare Regression. Um diese Diskussion über den Gradientenabstieg allgemeiner zu gestalten, stellt sich heraus, dass der Gradientenabstieg für allgemeinere Funktionen gilt, einschließlich anderer Kostenfunktionen, die mit Modellen arbeiten, die mehr als zwei Parameter haben. Wenn Sie beispielsweise eine Kostenfunktion J als Funktion von w_1, w_2 bis w_n und b haben, besteht Ihr Ziel darin, j über die Parameter w_1 bis w_n und b zu minimieren. Mit anderen Worten: Sie möchten Werte für w_1 bis w_n und b auswählen, die Ihnen den kleinstmöglichen Wert von j ergeben. Es stellt sich heraus, dass der Gradientenabstieg ein Algorithmus ist, den Sie anwenden können, um auch diese Kostenfunktion j zu minimieren.



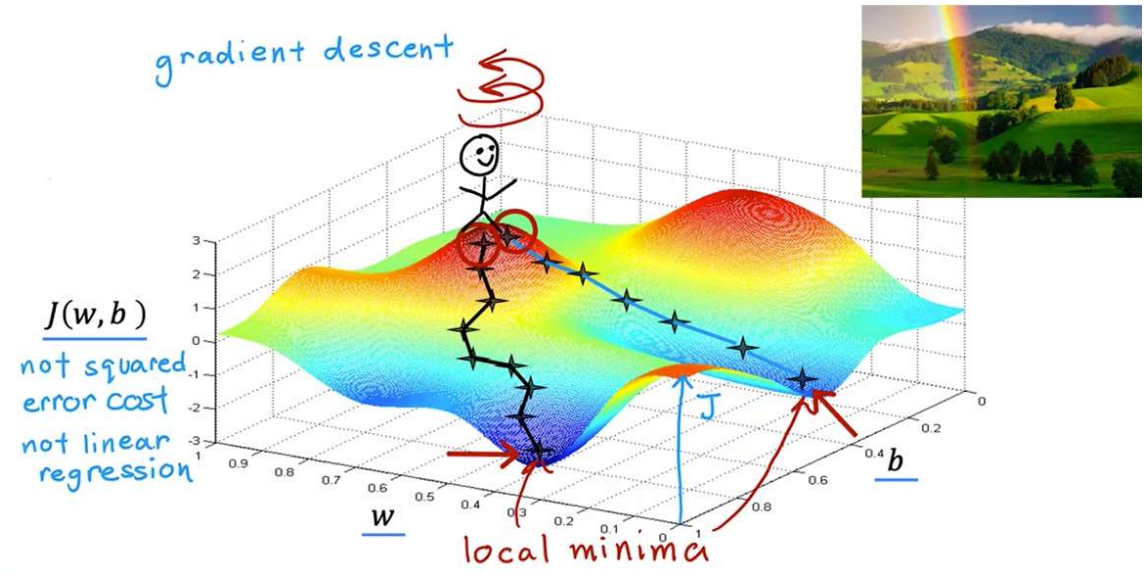
Was Sie tun werden, ist, zunächst mit einigen ersten Vermutungen für w und b zu beginnen. Bei der linearen Regression spielt es keine allzu große Rolle, wie hoch der Anfangswert ist, daher **besteht eine häufige Wahl darin, beide auf 0 zu setzen**. Beispielsweise können Sie als anfängliche Schätzung w auf 0 und b auf 0 setzen.

Was Sie mit dem Gradientenabstiegsalgorithmus tun werden, ist, dass Sie die Parameter w und b jedes Mal ein bisschen ändern, um zu versuchen, die Kosten j von w, b zu reduzieren, bis sich j hoffentlich auf oder nahe einem Minimum einpendelt. Eine Sache, die ich beachten

sollte, ist, dass es für einige Funktionen j, die möglicherweise keine Bogen- oder Hängemattenform haben, möglich ist, **dass es mehr als ein mögliches Minimum gibt**. Schauen wir uns ein Beispiel eines komplexeren Oberflächendiagramms j an, um zu sehen, was der Gradient bewirkt.

Diese Funktion ist **keine quadratische Fehlerkostenfunktion**. Bei der linearen Regression mit der quadrierten Fehlerkostenfunktion erhält man immer eine Bogenform oder eine Hängemattenform. Dies ist jedoch eine Art Kostenfunktion, die Sie möglicherweise erhalten, wenn Sie ein neuronales Netzwerkmodell trainieren. Beachten Sie die Achsen, also w und b auf der unteren Achse. Für unterschiedliche Werte von w und b erhalten Sie unterschiedliche Punkte auf dieser Oberfläche, j von w, b, wobei die Höhe der Oberfläche an einem bestimmten Punkt der Wert der Kostenfunktion ist.

Stellen wir uns nun vor, dass es sich bei diesem Oberflächengrundstück tatsächlich um einen Blick auf einen leicht hügeligen Outdoor-Park oder einen Golfplatz handelt, bei dem die höchsten Punkte Hügel und die niedrigen Punkte Täler sind. Ich möchte, dass Sie sich vorstellen, dass Sie physisch an dieser Stelle auf dem Hügel stehen. Wenn es Ihnen hilft, sich zu entspannen, stellen Sie sich vor, dass es einen wirklich schönen Hügel mit viel schönem grünen Gras, Schmetterlingen und Blumen gibt. Ihr Ziel ist es, hier oben zu beginnen und so effizient wie möglich auf den Grund eines dieser Täler zu gelangen.



Was der **Gradientenabstiegsalgorithmus** bewirkt, ist, dass Sie sich um 360 Grad drehen, sich umschauen und sich fragen, ob ich einen winzigen kleinen Schritt in eine Richtung machen würde und so schnell wie möglich bergab gehen möchte, bis oder eines dieser Täler. Welche Richtung wähle ich, um diesen kleinen Schritt zu machen? Nun, wenn Sie diesen Hügel so effizient wie möglich hinuntergehen möchten, stellt sich heraus, dass Sie, wenn Sie an dieser Stelle des Hügels stehen und sich umschauen, feststellen werden, dass die beste Richtung für Ihren nächsten Schritt bergab ungefähr ist diese Richtung. Mathematisch gesehen ist dies die Richtung des steilsten Abstiegs.

Das bedeutet, dass Sie mit einem winzigen kleinen Schritt schneller bergab gelangen als mit einem winzigen kleinen Schritt in jede andere Richtung. Nachdem Sie diesen ersten Schritt getan haben, befinden Sie sich nun an diesem Punkt des Hügels hier. Nun wiederholen wir den Vorgang. Wenn Sie an diesem neuen Punkt stehen, drehen Sie sich erneut um 360 Grad und fragen sich: In welche Richtung werde ich den nächsten kleinen Schritt machen, um bergab zu gelangen? Wenn Sie das tun und einen weiteren Schritt machen, bewegen Sie sich am Ende ein Stück in diese Richtung und können weitermachen. Von diesem neuen Punkt aus können Sie sich erneut umschauen und entscheiden, in welche Richtung Sie am schnellsten bergab gelangen. Machen Sie einen weiteren Schritt, einen weiteren Schritt und so weiter, bis Sie sich am Ende dieses Tals befinden, an diesem lokalen Minimum, genau hier.

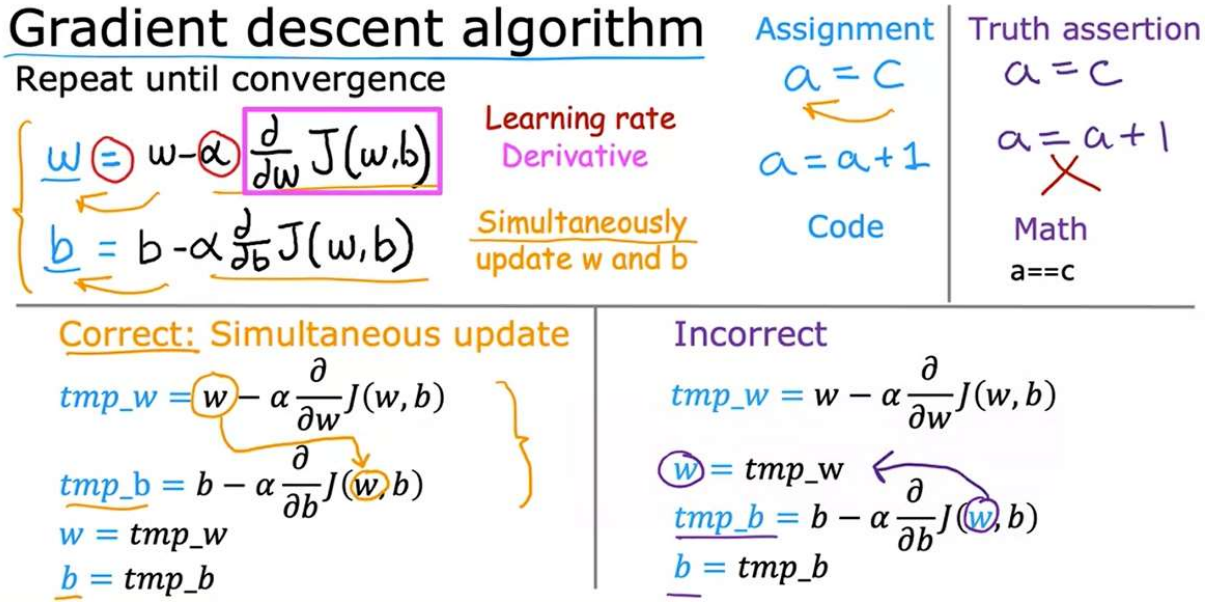
Sie haben gerade mehrere Schritte des Gradientenabstiegs durchlaufen. Es stellt sich heraus, dass der Gradientenabstieg eine interessante Eigenschaft hat. Denken Sie daran, dass Sie einen Startpunkt an der Oberfläche wählen können, indem Sie Startwerte für die Parameter w und b wählen. Als Sie vorhin einen Gefälleabstieg durchgeführt haben, hatten Sie an diesem Punkt hier begonnen. Stellen Sie sich nun vor, Sie würden den Gradientenabstieg noch einmal versuchen, aber dieses Mal wählen Sie einen anderen Startpunkt, indem Sie Parameter wählen, die Ihren Startpunkt hier nur ein paar Schritte weiter rechts platzieren.

Wenn Sie dann den Steigungsvorgang wiederholen, das heißt, Sie schauen sich um, machen Sie einen kleinen Schritt in Richtung des steilsten Anstiegs, sodass Sie hier landen. Dann schaust du dich noch einmal um, machst einen weiteren Schritt und so weiter. Wenn Sie dieses zweite Mal den Gefälleabstieg laufen und nur ein paar Schritte rechts von der Stelle beginnen, an der wir ihn das erste Mal gemacht haben, dann landen Sie in einem völlig anderen Tal. Dieses andere Minimum hier rechts.

Die Böden sowohl des ersten als auch des zweiten Tals werden lokale Minima genannt. Denn wenn Sie beginnen, das erste Tal hinunterzugehen, führt Sie der Gefälleabstieg nicht zum zweiten Tal, und das Gleiche gilt, wenn Sie beginnen, das zweite Tal hinunterzugehen, Sie bleiben in diesem zweiten Minimum und finden nicht den Weg in das erste lokale Minimum. Dies ist eine interessante Eigenschaft des Gradientenabstiegsalgorithmus, über die Sie später mehr erfahren werden. In diesem Video haben Sie gesehen, wie Ihnen der Gefälleabstieg beim Bergabfahren hilft. Schauen wir uns im nächsten Video die mathematischen Ausdrücke an, die Sie implementieren können, damit der Gradientenabstieg funktioniert.

Implementierung des Gradientenabstiegs

Schauen wir uns an, wie Sie den Gradientenabstiegsalgorithmus tatsächlich implementieren können. Lassen Sie mich den Gradientenabstiegsalgorithmus aufschreiben. Hier ist es.



Bei jedem Schritt wird der Parameter w auf den alten Wert von w minus Alpha mal diesem Term d/dw der Kostenfunktion J(wb) aktualisiert. Was dieser Ausdruck sagt, ist, dass Sie nach Ihrem Parameter w den aktuellen Wert von w nehmen und ihn um einen kleinen Betrag anpassen, was diesem Ausdruck auf der rechten Seite minus Alpha mal diesem Term hier entspricht. Erstens diese Gleichheitsschreibweise hier. Da ich nun sagte, dass wir den wa-Wert mithilfe dieses Gleichheitszeichens zuweisen, ist dieses Gleichheitszeichen in diesem Zusammenhang der Zuweisungsoperator.

Wenn Sie in diesem Zusammenhang insbesondere Code schreiben, der besagt, dass a = c ist, bedeutet dies, dass Sie den Wert c nehmen und ihn in Ihrem Computer in der Variablen a speichern. Oder wenn Sie schreiben, dass a =a+1 ist, bedeutet dies, dass der Wert von a = a + 1 gesetzt wird oder der Wert von a um eins erhöht wird.

Die Kodierung des Zuweisungsoperators unterscheidet sich von Wahrheitsbehauptungen in der Mathematik. Wenn ich schreibe, dass a = c ist, behaupte ich, dass die Werte von a und c einander gleich sind. Hoffentlich werde ich nie eine Wahrheitsbehauptung schreiben, die gleich a plus 1 ist, denn das kann einfach nicht wahr sein. In Python und anderen Programmiersprachen werden Wahrheitsbehauptungen manchmal als „equals equal“ geschrieben, sodass Sie möglicherweise sehen, dass „a = c“ bedeutet, wenn Sie testen, ob „a“ = „c“ ist. Aber in der mathematischen Notation, wie wir sie üblicherweise verwenden, wie in diesen Videos, kann das Gleichheitszeichen entweder für Aufgaben oder zur Wahrheitsbehauptung verwendet werden. Ich versuche sicherzustellen, dass mir beim

Schreiben eines Gleichheitszeichens klar ist, ob wir einer Variablen einen Wert zuweisen oder ob wir die Wahrheit der Gleichheit zweier Werte behaupten.

Tauchen Sie nun tiefer in die Bedeutung der Symbole in dieser Gleichung ein. Das Symbol hier ist das griechische Alphabet A. In dieser Gleichung wird α auch als Lernrate bezeichnet. Die Lernrate ist normalerweise eine kleine positive Zahl zwischen 0 und 1 und könnte beispielsweise 0,01 betragen.

Was α tut, ist im Wesentlichen, dass es steuert, wie groß der Schritt ist, den Sie bergab machen. Wenn α sehr groß ist, entspricht das einem sehr aggressiven Gefälleabstieg, bei dem Sie versuchen, große Schritte bergab zu machen. Wenn α sehr klein ist, gehen Sie kleine Schritte bergab. Wir werden später noch einmal darauf zurückkommen, um uns eingehender mit der Auswahl einer guten α -Lernrate zu befassen. Schließlich ist dieser Term hier der Ableitungsterm der Kostenfunktion J. Machen wir uns jetzt keine Gedanken über die Details dieser Ableitung. Aber später erfahren Sie mehr über den Ableitungsterm. Aber im Moment können Sie sich diesen abgeleiteten Begriff, um den ich ein magentafarbenes Kästchen gezeichnet habe, so vorstellen, dass er Ihnen sagt, in welche Richtung Sie Ihren kleinen Schritt machen möchten. In Kombination mit der Lernrate α bestimmt sie auch, wie groß die Schritte sind, die man bergab machen möchte.

Nun möchte ich erwähnen, dass Ableitungen aus der Analysis stammen. Denken Sie daran, dass Ihr Modell zwei Parameter hat, nicht nur w, sondern auch b. Sie haben auch eine Zuweisungsoperation, die den Parameter b aktualisiert, die sehr ähnlich aussieht. b wird der alte Wert von b minus der Lernrate α multipliziert mit diesem etwas anderen Ableitungsterm d/ db von J(wb) zugewiesen .

Denken Sie daran, dass Sie im Diagramm des Oberflächendiagramms kleine Schritte machen, bis Sie das untere Ende des Werts erreichen. Für den Gradientenabstiegsalgorithmus werden Sie diese beiden Aktualisierungsschritte wiederholen, bis der Algorithmus konvergiert. Mit konvergiert meine ich, dass Sie den Punkt bei einem lokalen Minimum erreichen, an dem sich die Parameter w und b mit jedem weiteren Schritt, den Sie unternehmen, nicht mehr wesentlich ändern.

Nun gibt es noch ein weiteres subtiles Detail zur korrekten Vorgehensweise beim semantischen Gradientenabstieg: Sie aktualisieren zwei Parameter, w und b. Diese Aktualisierung erfolgt für beide Parameter w und b.

Gradient descent is an algorithm for finding values of parameters w and b that minimize the cost function J. What does this update statement do? (Assume α is small.)

$$w = w - \alpha \frac{\partial J(w,b)}{\partial w}$$

- (●) Updates parameter w by a small amount
- ( ) Checks whether w is equal to $w - \alpha \frac{\partial J(w,b)}{\partial w}$

Ein wichtiges Detail ist, dass Sie für den Gradientenabstieg w und b gleichzeitig aktualisieren möchten, was bedeutet, dass Sie beide Parameter gleichzeitig aktualisieren möchten. Damit meine ich, dass Sie in diesem Ausdruck w vom alten w auf ein neues w aktualisieren und auch b von seinem ältesten Wert auf den neuen Wert von b aktualisieren. Der Weg, dies zu implementieren, besteht darin, die rechte Seite zu berechnen, diese Sache für w und b zu berechnen und gleichzeitig w und b auf die neuen Werte zu aktualisieren.

Werfen wir einen Blick darauf, was das bedeutet. Hier erfahren Sie, wie Sie den Gradientenabstieg richtig implementieren, der eine gleichzeitige Aktualisierung durchführt. Dadurch wird eine Variable temp_w gleich diesem Ausdruck gesetzt, der hier w minus diesem Term ist.

Dazu gibt es auch eine Menge in einer anderen Variablen temp_b, die b minus diesem Term ist. Sie berechnen beide für die Handseiten, beide Aktualisierungen und speichern sie in den Variablen temp_w und temp_b . Dann kopieren Sie den Wert von temp_w nach w, und Sie kopieren auch den Wert von temp_b nach b. Nun fällt Ihnen vielleicht auf, dass dieser Wert von w von für w aktualisiert wird. Hier ist mir aufgefallen, dass das w vor dem Update dort ist, wo es in den Ableitungsterm geht. Im Gegensatz dazu handelt es sich hier um eine falsche Implementierung des Gradientenabstiegs, die keine gleichzeitige Aktualisierung durchführt. In dieser falschen Implementierung berechnen wir temp_w wie zuvor, soweit ist das in Ordnung. Hier beginnt der Unterschied. Anschließend aktualisieren wir w mit dem Wert in temp_w, bevor wir den neuen Wert für den anderen Parameter berechnen. Als nächstes berechnen wir temp_b als b minus diesem Term hier und aktualisieren schließlich b mit dem Wert in temp_b.

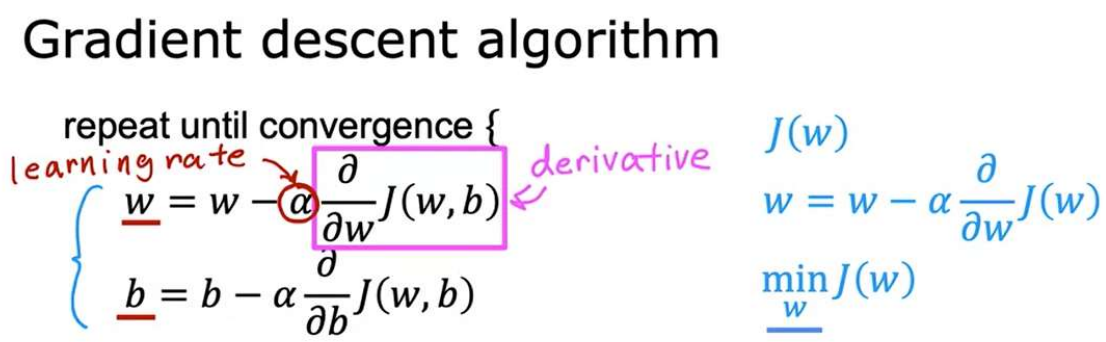
Der Unterschied zwischen der Implementierung auf der rechten und der linken Seite besteht darin, dass, wenn Sie hierher schauen, dieses w bereits auf diesen neuen Wert aktualisiert wurde, und dies ist das aktualisierte w, das tatsächlich in die Kostenfunktion j von w eingeht. B. Das bedeutet, dass dieser Begriff hier rechts nicht mit dem Begriff hier links übereinstimmt. Das bedeutet auch, dass dieser temp_b- Term auf der rechten Seite nicht ganz mit dem temp_b-Term auf der linken Seite übereinstimmt und daher dieser aktualisierte Wert für b auf der rechten Seite nicht mit diesem aktualisierten Wert für die Variable b auf der linken Seite übereinstimmt.

Aufgrund der Art und Weise, wie der Gradientenabstieg im Code implementiert wird, erweist es sich tatsächlich als natürlicher, ihn bei gleichzeitigen Aktualisierungen richtig zu implementieren. Wenn jemand vom Gradientenabstieg spricht, meint er immer den Gradientenabstieg, bei dem Sie gleichzeitig eine Aktualisierung der Parameter durchführen.

Wenn Sie jedoch ein nicht gleichzeitiges Update implementieren, stellt sich heraus, dass es wahrscheinlich ohnehin mehr oder weniger funktioniert. Aber es auf diese Weise zu tun, ist nicht wirklich die richtige Art, es zu implementieren, sondern es handelt sich tatsächlich um einen anderen Algorithmus mit anderen Eigenschaften. Ich würde Ihnen raten, sich einfach an das richtige gleichzeitige Update zu halten und nicht diese falsche Version auf der rechten Seite zu verwenden.

Intuition des Gradientenabstiegs

Lassen Sie uns nun tiefer in den Gradientenabstieg eintauchen, um eine bessere Vorstellung davon zu bekommen, was er tut und warum er sinnvoll sein könnte. Hier ist der Gradientenabstiegsalgorithmus, den Sie im vorherigen Video gesehen haben. Zur Erinnerung: Diese Variable, dieses griechische Symbol α, ist die Lernrate.



Die Lernrate steuert, wie groß der Schritt ist, den Sie machen, wenn Sie die Parameter w und b des Modells aktualisieren. Dieser Term hier, dieses d über dw, das ist ein abgeleiteter Term. Gemäß der Konvention in der Mathematik wird dieses d hier in dieser lustigen Schriftart geschrieben. Falls jemand, der sich das ansieht, einen Doktortitel in Mathematik hat oder ein Experte für multivariate Analysis ist, fragt er sich vielleicht: Das ist nicht die Ableitung, das ist die partielle Ableitung. Ja, sie haben Recht. Aber für die Zwecke der Implementierung eines Algorithmus für maschinelles Lernen nenne ich ihn einfach Ableitung.

Nehmen wir an, Sie haben eine Kostenfunktion J mit nur einem Parameter w, wobei w eine Zahl ist. Das bedeutet, dass der Gradientenabstieg jetzt so aussieht. W wird auf w minus der Lernrate α mal d über dw von J von w aktualisiert. Sie versuchen, die Kosten zu minimieren, indem Sie den Parameter w anpassen. Dies ähnelt unserem vorherigen Beispiel, indem wir b vorübergehend auf 0 gesetzt hatten, mit einem Parameter w anstelle von zwei. Sie können zweidimensionale Diagramme der Kostenfunktion j anstelle von dreidimensionalen Diagrammen betrachten. Schauen wir uns an, was der Gradientenabstieg nur bei der Funktion J von w bewirkt. Hier ist auf der horizontalen Achse der Parameter w und auf der vertikalen Achse die Kosten j von w aufgetragen. Jetzt weniger initialisierter Gradientenabstieg mit einem Startwert für w. Lassen Sie es uns an dieser Stelle initialisieren. Stellen Sie sich vor, Sie beginnen an diesem Punkt genau hier mit der Funktion J. Der Gradientenabstieg führt dazu, dass w auf w minus Lernrate aktualisiert wird. Alpha mal d über dw von J von w. Schauen wir uns an, was dieser abgeleitete Begriff hier bedeutet. Eine Möglichkeit, über die Ableitung an diesem Punkt der Geraden nachzudenken, besteht darin, eine Tangente zu zeichnen, also eine gerade Linie, die diese Kurve an diesem Punkt berührt.

Genug, die Steigung dieser Geraden ist die Ableitung der Funktion j an diesem Punkt. Um die Steigung zu ermitteln, können Sie ein kleines Dreieck wie dieses zeichnen. Wenn Sie die Höhe dividiert durch die Breite dieses Dreiecks berechnen, ist das die Steigung. Diese Steigung könnte beispielsweise 2 über 1 betragen und wenn die Tangente nach oben und

rechts zeigt, ist die Steigung positiv, was bedeutet, dass diese Ableitung eine positive Zahl ist, also größer als 0. Das aktualisierte w wird w minus der Lernrate mal einer positiven Zahl sein. Die Lernrate ist immer eine positive Zahl. Wenn Sie w minus einer positiven Zahl nehmen, erhalten Sie einen neuen Wert für w, der kleiner ist. Wenn Sie sich im Diagramm nach links bewegen, verringern Sie den Wert von w. Möglicherweise stellen Sie fest, dass dies die richtige Vorgehensweise ist, wenn Ihr Ziel darin besteht, die Kosten J zu verringern, denn wenn wir uns auf dieser Kurve nach links bewegen, nehmen die Kosten j ab und Sie nähern sich dem Minimum für J ist hier drüben. Bisher scheint der Gradientenabstieg das Richtige zu sein.

Schauen wir uns nun ein weiteres Beispiel an. Nehmen wir die gleiche Funktion j von w wie oben und nehmen wir nun an, dass Sie den Gradientenabstieg an einer anderen Stelle initialisiert haben. Sagen wir, indem wir einen Startwert für w wählen, der hier links steht. Das ist dieser Punkt der Funktion j. Denken Sie daran, der Ableitungsterm ist d über dw von J von w, und wenn wir die Tangente an diesem Punkt hier betrachten, ist die Steigung dieser Linie eine Ableitung von J an diesem Punkt. Aber diese Tangente fällt nach rechts ab. Diese nach rechts abfallende Linie weist eine negative Steigung auf. Mit anderen Worten, die Ableitung von J ist an diesem Punkt eine negative Zahl. Wenn Sie beispielsweise ein Dreieck zeichnen, beträgt die Höhe negativ 2 und die Breite 1. Die Steigung beträgt negativ 2 geteilt durch 1, was negativ 2 ist, was einer negativen Zahl entspricht. Wenn Sie w aktualisieren, erhalten Sie w minus der Lernrate mal einer negativen Zahl. Das bedeutet, dass Sie von w eine negative Zahl subtrahieren.

Aber das Subtrahieren einer negativen Zahl bedeutet das Addieren einer positiven Zahl, und so erhöht sich am Ende w. Denn das Subtrahieren einer negativen Zahl ist dasselbe wie das Addieren einer positiven Zahl zu w. Dieser Schritt des Gradientenabfalls führt dazu, dass w zunimmt, was bedeutet, dass Sie sich im Diagramm nach rechts bewegen und Ihre Kosten J bis hierher gesunken sind. Auch hier sieht es so aus, als würde der Gradientenabstieg etwas Vernünftiges bewirken und Sie näher an das Minimum bringen. Hoffentlich zeigen diese beiden letzten Beispiele etwas von der Intuition, die dahinter steckt, was ein Ableitungsterm bewirkt und warum dieser Host-Gradientenabstieg w ändert, um Sie näher an das Minimum zu bringen.

Ich hoffe, dieses Video hat Ihnen einen Sinn dafür vermittelt, warum der Ableitungsterm beim Gradientenabstieg Sinn macht. Eine weitere Schlüsselgröße im Gradientenabstiegsalgorithmus ist die Lernrate α. Wie wählen Sie α aus? Was passiert, wenn es zu klein ist, oder was passiert, wenn es zu groß ist? Im nächsten Video werfen wir einen genaueren Blick auf den Parameter α, um eine Vorstellung davon zu bekommen, was er bewirkt und wie man eine gute Wahl für einen guten Wert von α für die Implementierung des Gradientenabstiegs trifft.

Lernrate

Die Wahl der Lernrate α hat einen großen Einfluss auf die Effizienz Ihrer Implementierung des Gradientenabstiegs. Und wenn α, die Lernrate schlecht gewählt ist, funktioniert die Abstiegsrate möglicherweise überhaupt nicht. In diesem Video werfen wir einen genaueren

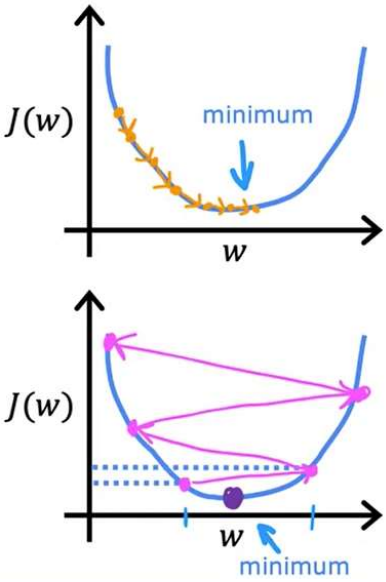
Blick auf die Lernrate. Dies wird Ihnen auch dabei helfen, bessere Lernraten für Ihre Implementierungen des Gradientenabstiegs zu wählen. Hier gilt also noch einmal die Gradientenabstiegsregel. W wird auf W minus der Lernrate, α multipliziert mit dem Ableitungsterm, aktualisiert.

$$w = w - \alpha \frac{d}{dw} J(w)$$

If α is too small...
Gradient descent may be slow.

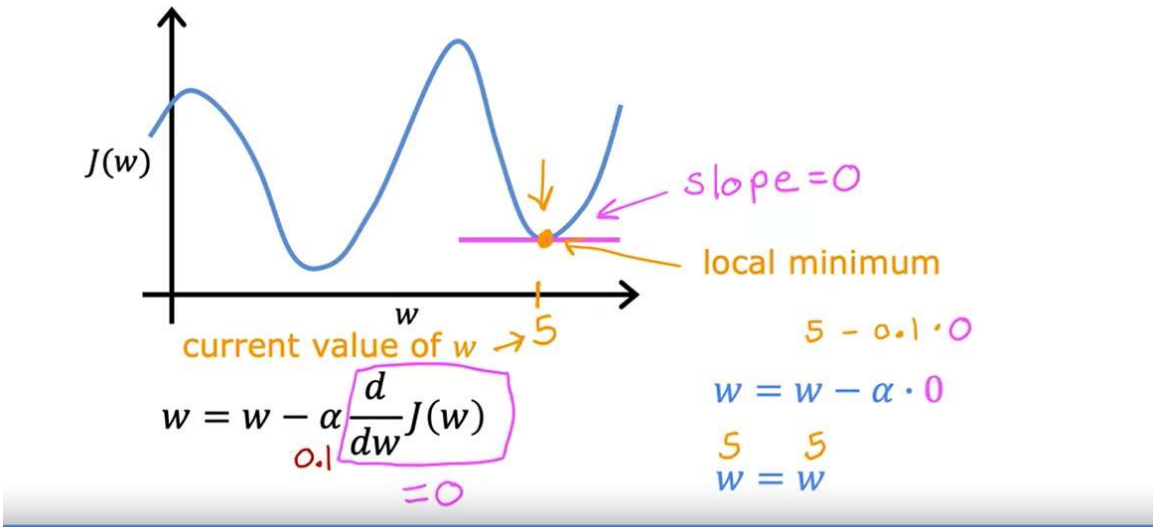
If α is too large...

Gradient descent may:
- Overshoot, never reach minimum
- Fail to converge, diverge



Erfahren Sie mehr darüber, was die Lernrate α bewirkt. Sehen wir uns an, was passieren könnte, wenn die Lernrate Alpha entweder zu klein oder zu groß ist. Für den Fall, dass die Lernrate zu gering ist. Hier ist ein Diagramm, bei dem die horizontale Achse W und die vertikale Achse die Kosten J sind. Und hier ist das Diagramm der Funktion J von W. Beginnen wir hier mit der Bewertung des Abstiegs, wenn die Lernrate zu gering ist.

Was dann passiert, ist, dass Sie Ihren Ableitungsterm mit einer wirklich sehr kleinen Zahl multiplizieren. Sie multiplizieren also mit der Zahl α. Das ist wirklich klein, etwa 0,0000001. Und so machen Sie am Ende so einen ganz kleinen, kleinen Schritt. Von diesem Punkt an werden Sie dann einen weiteren winzigen kleinen Schritt machen. Aber weil die Lernrate so gering ist, ist auch der zweite Schritt nur winzig. Das Ergebnis dieses Prozesses ist, dass Sie letztendlich die Kosten J senken, allerdings unglaublich langsam. Hier ist also ein weiterer Schritt und ein weiterer Schritt, ein weiterer kleiner Schritt, bis Sie sich endlich dem Minimum nähern. Aber wie Sie vielleicht bemerken, sind viele Schritte nötig, um das Minimum zu erreichen. Zusammenfassend lässt sich sagen, dass Gradientenabstiege funktionieren, wenn die Lernrate zu gering ist, aber langsam. Es wird sehr lange dauern, weil es diese winzigen Babyschritte erfordern. Und es werden viele Schritte nötig sein, bis es auch nur annähernd das Minimum erreicht.



Schauen wir uns nun einen anderen Fall an. Was passiert, wenn die Lernrate zu groß ist? Hier ist ein weiteres Diagramm der Kostenfunktion. Nehmen wir an, wir beginnen hier mit dem Gitterabstieg mit W bei diesem Wert. Es liegt also eigentlich schon ziemlich nah am Minimum. Die Deko zeigt also nach rechts. Wenn die Lernrate jedoch zu groß ist, aktualisieren Sie W in einem sehr großen Schritt, um bis hierher zu gelangen. Und das ist dieser Punkt hier auf der Funktion J. Sie bewegen sich also von diesem Punkt links bis zu diesem Punkt rechts. Und jetzt sind die Kosten tatsächlich noch schlimmer geworden.

Es ist gestiegen, weil es bei diesem Wert hier angefangen hat und nach einem Schritt tatsächlich auf diesen Wert hier gestiegen ist. Nun besagt die Ableitung an diesem neuen Punkt, dass W verringert werden soll, aber wenn die Lernrate zu groß ist. Dann können Sie von hier bis hierher einen großen Schritt machen. Jetzt sind Sie also an diesem Punkt angelangt, an dem es immer wieder zu einer zu großen Lernrate kommt. Dann macht man einen weiteren großen Beschleunigungsschritt und überschreitet das Minimum wieder deutlich. Jetzt sind Sie an diesem Punkt auf der rechten Seite und führen noch einmal ein weiteres Update durch. Und am Ende sind Sie hier angekommen, und jetzt sind Sie an diesem Punkt angelangt.

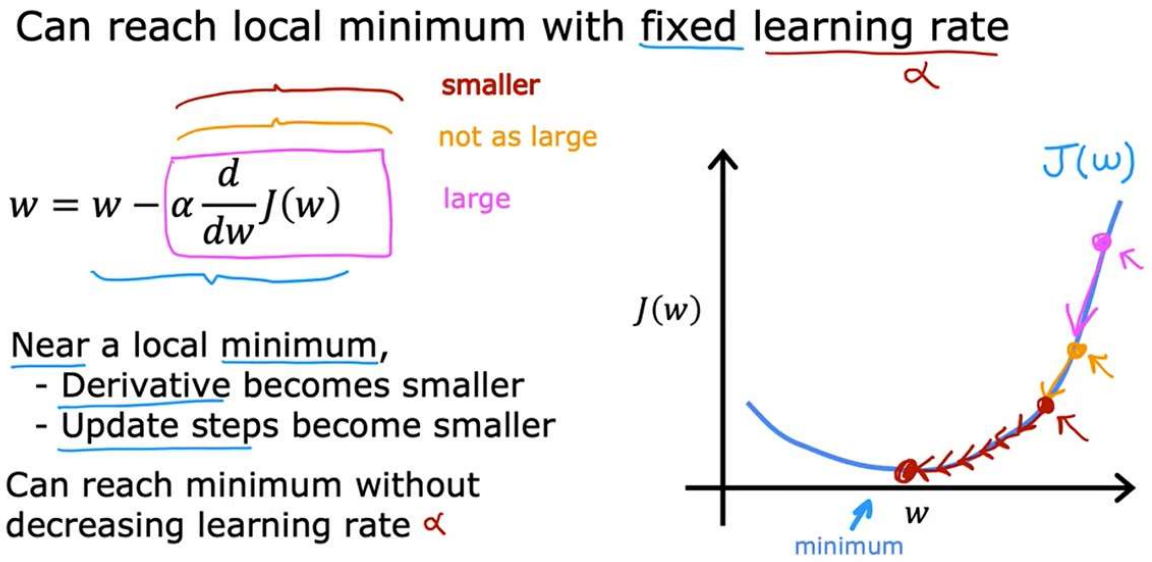
Wie Sie vielleicht bemerken, entfernen Sie sich tatsächlich immer weiter vom Minimum. Wenn also die Lernrate zu groß ist, kann es sein, dass die Sinnbildung überschießt und das Minimum nie erreicht. Anders ausgedrückt kann es sein, dass große Schnittpunkte möglicherweise nicht konvergieren und sogar divergieren. Hier ist also eine weitere Frage. Möglicherweise fragen Sie sich, ob sich einer Ihrer Parameter W bereits an dieser Stelle befindet. Damit sind Ihre Kosten J bereits auf einem lokalen Minimum. Was denken Sie? Ein Schritt des Gefälles reicht aus, wenn Sie bereits ein Minimum erreicht haben? Das ist also eine knifflige Angelegenheit. Als ich dieses Zeug zum ersten Mal lernte, dauerte es tatsächlich lange, bis ich es herausgefunden hatte.

Aber lasst uns das gemeinsam durcharbeiten. Nehmen wir an, Sie haben eine Kostenfunktion J. Und die, die Sie hier sehen, ist keine quadratische Fehlerkostenfunktion und diese Kostenfunktion hat zwei lokale Minima, die den beiden Tälern entsprechen, die Sie hier sehen. Nehmen wir nun an, dass Ihr Parameter W nach einigen Schritten des

Gradientenabstiegs hier drüben ist, sagen wir, gleich fünf. Das ist also der aktuelle Wert von W. Das bedeutet, dass Sie sich an diesem Punkt der Kostenfunktion J befinden. Und das ist zufällig ein lokales Minimum, wie sich herausstellt, wenn Sie die Aufmerksamkeit auf die Funktion an diesem Punkt lenken. Die Steigung dieser Geraden ist Null und damit der Ableitungsterm.

Hier ist gleich Null für den aktuellen Wert von W. Und so wird bei der Bewertungsabstiegsaktualisierung W auf W minus der Lernrate multipliziert mit Null aktualisiert. Wir sind hier, weil der Ableitungsterm gleich Null ist. Und das ist das Gleiche, als würde man sagen: „Setzen wir W gleich W“. Das bedeutet also, dass W beim Gradientenabstieg unverändert bleibt, wenn Sie bereits ein lokales Minimum erreicht haben. Weil dadurch lediglich der neue Wert von W so aktualisiert wird, dass er genau der gleiche alte Wert von W ist. Nehmen wir also konkret an, dass der aktuelle Wert von W fünf beträgt.

Und Alpha ist nach einer Iteration 0,1, Sie aktualisieren W als W minus Alpha mal Null und es ist immer noch gleich fünf. Wenn Ihre Parameter Sie also bereits auf ein lokales Minimum gebracht haben, führt ein weiterer Gradientenabstieg zu absolut nichts. Die gewünschten Parameter werden dadurch nicht geändert, da die Lösung auf diesem lokalen Minimum bleibt. Dies erklärt auch, warum der Gradientenabstieg selbst bei einer festen Lernrate Alpha ein lokales Minimum erreichen kann. Folgendes meine ich: Um dies zu veranschaulichen, schauen wir uns ein anderes Beispiel an. Hier ist die Kostenfunktion J von W, die wir minimieren möchten. Lassen Sie uns hier oben den Gradientenabstieg initialisieren. Wenn wir einen Aktualisierungsschritt unternehmen, gelangen wir möglicherweise an diesen Punkt.



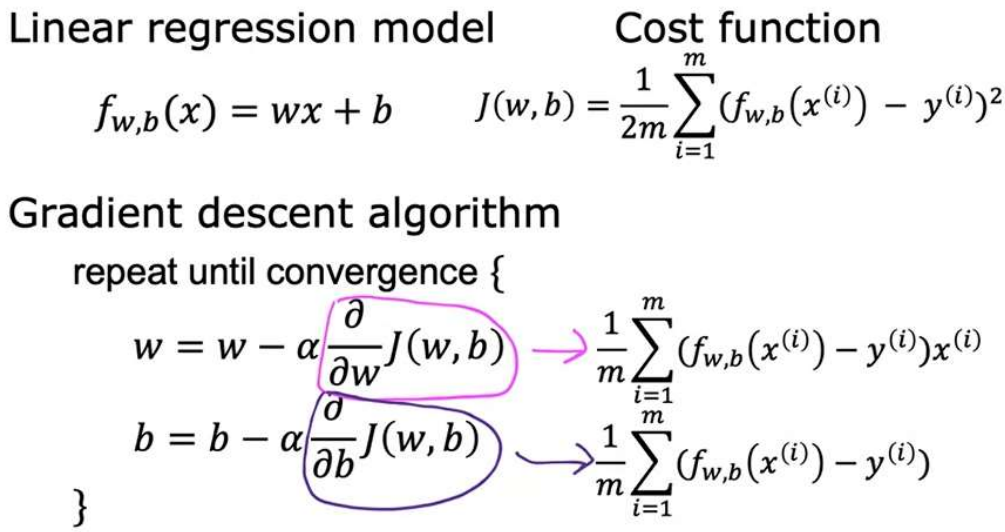
Und weil diese Ableitung ziemlich groß ist, macht der Abstieg einen relativ großen Schritt nach rechts. Jetzt sind wir an diesem zweiten Punkt angelangt, an dem wir einen weiteren Schritt machen. Und Sie werden vielleicht bemerken, dass der Hang nicht mehr so steil ist wie am ersten Punkt. Die Ableitung ist also nicht so groß. Daher wird der nächste Aktualisierungsschritt nicht so umfangreich sein wie der erste Schritt. Lesen Sie nun diesen

dritten Punkt hier und die Ableitung ist kleiner als im vorherigen Schritt. Und wir werden einen noch kleineren Schritt machen, wenn wir uns dem Minimum nähern.

Das Dekorative nähert sich immer mehr der Null. Während wir also den Gradientenabstieg ausführen, machen wir schließlich sehr kleine Schritte, bis wir schließlich ein lokales Minimum erreichen. Um es noch einmal zusammenzufassen: Je näher wir einem lokalen minimalen Gradienten kommen, desto kleiner werden die Schritte. Und das liegt daran, dass die Ableitung automatisch kleiner wird, wenn wir uns dem lokalen Minimum nähern. Und das bedeutet, dass auch die Aktualisierungsschritte automatisch kleiner werden. Auch wenn die Lernrate α auf einem festen Wert gehalten wird.

Gradientenabstieg für lineare Regression

Zuvor haben Sie sich das lineare Regressionsmodell angesehen, dann die Kostenfunktion und dann den Gradientenabstiegsalgorithmus. In diesem Video werden wir gemeinsam die quadratische Fehlerkostenfunktion für das lineare Regressionsmodell mit Gradientenabstieg verwenden.



Dies ermöglicht es uns, das lineare Regressionsmodell so zu trainieren, dass es an eine gerade Linie angepasst wird, um die Trainingsdaten zu erhalten. Lasst uns anfangen. Hier ist das lineare Regressionsmodell. Rechts ist die quadrierte Fehlerkostenfunktion. Unten finden Sie den Gradientenabstiegsalgorithmus. Wenn man diese Ableitungen berechnet, erhält man diese Terme. Die Ableitung nach W ist diese 1 über m, die Summe von i ist gleich 1 bis m. Dann der Fehlerterm, also die Differenz zwischen den vorhergesagten und den tatsächlichen Werten multipliziert mit dem Eingabemerkmal xi. Die Ableitung nach b ist diese Formel hier, die genauso aussieht wie die obige Gleichung, außer dass sie am Ende nicht den xi-Term hat. Wenn Sie diese Formeln verwenden, um diese beiden Ableitungen zu berechnen und den

Gradientenabstieg auf diese Weise implementieren, wird es funktionieren. Nun fragen Sie sich vielleicht: Woher habe ich diese Formeln? Sie werden mithilfe von Analysis abgeleitet. Wenn Sie die vollständige Ableitung sehen möchten, werde ich die Ableitung auf der nächsten Folie kurz durchgehen.

Aber wenn Sie sich nicht an die Infinitesimalrechnung erinnern oder sich nicht dafür interessieren, machen Sie sich darüber keine Sorgen. Sie können die Materialien auf der nächsten Folie vollständig überspringen und trotzdem den Gradientenabstieg implementieren und diesen Kurs beenden, und alles wird gut funktionieren. Auf dieser Folie, die eine der mathematischsten Folien der gesamten Spezialisierung ist und wiederum völlig optional ist, zeigen wir Ihnen, wie Sie die Ableitungsterme berechnen. Beginnen wir mit dem ersten Semester. Die Ableitung der Kostenfunktion J nach w. Wir beginnen mit der Definition der Kostenfunktion J. J von WP ist dies. 1 über 2m mal diese Summe der quadrierten Fehlerterme. Denken Sie nun auch daran, dass f von wb von X^i gleich diesem Term hier ist, der WX^i plus b ist. Was wir tun möchten, ist die Ableitung, auch partielle Ableitung nach w genannt, dieser Gleichung hier rechts zu berechnen.

Wenn Sie schon einmal an einem Analysis-Kurs teilgenommen haben und auch das ist völlig in Ordnung, wenn Sie das nicht getan haben, wissen Sie vielleicht, dass nach den Regeln der Analysis die Ableitung gleich diesem Term hier ist. Aus diesem Grund heben sich die beiden hier und die beiden hier auf und es bleibt die Gleichung übrig, die Sie auf der vorherigen Folie gesehen haben. Aus diesem Grund mussten wir Anfang dieser Woche die Kostenfunktion mit 1,5 finden, weil sie die partielle Ableitung übersichtlicher macht. Es hebt die beiden Werte auf, die sich aus der Berechnung der Ableitung ergeben. Für die andere Ableitung nach b ist das ganz ähnlich. Ich kann es so aufschreiben und noch einmal die Definition von f von X^i einsetzen, um diese Gleichung zu erhalten. Nach den Regeln der Analysis ist dies gleichbedeutend damit, dass am Ende kein X^i mehr steht. Die 2er heben ein kleines auf und Sie erhalten diesen Ausdruck für die Ableitung nach b. Jetzt haben Sie diese beiden Ausdrücke für die Ableitungen.

Sie können sie in den Gradientenabstiegsalgorithmus einbinden. Hier ist der Gradientenabstiegsalgorithmus für die lineare Regression. Diese Aktualisierungen an w und b führen Sie bis zur Konvergenz wiederholt durch. Denken Sie daran, dass dieses f(x) ein lineares Regressionsmodell ist, also gleich w mal x plus b. Dieser Ausdruck hier ist die Ableitung der Kostenfunktion nach w.

Dieser Ausdruck ist die Ableitung der Kostenfunktion nach b. Zur Erinnerung: Sie möchten w und b bei jedem Schritt gleichzeitig aktualisieren. Machen wir uns nun mit der Funktionsweise des Gradientenabstiegs vertraut. Eines der Dinge, die wir beim Gradientenabstieg gesehen haben, ist, dass es zu einem lokalen Minimum anstelle eines globalen Minimums führen kann. Ob das globale Minimum den Punkt bedeutet, der von allen möglichen Punkten den niedrigsten möglichen Wert für die Kostenfunktion J hat.

Diese Funktion hat mehr als ein lokales Minimum. Denken Sie daran: Je nachdem, wo Sie die Parameter w und b initialisieren, können Sie unterschiedliche lokale Minima erreichen.

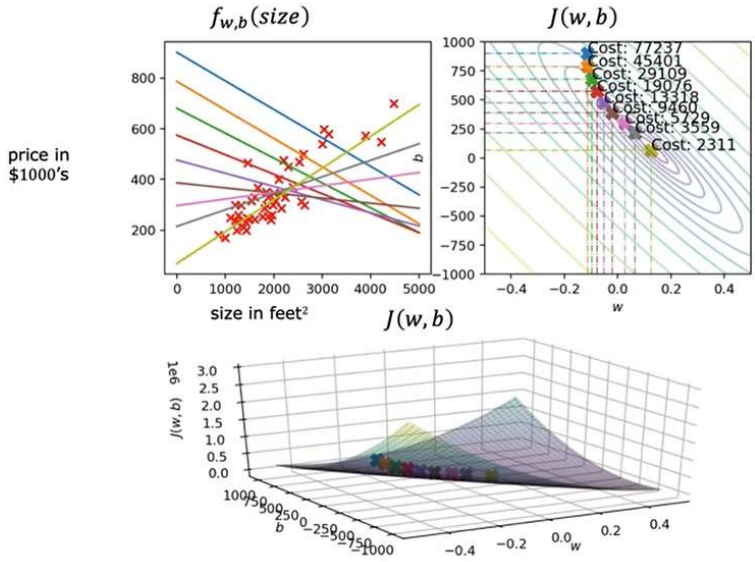
Du kannst hier landen, oder du kannst hier landen. Wenn Sie jedoch eine quadrierte Fehlerkostenfunktion mit linearer Regression verwenden, stellt sich heraus, dass die Kostenfunktion nicht mehrere lokale Minima aufweist und dies auch nie tun wird. Aufgrund dieser Schüsselform gibt es ein einziges globales Minimum. Der Fachbegriff dafür ist, dass diese Kostenfunktion eine konvexe Funktion ist.

Informell ist eine konvexe Funktion eine schalenförmige Funktion und kann keine anderen lokalen Minima als das einzelne globale Minimum haben. Wenn Sie den Gradientenabstieg für eine konvexe Funktion implementieren, besteht eine nette Eigenschaft darin, dass Ihre Lernrate immer zum globalen Minimum konvergiert, solange Sie sie entsprechend wählen. Herzlichen Glückwunsch, Sie wissen jetzt, wie Sie den Gradientenabstieg für die lineare Regression implementieren.

```python
def compute_gradient(x, y, w, b):
    """
    Computes the gradient for linear regression
    Args:
      x (ndarray (m,)): Data, m examples
      y (ndarray (m,)): target values
      w,b (scalar)    : model parameters
    Returns
      dj_dw (scalar): The gradient of the cost w.r.t. the parameters w
      dj_db (scalar): The gradient of the cost w.r.t. the parameter b
     """

    # Number of training examples
    m = x.shape[0]
    dj_dw = 0
    dj_db = 0

    for i in range(m):
        f_wb = w * x[i] + b
        dj_dw_i = (f_wb - y[i]) * x[i]
        dj_db_i = f_wb - y[i]
        dj_db += dj_db_i
        dj_dw += dj_dw_i
    dj_dw = dj_dw / m
    dj_db = dj_db / m

    return dj_dw, dj_db
```

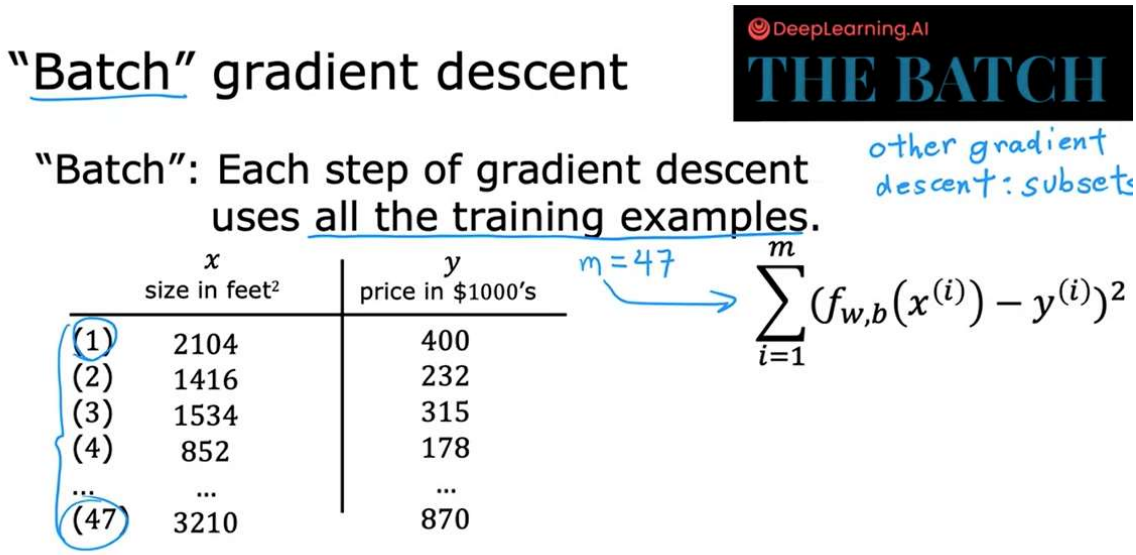
Laufender Gefälleabstieg

Mal sehen, was passiert, wenn Sie den Gradientenabstieg für lineare Regression laufen lassen. Sehen wir uns den Algorithmus in Aktion an. Hier ist eine Darstellung des Modells und Daten oben links und ein Konturdiagramm der Kosten Oben rechts befindet sich die Funktion und unten ist die Oberfläche Diagramm derselben Kostenfunktion. Oft w und b werden beide auf 0 initialisiert, aber für diese Demonstration: Initialisieren wir w = -0,1 und b = 900. Das entspricht also f(x) = -0,1x + 900.



Wenn wir jetzt einen Schritt machen Mithilfe des Gradientenabstiegs kamen wir schließlich hierher Punkt der Kostenfunktion hier bis zu diesem Punkt einfach nach unten und nach rechts und beachten Sie, dass die gerade Linie Auch die Passform wurde etwas verändert. Machen wir einen weiteren Schritt. Die Kostenfunktion hat jetzt in dieses Drittel verschoben und erneut die Funktion f(x) hat sich auch etwas verändert. Wenn Sie weitere dieser Schritte unternehmen, die Kosten sinken mit jedem Update. Also die Parameter w und b folgen dieser Flugbahn. Und wenn Sie nach links schauen, Sie erhalten diese entsprechende gerade Linienanpassung, die immer besser zu den Daten passt bis wir das globale Minimum erreicht haben.

Das ist also ein Gradientenabstieg und wir werden ihn passend nutzen ein Modell zu den Bestandsdaten. Und Sie können jetzt dieses f(x) verwenden **Modell, um den Preis des** Hauses oder der Immobilie Ihres Kunden vorherzusagen das Haus eines anderen. Zum Beispiel, wenn die Hausgröße 1250 Quadratmeter beträgt, können Sie jetzt den Wert ablesen und Sagen Sie, dass sie vielleicht, ich weiß nicht, 250.000 Dollar für das Haus bekommen könnten. Genauer gesagt, dieser Gefälleabstieg Der Prozess wird als Batch-Gradientenabstieg bezeichnet. Der Begriff „Batch Gradient Descent“ bezieht sich darauf auf die Tatsache, dass wir bei jedem Schritt des Gefälles hinschauen an allen Trainingsbeispielen statt nur an einer Teilmenge der Trainingsdaten.

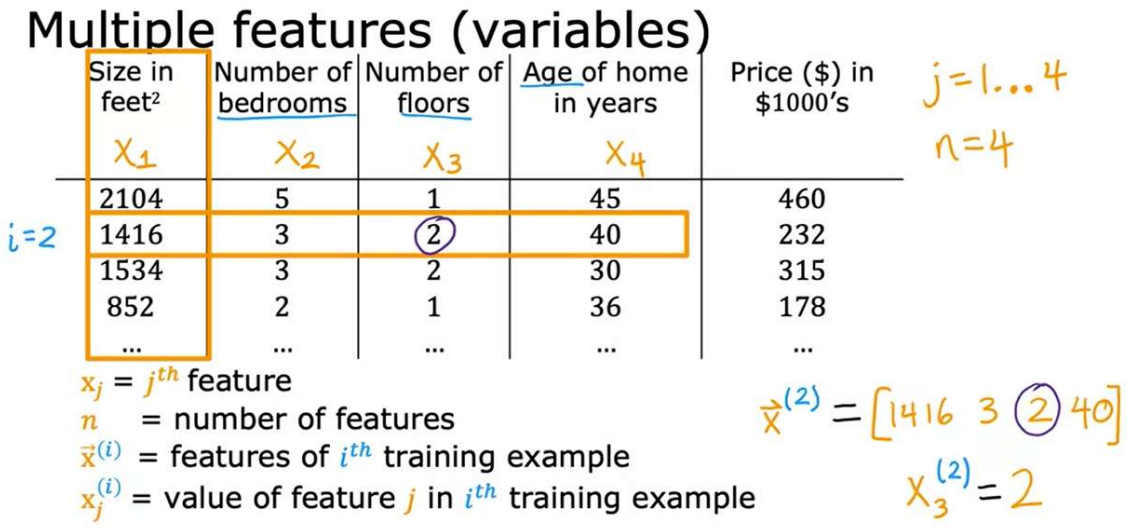


Bei der Berechnung des Notenabstiegs gilt also: bei der Berechnung von Ableitungen, bei der Berechnung der Summe von i =1 bis m. Und Batch-Gradientenabstieg ist Schauen Sie sich bei jedem Update den gesamten Stapel an Trainingsbeispielen an. Ich weiß, dass die Batch-Einstufung in Prozent möglich ist Es ist nicht der intuitivste Name, aber das ist es, was die Leute in der Maschine tun Lerngemeinschaft nennt es. Wenn Sie davon gehört haben der Newsletter The Batch, der von DeepLearning.AI veröffentlicht wird. Der Newsletter Die Charge wurde auch benannt für dieses Konzept im maschinellen Lernen. Und dann stellt sich heraus, dass es noch andere gibt Versionen des Gradientenabstiegs, die nicht den gesamten Trainingssatz betrachten, sondern stattdessen kleinere Teilmengen davon die Trainingsdaten bei jedem Aktualisierungsschritt. Aber wir werden dafür den Batch-Gradientenabstieg verwenden lineare Regression.

Multiple lineare Regression

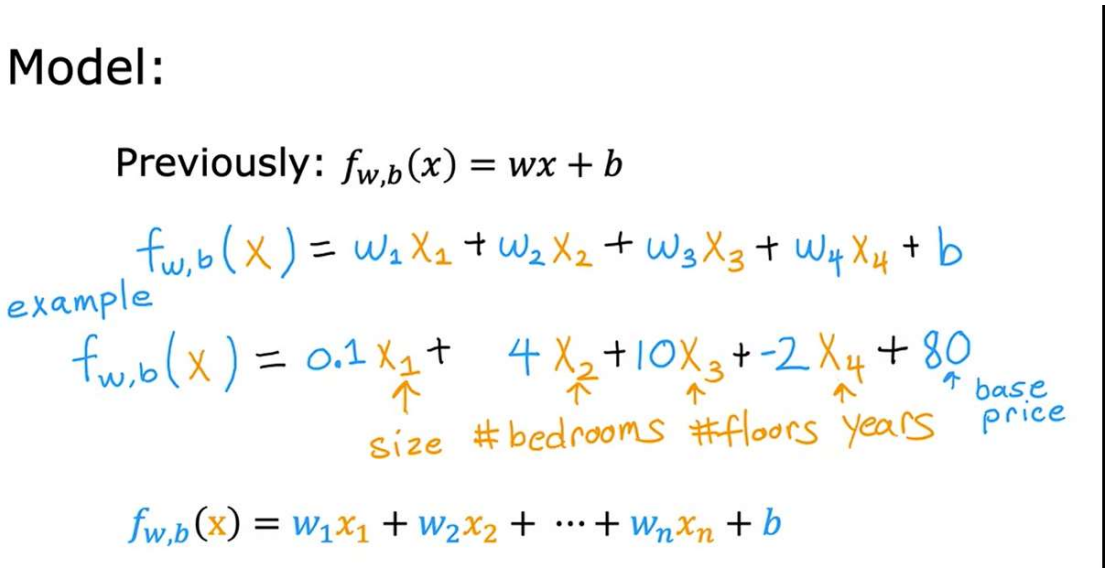
Mehrere Funktionen

In der ursprünglichen Version der linearen Regression hatten Sie ein einziges Merkmal x, die Größe des Hauses, und konnten y, den Preis des Hauses, vorhersagen. Das Modell war fwb von x gleich wx + b. Aber was wäre, wenn Sie nicht nur die Größe des Hauses als Merkmal hätten, um zu versuchen, den Preis vorherzusagen, sondern wenn Sie auch die Anzahl der Schlafzimmer, die Anzahl der Stockwerke und das Alter des Hauses in Jahren kennen würden? Es sieht so aus, als würde Ihnen dies viel mehr Informationen liefern, anhand derer Sie den Preis vorhersagen könnten. Um ein wenig neue Notation einzuführen, verwenden wir die Variablen X_1, X_2, X_3 und X_4, um die vier Features zu bezeichnen. Der Einfachheit halber führen wir etwas mehr Notation ein. Wir schreiben X subscript j oder manchmal sage ich auch kurz X sub j, um die Liste der Funktionen darzustellen. Hier wird j von eins auf vier gehen, weil wir vier Features haben.



Ich verwende den Kleinbuchstaben n, um die Gesamtzahl der Features anzugeben. In diesem Beispiel ist n also gleich 4. Wie zuvor verwenden wir das hochgestellte X i, um das i-te Trainingsbeispiel zu bezeichnen. Hier ist das hochgestellte Als konkretes Beispiel ist das hochgestellte i. Daher wird dies manchmal eher als Zeilenvektor als als Spaltenvektor bezeichnet. Aber wenn Sie den Unterschied nicht kennen, machen Sie sich keine Sorgen, er ist für diesen Zweck nicht so wichtig. Um im i-ten Trainingsbeispiel auf ein bestimmtes Feature zu verweisen, schreibe ich X hochgestellt i und tiefgestellt j, sodass zum Beispiel X hochgestellt 2 tiefgestellt 3 der Wert des dritten Features ist, also die Anzahl der Stockwerke im zweiten Training Beispiel und das wird also gleich 2 sein.

Um zu betonen, dass dieses Zeigen Sie, dass es sich hier um einen Vektor handelt, aber Sie müssen diesen Pfeil nicht in Ihre Notation einzeichnen. Sie können sich den Pfeil als optionalen Signifikanten vorstellen. Sie werden manchmal nur verwendet, um zu betonen, dass es sich um einen Vektor und nicht um eine Zahl handelt. Nachdem wir nun über mehrere Funktionen verfügen, werfen wir einen Blick darauf, wie ein Modell aussehen würde. Zuvor haben wir das Modell so definiert, wobei X ein einzelnes Merkmal, also eine einzelne Zahl, war. Aber jetzt, da es mehrere Funktionen gibt, werden wir es anders definieren. Stattdessen lautet das Modell:



fwb von X ist gleich w1x1 plus w2x2 plus w3x3 plus w4x4 plus b. Konkret für die Immobilienpreisvorhersage könnte ein mögliches Modell darin bestehen, dass wir den Preis des Hauses auf 0,1 mal X_1, die Größe des Hauses, plus viermal X_2, die Anzahl der Schlafzimmer, plus zehnmal X_3, die Anzahl der Stockwerke, schätzen. minus 2 mal X_4, das Alter des Hauses in Jahren plus 80. Lassen Sie uns ein wenig darüber nachdenken, wie Sie diese Parameter interpretieren könnten. Wenn das Modell versucht, den Preis des Hauses in Tausend Dollar vorherzusagen, können Sie sich b = 80 so vorstellen, dass der Grundpreis eines Hauses bei vielleicht 80.000 Dollar beginnt, vorausgesetzt, es hat keine Größe, keine

Schlafzimmer, nein Boden und kein Alter. Sie können sich diese 0,1 so vorstellen, dass sich der Preis für jeden zusätzlichen Quadratfuß um 0,1 $ 1.000 oder um 100 $ erhöht, denn wir sagen, dass sich der Preis für jeden Quadratfuß um 0,1 mal 1.000 $ erhöht, also 100 $. Möglicherweise erhöht sich der Preis für jedes weitere Badezimmer um 4.000 US-Dollar und für jede weitere Etage um 10.000 US-Dollar und für jedes weitere Lebensjahr des Hauses kann der Preis um 2.000 US-Dollar sinken, da der Parameter negativ 2 ist.

Im Allgemeinen gilt, wenn Sie haben n Features, dann sieht das Modell so aus. Hier ist noch einmal die Definition des Modells mit n Features. Als Nächstes führen wir ein wenig Notation ein, um diesen Ausdruck einfacher, aber gleichwertig umzuschreiben. Definieren wir W als eine Liste von Zahlen, die die Parameter W_1, W_2, W_3 bis W_n auflisten.

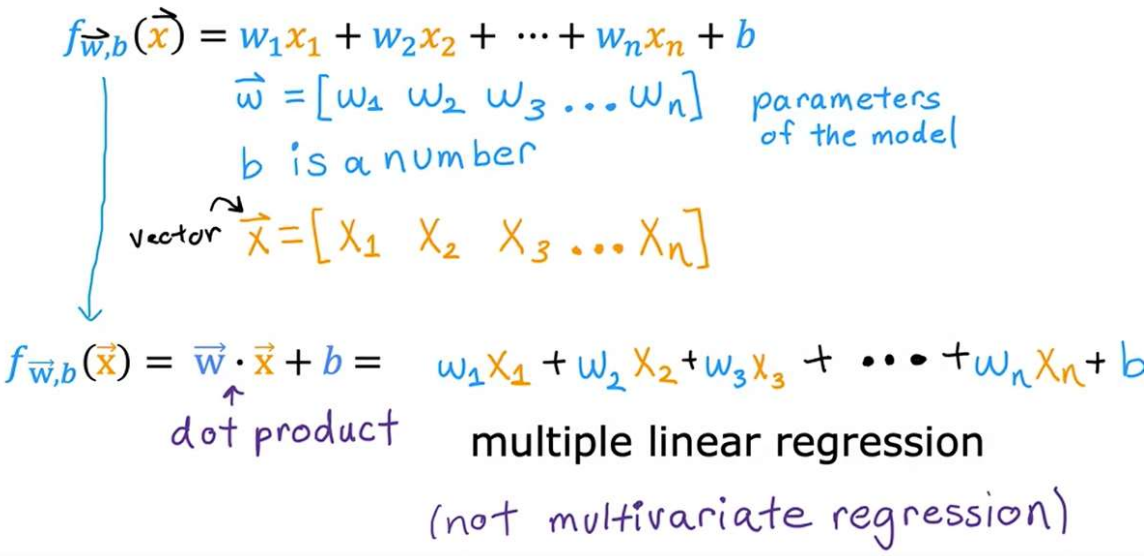
In der Mathematik nennt man das einen Vektor und manchmal zeichne ich oben einen kleinen Pfeil, um zu kennzeichnen, dass es sich um einen Vektor handelt, der einfach nur eine Liste von Zahlen bedeutet. Sie müssen diesen Pfeil nicht immer zeichnen und können dies in Ihrer eigenen Notation tun oder auch nicht. Sie können sich diesen kleinen Pfeil also nur als optionalen Hinweis vorstellen, der uns daran erinnert, dass es sich um einen Vektor handelt. Wenn Sie den Kurs „Lineare Algebra“ schon einmal besucht haben, erkennen Sie vielleicht, dass es sich hierbei um einen Zeilenvektor und nicht um einen Spaltenvektor handelt.

Aber wenn Sie nicht wissen, was diese Begriffe bedeuten, brauchen Sie sich darüber keine Sorgen zu machen. Als nächstes ist b wie zuvor eine einzelne Zahl und kein Vektor, und daher sind dieser Vektor W zusammen mit dieser Zahl b die Parameter des Modells. Lassen

Sie mich auch X als Liste oder Vektor schreiben, wiederum einen Zeilenvektor, der alle Merkmale X_1, X_2, um anzudeuten. In der Notation oben können wir hier und hier auch kleine Pfeile hinzufügen, um anzuzeigen, dass W und X tatsächlich diese Zahlenlisten sind, dass es sich tatsächlich um diese Vektoren handelt.

Mit dieser Notation kann das Modell nun prägnanter umgeschrieben werden, da f(x) gleich dem Vektor w ist und dieser Punkt sich auf ein Skalarprodukt aus der linearen Algebra von Was ist das für ein Skalarprodukt? Nun, die Skalarprodukte zweier Vektoren zweier Zahlenlisten W und X werden berechnet, indem die entsprechenden Zahlenpaare überprüft werden: W_1 und X_n multipliziert das und summiert dann alle diese Produkte.

Das bedeutet, dass das Skalarprodukt gleich W_1X_1 plus W_2X_2 plus W_3X_3 plus bis hin zu W_nX_n ist. Dann fügen wir schließlich wieder das b oben hinzu. Sie bemerken, dass wir dadurch genau den gleichen Ausdruck erhalten, den wir oben hatten. Mit der Dot-Traffic-Notation können Sie das Modell in einer kompakteren Form mit weniger Zeichen schreiben.



Der Name für diesen Typ eines linearen Regressionsmodells mit mehreren Eingabemerkmalen ist multiple lineare Regression. Dies steht im Gegensatz zur univariaten Regression, die nur ein Merkmal aufweist. Man könnte übrigens denken, dass dieser Algorithmus multivariate Regression genannt wird, aber dieser Begriff bezieht sich tatsächlich auf etwas anderes, das wir hier nicht verwenden werden. Ich werde dieses Modell als multiple lineare Regression bezeichnen.

Das ist alles für die lineare Regression mit mehreren Merkmalen, die auch als multiple lineare Regression bezeichnet wird. Um dies umzusetzen, gibt es einen wirklich tollen Trick namens Vektorisierung, der die Implementierung dieses und vieler anderer Lernalgorithmen viel einfacher macht. Fahren wir mit dem nächsten Video fort, um einen Blick darauf zu werfen, was Vektorisierung ist.

Vektorisierung Teil 1

Wenn Sie einen Lernalgorithmus implementieren, wird Ihr Code durch die Verwendung der Vektorisierung sowohl kürzer als auch wesentlich effizienter ausgeführt. Wenn Sie lernen, vektorisierten Code zu schreiben, können Sie auch die Vorteile moderner numerischer linearer Algebra-Bibliotheken und vielleicht sogar GPU-Hardware (Graphics Processing Unit) nutzen. Hierbei handelt es sich um Hardware, die objektiv darauf ausgelegt ist, die Computergrafik in Ihrem Computer zu beschleunigen. Es stellt sich jedoch heraus, dass sie auch beim Schreiben von vektorisiertem Code verwendet werden kann, um Ihnen dabei zu helfen, Ihren Code viel schneller auszuführen.

Schauen wir uns ein konkretes Beispiel dafür an, was Vektorisierung bedeutet. Hier ist ein Beispiel mit den Parametern w und b, wobei w ein Vektor mit drei Zahlen ist und Sie außerdem einen Vektor von Merkmalen x mit ebenfalls drei Zahlen haben. Hier ist n gleich 3. Beachten Sie, dass in der linearen Algebra der Index oder die Zählung bei 1 beginnt und der erste Wert daher mit w1 und x1 tiefgestellt ist. Im Python-Code können Sie diese Variablen w, b und x mithilfe von Arrays wie diesem definieren. Hier verwende ich tatsächlich eine numerische lineare Algebra-Bibliothek in Python namens **NumPy**, die bei weitem die am weitesten verbreitete numerische lineare Algebra-Bibliothek in Python und beim maschinellen Lernen ist.

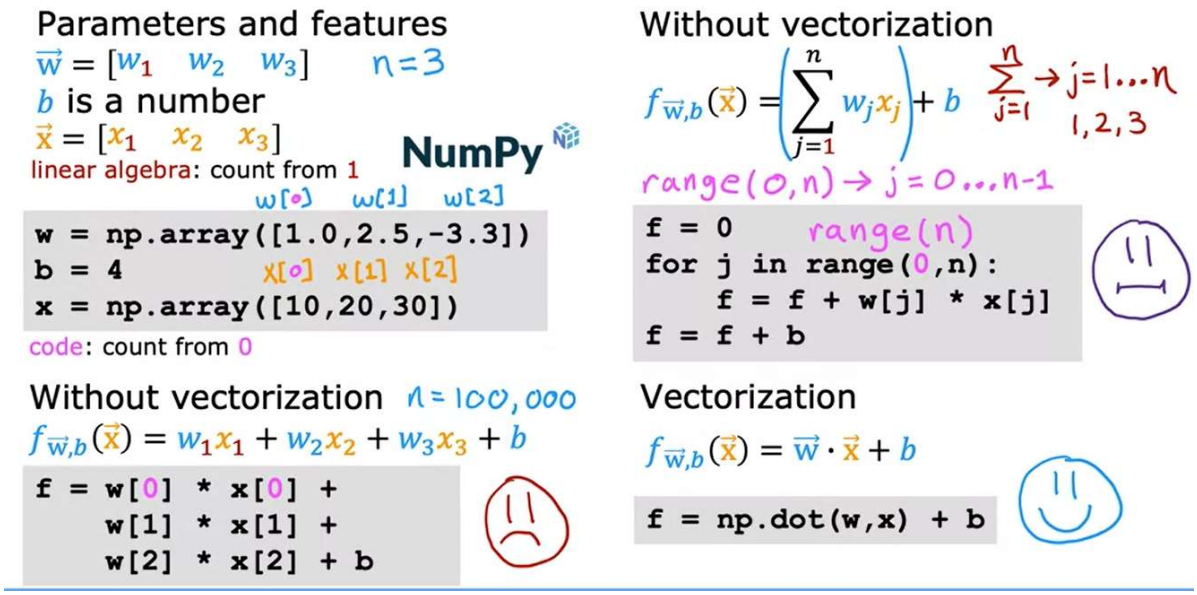
Da in Python die Indizierung von Arrays beim Zählen in Arrays bei 0 beginnt, würden Sie auf den ersten Wert von w zugreifen, indem Sie w in eckigen Klammern 0 verwenden. Auf den zweiten Wert würden Sie in w eckigen Klammern 1 und auf den dritten Wert in w eckigen Klammern 2 zugreifen Bei der Indizierung geht es hier von 0,1 auf 2 und nicht von 1, 2 auf 3. Um auf einzelne Merkmale von x zuzugreifen, verwenden Sie ebenfalls x0, x1 und x2. Viele Programmiersprachen, einschließlich Python, beginnen mit dem Zählen bei 0 statt bei 1.

Schauen wir uns nun eine Implementierung ohne Vektorisierung zur Berechnung der Modellvorhersage an. In Codes sieht es so aus. Sie nehmen jeden Parameter w und multiplizieren ihn mit seinem zugehörigen Merkmal. Nun könnten Sie Ihren Code so schreiben, aber wenn n nicht drei ist, sondern n 100 oder 100.000 ist, ist das sowohl für Ihren Code als auch für die Berechnung durch Ihren Computer ineffizient .

Hier ist ein anderer Weg. Ohne Vektorisierung, aber mit einer for- Schleife. In der Mathematik können Sie einen Summationsoperator verwenden, um alle Produkte von w_j und x_j zu addieren, wenn j = 1 bis n ist. Dann zitiere ich die Summe, die Sie am Ende mit b hinzufügen. Die Summierung geht von j gleich 1 bis einschließlich n. Für n gleich 3 geht j daher von 1, 2 auf 3. Im Code können Sie nach 0 initialisieren. Dann führt dies für j im Bereich von 0 bis n tatsächlich dazu, dass j von 0 auf n minus 1 geht. Von 0, 1 zu 2 können Sie dann zu f das Produkt aus w_j mal x_j addieren. Schließlich fügen Sie außerhalb der for-Schleife b hinzu. Beachten Sie, dass in Python der Bereich 0 bis n bedeutet, dass j von 0 bis n minus 1 reicht und n selbst nicht einschließt. Dies ist der geschriebene Bereich n in Python. Aber in diesem Video habe ich hier eine 0 hinzugefügt, nur um zu betonen, dass es bei 0 beginnt.

Diese Implementierung ist zwar etwas besser als die erste, verwendet jedoch immer noch keine Faktorisierung. Ist das nicht effizient? Schauen wir uns nun an, wie Sie dies mithilfe der Vektorisierung erreichen können. Dies ist der mathematische Ausdruck der Funktion f, die das Skalarprodukt von w und x plus b ist, und jetzt können Sie dies mit einer einzigen Codezeile implementieren, indem Sie fp gleich np berechnen.

Punkt Punkt, ich sagte Punkt Punkt, weil der erste Punkt ist der Punkt und der zweite Punkt ist die Funktion oder Methode namens DOT. Aber ist fp gleich np Punkt Punkt w Komma x und dies implementiert die mathematischen Skalarprodukte zwischen den Vektoren w und x. Dann können Sie am Ende schließlich b hinzufügen. Diese NumPy-Punktfunktion ist eine vektorisierte Implementierung der Skalarproduktoperation zwischen zwei Vektoren und läuft insbesondere dann, wenn n groß ist, viel schneller als die beiden vorherigen Codebeispiele.



Ich möchte betonen, dass die Vektorisierung tatsächlich zwei deutliche Vorteile hat. Erstens wird der Code dadurch kürzer, er besteht jetzt nur noch aus einer Codezeile. Ist das nicht cool? Zweitens führt dies auch dazu, dass Ihr Code viel schneller ausgeführt wird als bei den beiden vorherigen Implementierungen, die keine Vektorisierung verwendeten. Der Grund dafür, dass die vektorisierte Implementierung viel schneller ist, liegt im Hintergrund. Die NumPy-Dot-funktion ist in der Lage, parallele Hardware in Ihrem Computer zu verwenden. Dies gilt unabhängig davon, ob Sie sie auf einem normalen Computer ausführen, also auf einer normalen Computer-CPU, oder ob Sie häufig eine GPU, eine Grafikprozessoreinheit, verwenden. Wird verwendet, um maschinelle Lernaufgaben zu beschleunigen.

Die Fähigkeit der NumPy-Punktfunktion, parallele Hardware zu verwenden, macht sie viel effizienter als die for- Schleife oder die sequentielle Berechnung, die wir zuvor gesehen haben. Nun ist diese Version viel praktischer, wenn n groß ist, da Sie nicht w0 mal x0 plus w1 mal x1 plus viele zusätzliche Begriffe eingeben müssen, wie Sie es bei der vorherigen Version getan hätten. Dies spart zwar viel Tipparbeit, ist aber dennoch nicht besonders recheneffizient, da noch keine Vektorisierung zum Einsatz kommt. Zusammenfassend lässt sich sagen, dass die Vektorisierung Ihren Code kürzer macht, sodass er hoffentlich einfacher

zu schreiben und für Sie oder andere leichter zu lesen ist, und dass er außerdem viel schneller ausgeführt wird. Aber ehrlich, diese Magie hinter der Vektorisierung macht den Ablauf so viel schneller. Werfen wir einen Blick darauf, was Ihr Computer tatsächlich hinter den Kulissen tut, damit vektorisierter Code viel schneller ausgeführt wird.

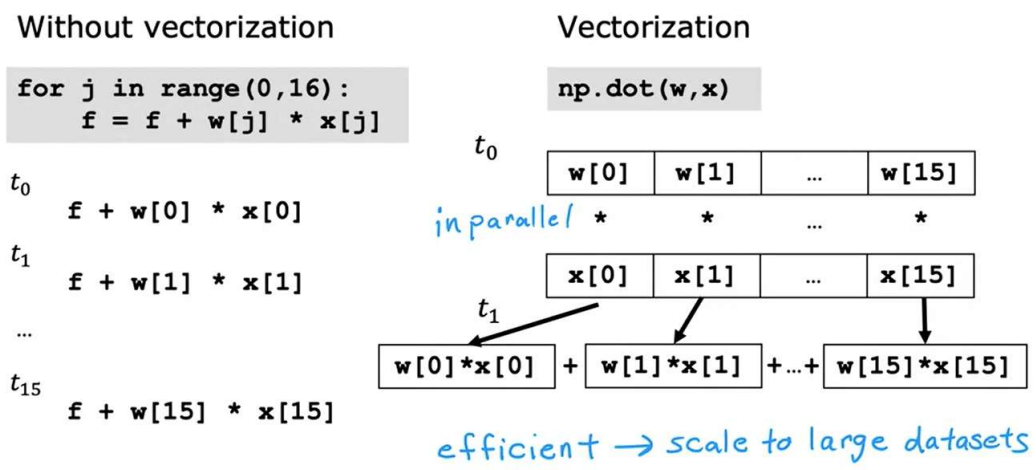
Vektorisierung Teil 2

Ich erinnere mich, als ich zum ersten Mal etwas über Vektorisierung lernte, verbrachte ich viele Stunden an meinem Computer, nahm eine nicht vektorisierte Version eines Algorithmus, ließ ihn laufen, schaute, wie lange er lief, und ließ dann eine vektorisierte Version des Codes laufen und sah, wie viel schneller das ging laufen, und ich habe einfach Stunden damit verbracht, damit zu spielen.

Und es hat mich ehrlich gesagt umgehauen, dass derselbe Algorithmus vektorisiert so viel schneller laufen würde. Für mich fühlte es sich fast wie ein Zaubertrick an. In diesem Video wollen wir herausfinden, wie dieser Zaubertrick wirklich funktioniert. Werfen wir einen genaueren Blick darauf, wie eine vektorisierte Implementierung hinter den Kulissen auf Ihrem Computer funktionieren kann. Schauen wir uns diese for-Schleife an. Die for- Schleife wie diese läuft ohne Vektorisierung. Wenn j zwischen 0 und beispielsweise 15 liegt, führt dieser Code die Operationen nacheinander aus. Auf dem ersten Zeitstempel, den ich als t0 schreiben werde.

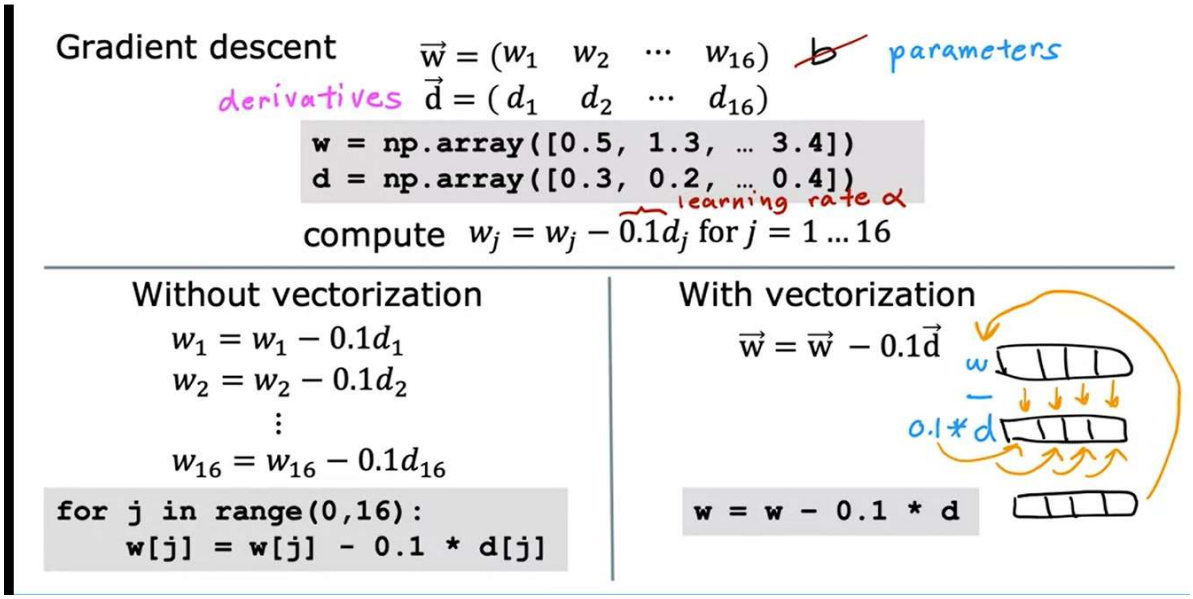
Es bearbeitet zunächst die Werte bei Index 0. Im nächsten Zeitschritt berechnet es Werte, die Index 1 entsprechen, und so weiter bis zum 15. Schritt, wo es diese berechnet. Mit anderen Worten: Es berechnet diese Berechnungen Schritt für Schritt, einen Schritt nach dem anderen. Im Gegensatz dazu wird diese Funktion in NumPy mit Vektorisierung in der Computerhardware implementiert.

Der Computer kann alle Werte der Vektoren w und x abrufen und in einem einzigen Schritt jedes Paar von w und x gleichzeitig und parallel miteinander multiplizieren . Anschließend nimmt der Computer diese 16 Zahlen und addiert sie mithilfe spezieller Hardware sehr effizient zusammen, sodass zum Addieren dieser 16 Zahlen keine einzelnen Additionen nacheinander durchgeführt werden müssen.



Dies bedeutet, dass Code mit Vektorisierung Berechnungen in viel kürzerer Zeit durchführen können als Codes ohne Vektorisierung. Dies ist von größerer Bedeutung, wenn Sie Algorithmen für große Datenmengen ausführen oder versuchen, große Modelle zu trainieren, was beim maschinellen Lernen häufig der Fall ist. Aus diesem Grund war die Möglichkeit, Implementierungen von Lernalgorithmen zu vektorisieren, ein wichtiger Schritt, um Lernalgorithmen effizient laufen zu lassen und sich daher gut auf große Datensätze zu skalieren, mit denen viele moderne Algorithmen für maschinelles Lernen jetzt arbeiten müssen.

Schauen wir uns nun ein konkretes Beispiel an, wie dies bei der Implementierung der multiplen linearen Regression und dieser linearen Regression mit mehreren Eingabemerkmalen hilft. Angenommen, Sie haben ein Problem mit 16 Features und 16 Parametern, w1 bis w16, zusätzlich zum Parameter b. Sie berechnen 16 Ableitungsterme für diese 16 Gewichte und Codes. Möglicherweise speichern Sie die Werte von w und d in zwei np.arrays, wobei d die Werte der Ableitungen speichert. In diesem Beispiel ignoriere ich einfach den Parameter b.



Nun möchten Sie für jeden dieser 16 Parameter ein Update berechnen. W_j wird auf w_j minus der Lernrate aktualisiert, beispielsweise 0,1 mal d_j, für j von 1 bis 16. Bei der Codierung ohne Vektorisierung würden Sie etwa Folgendes tun. Aktualisieren Sie w1 auf w1 minus der Lernrate 0,1 mal d1, aktualisieren Sie anschließend w2 auf ähnliche Weise und so weiter bis w16, aktualisiert als w16 minus 0,1 mal d16. Für die Codierung ohne Vektorisierung können Sie eine for-Schleife wie diese für j im Bereich 016 verwenden, der wiederum von 0-15 reicht, wobei w_j gleich w_j minus 0,1 mal d_j ist.

Im Gegensatz dazu kann man sich bei der Faktorisierung die Parallelverarbeitungshardware des Computers so vorstellen. Es nimmt alle 16 Werte im Vektor w und subtrahiert parallel das 0,1-fache aller 16 Werte im Vektor d und weist alle 16 Berechnungen gleichzeitig und in einem Schritt wieder auf wall zurück. Im Code können Sie dies wie folgt implementieren: w wird w minus 0,1 mal d zugewiesen. Hinter den Kulissen verwendet der Computer diese

NumPy-Arrays w und d und verwendet parallele Verarbeitungshardware, um alle 16 Berechnungen effizient auszuführen.

Mit einer vektorisierten Implementierung sollten Sie eine wesentlich effizientere Implementierung der linearen Regression erhalten. Vielleicht ist der Geschwindigkeitsunterschied nicht groß, wenn Sie 16 Features haben, aber wenn Sie Tausende von Features und möglicherweise sehr große Trainingssätze haben, wird diese Art der vektorisierten Implementierung einen großen Unterschied in der Laufzeit Ihres Lernalgorithmus machen. Es könnte den Unterschied zwischen Codes ausmachen, die in ein oder zwei Minuten fertig sind, und denen, die viele Stunden brauchen, um das Gleiche zu tun. In der optionalen Übung, die diesem Video folgt, sehen Sie eine Einführung in eine der am häufigsten verwendeten Python-Bibliotheken und maschinelles Lernen, die wir bereits in diesem Video namens NumPy angesprochen haben.

Sie sehen, wie sie Vektoren kodieren und diese Vektoren oder Zahlenlisten NumPy-Arrays nennen, und Sie sehen auch, wie man das Skalarprodukt zweier Vektoren mithilfe einer NumPy-Funktion namens dot bildet. Sie erfahren auch, wie vektorisierter Code, z. B. die Verwendung der Punktfunktion, viel schneller ausgeführt werden kann als eine for-Schleife. Tatsächlich könnten Sie diesen Code selbst timen und hoffentlich sehen, dass er viel schneller läuft. Dieses optionale Labor führt eine Menge neuer NumPy-Syntax ein.

Wenn Sie also das optionale Labor durchlesen, haben Sie immer noch das Gefühl, dass Sie den gesamten Code sofort verstehen müssen, aber Sie können dieses Notizbuch speichern und es als Referenz zum Ansehen verwenden wenn Sie mit Daten arbeiten, die in NumPy-Arrays gespeichert sind. Herzlichen Glückwunsch zum Abschluss dieses Videos zur Vektorisierung. Sie haben eine der wichtigsten und nützlichsten Techniken zur Implementierung von Algorithmen für maschinelles Lernen erlernt. Im nächsten Video kombinieren wir die Mathematik der multiplen linearen Regression mit der Vektorisierung, sodass Sie den Gradientenabstieg für die multiple lineare Regression mit Vektorisierung beeinflussen können. Kommen wir zum nächsten Video.

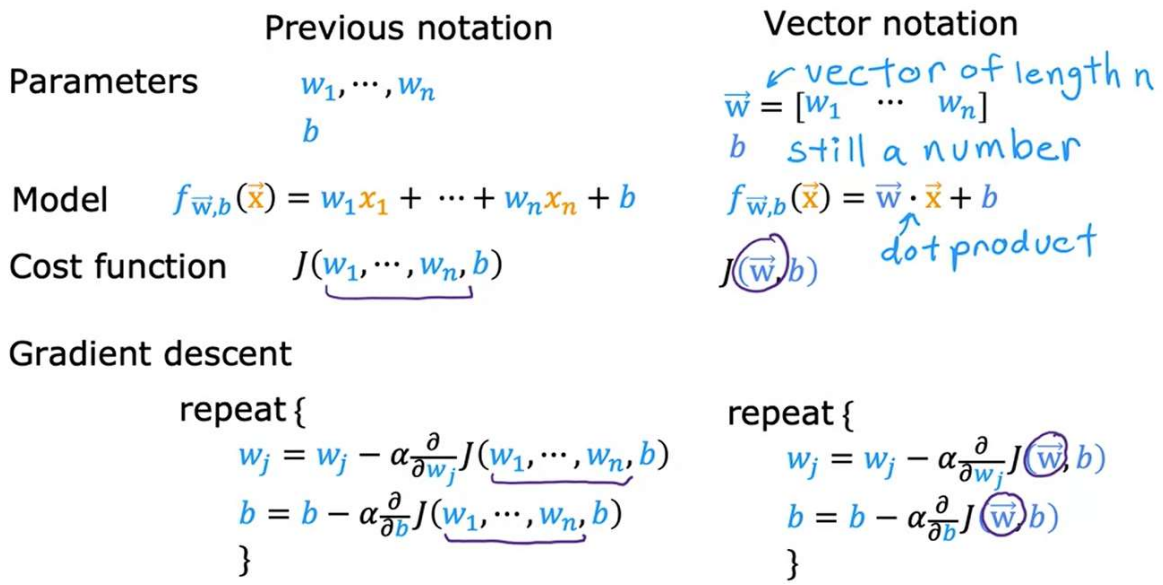
```
def compute_cost(X, y, w, b):
    """
    compute cost
    Args:
      X (ndarray (m,n)): Data, m examples with n features
      y (ndarray (m,)) : target values
      w (ndarray (n,)) : model parameters
      b (scalar)       : model parameter

    Returns:
      cost (scalar): cost
    """
    m = X.shape[0]
    cost = 0.0
    for i in range(m):
        f_wb_i = np.dot(X[i], w) + b           #(n,)(n,) = scalar (see np.dot)
        cost = cost + (f_wb_i - y[i])**2       #scalar
    cost = cost / (2 * m)                      #scalar
    return cost
```

Gradientenabstieg für die multiple lineare Regression

Sie haben etwas über den Gradientenabstieg bei der multiplen linearen Regression und auch über die Vektorisierung gelernt.

Lassen Sie uns alles zusammenfassen, um einen Gradientenabstieg für die multiple lineare Regression mit Vektorisierung zu implementieren. Sehen wir uns kurz an, wie die multiple lineare Regression aussieht. Sehen wir uns anhand unserer vorherigen Notation an, wie Sie es mithilfe der Vektornotation prägnanter schreiben können. Wir haben die Parameter w_1 bis w_n sowie b. Aber anstatt w_1 bis w_n als separate Zahlen, also separate Parameter, zu betrachten, beginnen wir damit, alle w in einem Vektor w zusammenzufassen, sodass w jetzt ein Vektor der Länge n ist.

Wir werden uns die Parameter dieses Modells einfach als einen Vektor w sowie b vorstellen, wobei b immer noch die gleiche Zahl wie zuvor ist. Während wir zuvor eine multiple lineare Regression wie diese finden mussten, können wir das Modell jetzt mithilfe der Vektornotation als f_w schreiben, b von x entspricht dem Skalarprodukt des Vektors w mit dem Vektor x plus b. Denken Sie daran, dass dieser Punkt hier „Produkt“ bedeutet. Unsere Kostenfunktion kann als J von w_1 bis w_n, b definiert werden. Aber anstatt uns J nur als Funktion dieser und verschiedener Parameter w_j sowie b vorzustellen, schreiben wir J als Funktion des Parametervektors w und der Zahl b. Dieses w_1 bis w_n wird durch diesen Vektor W ersetzt und J nimmt nun diese Eingabe des Vektors w und einer Zahl b und gibt eine Zahl zurück.

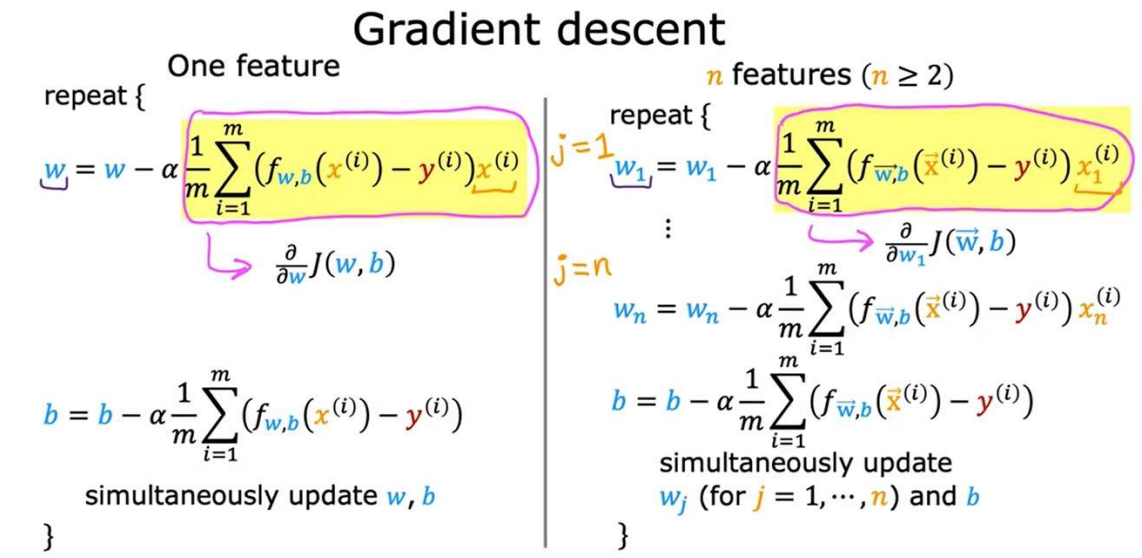


So sieht ein Gefälleabstieg aus. Wir werden jeden Parameter w_j wiederholt aktualisieren, sodass er w_j minus α mal der Ableitung der Kosten J ist, wobei J die Parameter w_1 bis w_n und b hat. Wir schreiben dies noch einmal einfach als J des Vektors w und der Zahl b. Sehen

wir uns an, wie das aussieht, wenn Sie den Gradientenabstieg implementieren, und werfen wir insbesondere einen Blick auf den Ableitungsterm. Wir werden sehen, dass der Gradientenabstieg bei mehreren Features ein wenig anders wird als bei nur einem Feature. Folgendes hatten wir, als wir einen Gradientenabstieg mit einer Funktion hatten. Wir hatten eine Aktualisierungsregel für w und eine separate Aktualisierungsregel für b.

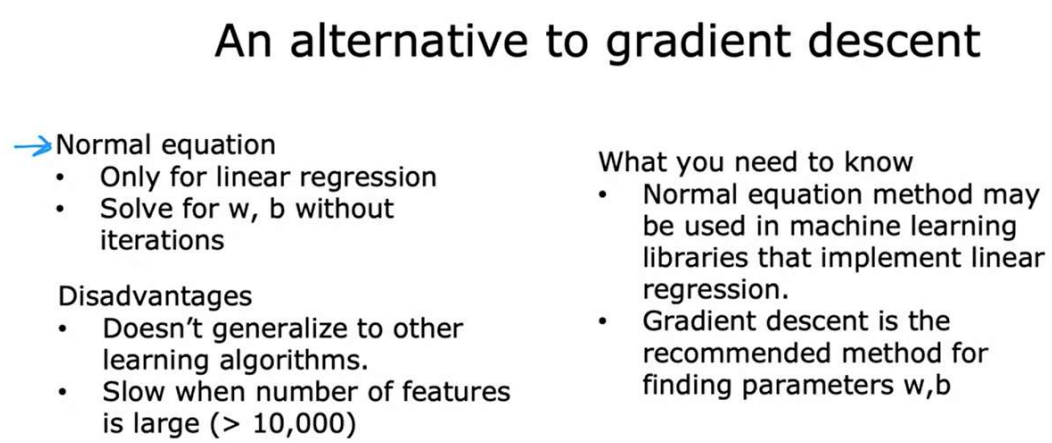
Hoffentlich kommen Ihnen diese bekannt vor. Dieser Term ist hier die Ableitung der Kostenfunktion J nach dem Parameter w. Ebenso haben wir eine Aktualisierungsregel für Parameter b. Bei der univariaten Regression hatten wir nur eine Funktion. Wir nennen dieses Feature xi ohne Index. Hier ist nun eine neue Notation für n Features, wobei n zwei oder mehr ist. Wir erhalten diese Aktualisierungsregel für den Gradientenabstieg. Aktualisieren Sie w_1 auf w_1 minus α mal diesen Ausdruck hier und diese Formel ist tatsächlich die Ableitung der Kosten J in Bezug auf w_1.

Die Formel für die Ableitung von J nach w_1 auf der rechten Seite sieht dem Fall eines Merkmals auf der linken Seite sehr ähnlich. Der Fehlerterm nimmt immer noch eine Vorhersage f(x) minus dem Ziel y an. Ein Unterschied besteht darin, dass w und x jetzt Vektoren sind und so wie w links jetzt hier rechts zu w_1 geworden ist, ist xi hier links jetzt stattdessen xi _1 hier rechts und dies ist nur für J gleich 1. Für Bei der multiplen linearen Regression haben wir J im Bereich von 1 bis n und daher aktualisieren wir die Parameter w_1, w_2 bis hin zu w_n, und dann aktualisieren wir wie zuvor b. Wenn Sie dies implementieren, erhalten Sie einen Gradientenabstieg für die mehrfache Regression. Das ist alles für den Gradientenabstieg für die multiple Regression.



Bevor ich mit diesem Video fortfahre, möchte ich eine kurze Randbemerkung zu einer alternativen Methode zum Ermitteln von w und b für die lineare Regression machen. Diese Methode wird Normalgleichung genannt. Während sich herausstellt, dass der Gradientenabstieg eine großartige Methode zur Minimierung der Kostenfunktion J ist, um w und b zu finden, gibt es einen anderen Algorithmus, der nur für die lineare Regression funktioniert, und so gut wie keinen der anderen Algorithmen, die Sie in dieser Spezialisierung zum Auflösen nach w sehen und b und diese andere Methode benötigt keinen iterativen Gradientenabstiegsalgorithmus.

Die so genannte Normalgleichungsmethode erweist sich als möglich, eine erweiterte lineare Algebra-Bibliothek zu verwenden, um in einem Ziel ohne Iterationen einfach nach w und b zu lösen. Einige Nachteile der Normalgleichungsmethode sind: Erstens wird dies im Gegensatz zum Gradientenabstieg nicht auf andere Lernalgorithmen verallgemeinert, wie etwa den logistischen Regressionsalgorithmus, den Sie nächste Woche kennenlernen werden, oder die neuronalen Netze oder andere Algorithmen, die Sie später in dieser Spezialisierung sehen werden.



Die normale Gleichungsmethode ist auch ziemlich langsam, wenn die Anzahl der Features groß ist. Fast niemand, der maschinelles Lernen anwendet, sollte die normale Gleichungsmethode selbst implementieren. Wenn Sie jedoch eine ausgereifte Bibliothek für maschinelles Lernen verwenden und die lineare Regression aufrufen, besteht die Möglichkeit, dass diese im Backend zur Lösung von w und b verwendet wird. Wenn Sie im Vorstellungsgespräch jemals den Begriff Normalgleichung hören, ist damit gemeint. Machen Sie sich keine Gedanken über die Details der Funktionsweise der Normalgleichung. Beachten Sie jedoch, dass einige Bibliotheken für maschinelles Lernen möglicherweise diese komplizierte Methode im Backend verwenden, um nach w und b zu suchen.

Aber für die meisten Lernalgorithmen, einschließlich der Art und Weise, wie Sie die lineare Regression selbst implementieren, bieten Gradientenabstiege eine bessere Möglichkeit, die Aufgabe zu erledigen. In der optionalen Übung, die diesem Video folgt, erfahren Sie, wie Sie eine Kodierung für ein multiples Regressionsmodell definieren und auch die Vorhersage f(x) berechnen. Außerdem erfahren Sie, wie Sie die Kosten berechnen und den Gradientenabstieg für ein multiples lineares Regressionsmodell implementieren. Dabei wird die NumPy-Bibliothek von Python verwendet. Wenn einer der Codes sehr neu aussieht, ist das in Ordnung, aber Sie können auch gerne einen Blick auf die vorherige optionale Übung werfen, in der NumPy und die Vektorisierung vorgestellt werden, um die NumPy-Funktionen aufzufrischen und zu erfahren, wie diese in Code implementiert werden.

Das ist es. Sie kennen jetzt die multiple lineare Regression. Dies ist heute wahrscheinlich der am weitesten verbreitete Lernalgorithmus der Welt. Aber es gibt noch mehr. Mit nur ein paar Tricks wie der richtigen Auswahl und Skalierung von Funktionen und der passenden

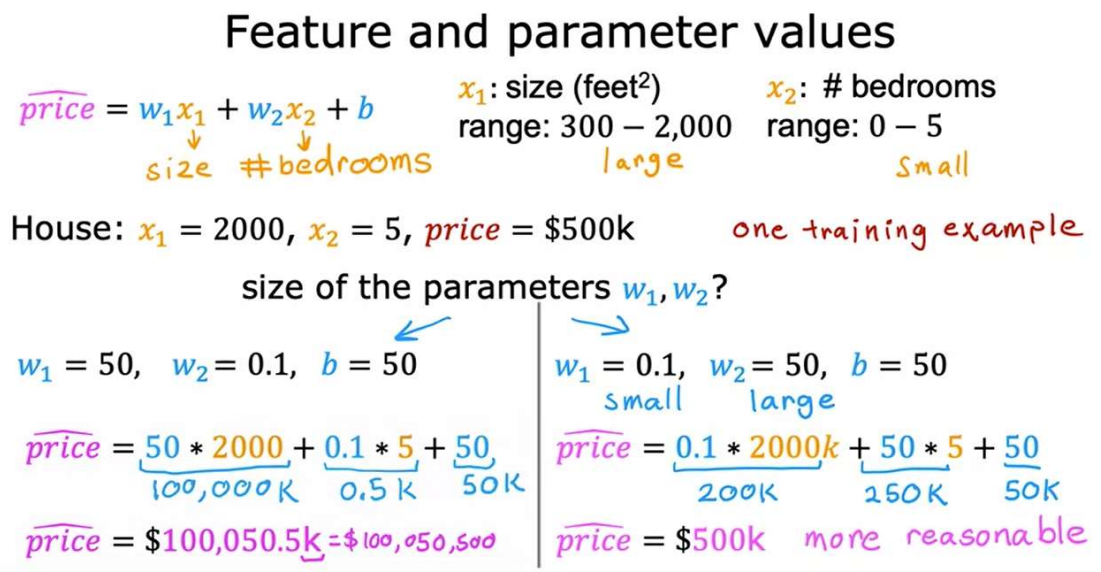
Auswahl der Lernrate α würden Sie die Arbeit wirklich viel besser machen. Nur noch ein paar Videos für diese Woche. Fahren wir mit dem nächsten Video fort, um die kleinen Tricks zu sehen, die Ihnen dabei helfen werden, dass die multiple lineare Regression viel besser funktioniert. Optionale Übung: Anweisungen zur multiplen linearen Regression In dieser optionalen Übung erfahren Sie, wie Sie ein multiples Regressionsmodell im Code definieren und wie Sie die Vorhersage f(x) berechnen.

Außerdem erfahren Sie, wie Sie die Kosten berechnen und den Gradientenabstieg für ein multiples lineares Regressionsmodell implementieren. Hierbei wird die Numpy-Bibliothek von Python verwendet. Wenn also Code sehr neu aussieht, schauen Sie sich bitte die vorherige optionale Übung an, in der Numpy und Vektorisierung vorgestellt werden, um die Numpy-Funktionen aufzufrischen und zu erfahren, wie diese in Code implementiert werden.

```
def compute_gradient(X, y, w, b):
    """
    Computes the gradient for linear regression
    Args:
      X (ndarray (m,n)): Data, m examples with n features
      y (ndarray (m,)) : target values
      w (ndarray (n,)) : model parameters
      b (scalar)       : model parameter

    Returns:
      dj_dw (ndarray (n,)): The gradient of the cost w.r.t. the parameters w.
      dj_db (scalar):       The gradient of the cost w.r.t. the parameter b.
    """
    m,n = X.shape           #(number of examples, number of features)
    dj_dw = np.zeros((n,))  # Initialisierung mit 0
    dj_db = 0.

    for i in range(m):
        err = (np.dot(X[i], w) + b) - y[i]
        for j in range(n):
            dj_dw[j] = dj_dw[j] + err * X[i, j]
        dj_db = dj_db + err
    dj_dw = dj_dw / m
    dj_db = dj_db / m

    return dj_db, dj_dw
```

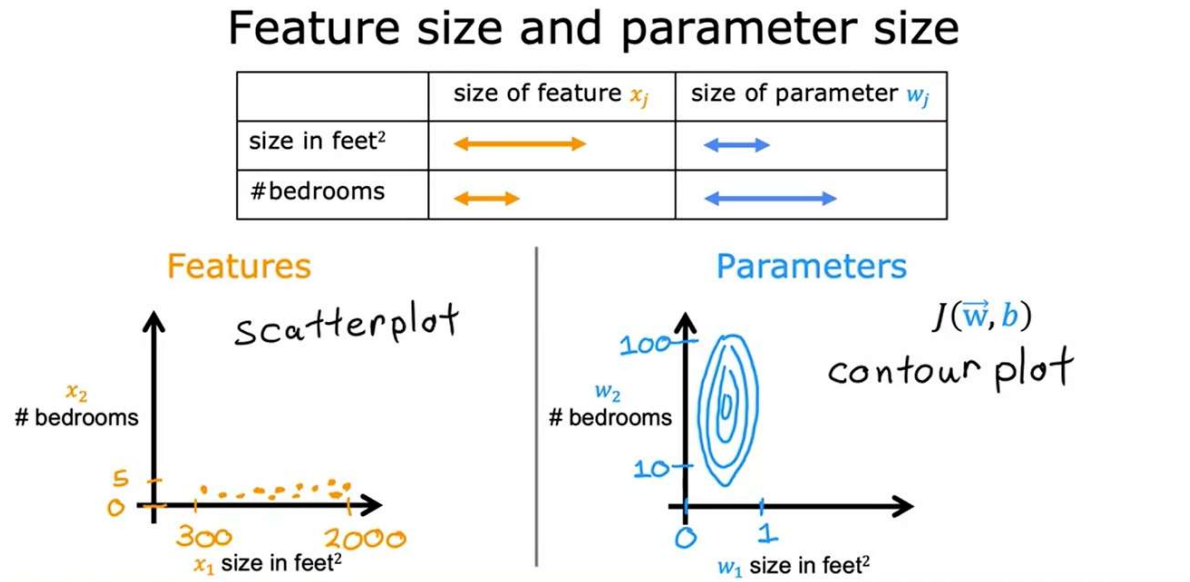
Feature-Skalierung Teil 1

Also willkommen zurück. Werfen wir einen Blick auf einige Techniken, die die Arbeit zwischen den Sinnen viel besser machen. In diesem Video sehen Sie eine Technik namens Feature-Skalierung, die einen viel schnelleren Verlauf des Gradientenabstiegs ermöglicht. Werfen wir zunächst einen Blick auf die Beziehung zwischen der Größe eines Features, d. h. wie groß die Zahlen für dieses Feature sind, und der Größe des zugehörigen Parameters.



Lassen Sie uns als konkretes Beispiel den Preis eines Hauses vorhersagen, indem wir zwei Merkmale verwenden: x1 die Größe des Hauses und x2 die Anzahl der Schlafzimmer. Nehmen wir an, dass x1 typischerweise zwischen 300 und 2000 Quadratfuß liegt. Und x2 im Datensatz reicht von 0 bis 5 Schlafzimmern. In diesem Beispiel nimmt also x1 einen relativ großen Wertebereich an und x2 einen relativ kleinen Wertebereich. Nehmen wir nun ein Beispiel für ein Haus mit einer Größe von 2000 Quadratmetern, fünf Schlafzimmern und einem Preis von 500.000 oder 500.000 US-Dollar. Was sind Ihrer Meinung nach für dieses eine Trainingsbeispiel sinnvolle Werte für die Größe der Parameter w1 und w2? Schauen wir uns einen möglichen Parametersatz an. Nehmen wir zur Diskussion an, w1 sei 50, w2 sei 0,1 und b sei 50. In diesem Fall beträgt der geschätzte Preis in Tausend Dollar hier also 100.000.000 plus 0,5.000 plus 50.000. Das sind etwas mehr als 100 Millionen Dollar. Das ist also eindeutig sehr weit vom tatsächlichen Preis von 500.000 US-Dollar entfernt.

Daher handelt es sich hier nicht um eine sehr gute Auswahl an Parametern für w1 und w2. Schauen wir uns nun eine andere Möglichkeit an. Angenommen, w1 und w2 wären umgekehrt. W1 ist 0,1 und w2 ist 50 und b ist immer noch auch 50. Bei dieser Wahl von w1 und w2 ist w1 relativ klein und w2 relativ groß, 50 ist viel größer als 0,1. Hier beträgt der vorhergesagte Preis also 0,1 mal 2000 plus 50 mal fünf plus 50. Der erste Term wird zu 200.000, der zweite Term zu 250.000 und das Plus zu 50. Diese Version des Modells sagt also einen Preis von 500.000 US-Dollar voraus, was viel vernünftiger ist Schätzung und entspricht zufällig dem tatsächlichen Preis des Hauses. Sie werden also hoffentlich bemerken, dass es wahrscheinlicher ist, dass ein gutes Modell lernt, einen relativ kleinen Parameterwert zu wählen, wenn der mögliche Wertebereich eines Features groß ist, etwa die Größe und die Quadratfuß, die bis zu 2000 reichen. wie 0,1.

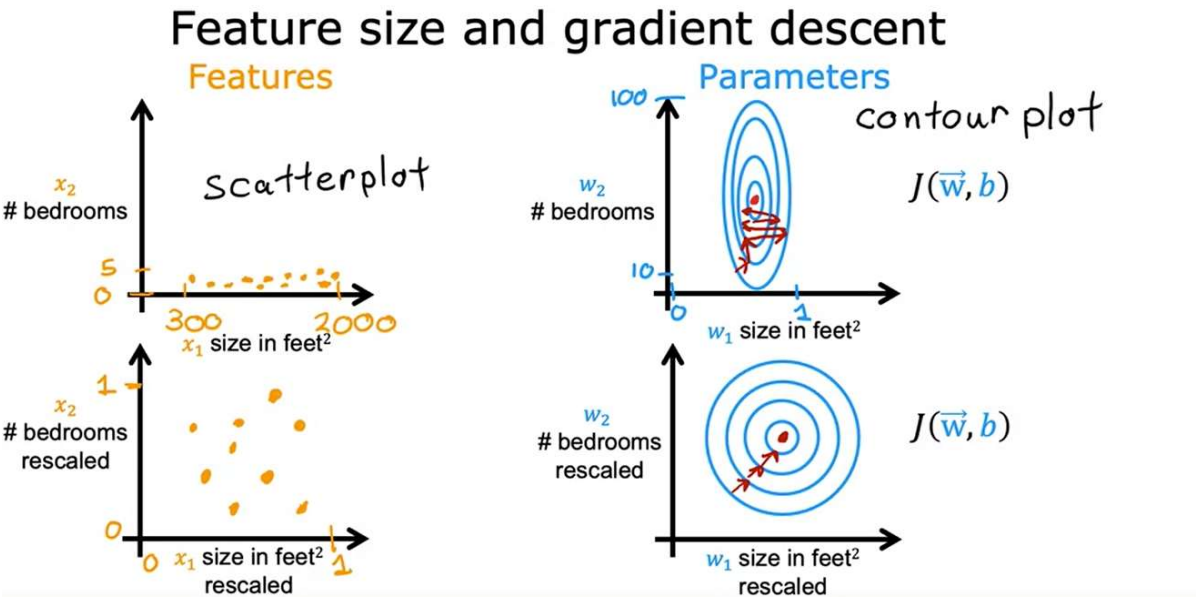


Wenn die möglichen Werte des Merkmals klein sind, beispielsweise die Anzahl der Schlafzimmer, ist ein angemessener Wert für seine Parameter ebenfalls relativ groß, beispielsweise 50. Wie hängt das also mit der Abstiegsbewertung zusammen? Schauen wir uns nun das Streudiagramm der Merkmale an, wobei die Größe in Quadratfuß auf der horizontalen Achse x1 und die Anzahl der Schlafzimmer auf der vertikalen Achse aufgetragen ist. Wenn Sie die Trainingsdaten grafisch darstellen, stellen Sie fest, dass die horizontale Achse im Vergleich zur vertikalen Achse einen viel größeren Maßstab oder einen viel größeren Wertebereich aufweist. Schauen wir uns als Nächstes an, wie die Kostenfunktion in einem Konturdiagramm aussehen könnte. Möglicherweise sehen Sie ein Konturdiagramm, bei dem die horizontale Achse einen viel engeren Bereich hat, beispielsweise zwischen Null und Eins, während die vertikale Achse viel größere Werte annimmt, beispielsweise zwischen 10 und 100.

Die Konturen bilden also Ovale oder Ellipsen und sind kurz auf der einen Seite und länger auf der anderen Seite. Und das liegt daran, dass eine sehr kleine Änderung an w1 einen sehr großen Einfluss auf den geschätzten Preis haben kann, und das hat einen sehr großen Einfluss auf die Kosten J. Denn w1 wird tendenziell mit einer sehr großen Zahl, der Größe und den Quadratfuß multipliziert. Im Gegensatz dazu ist eine viel größere Änderung in w2 erforderlich, um die Vorhersagen stark zu ändern. Und daher verändern kleine Änderungen an w2 die Kostenfunktion nicht annähernd so stark. Wohin führt uns das? Das könnte am Ende passieren, wenn Sie im Dissidenten gut abschneiden und Ihre Trainingsdaten so verwenden würden, wie sie sind.

Da die Konturen so hoch und dünn sind, kann es sein, dass der Abstieg lange hin und her schwankt, bevor er schließlich den Weg zum globalen Minimum findet. In solchen Situationen ist es sinnvoll, die Funktionen zu skalieren. Dies bedeutet, dass Sie eine Transformation Ihrer Trainingsdaten durchführen, sodass x1 beispielsweise jetzt im Bereich von 0 bis 1 und x2 ebenfalls im Bereich von 0 bis 1 liegen könnte. Die Datenpunkte sehen jetzt also eher so aus, und Sie werden möglicherweise feststellen, dass sich der Maßstab des Diagramms verändert hat Die Unterseite ist jetzt ganz anders als die Oberseite. Der

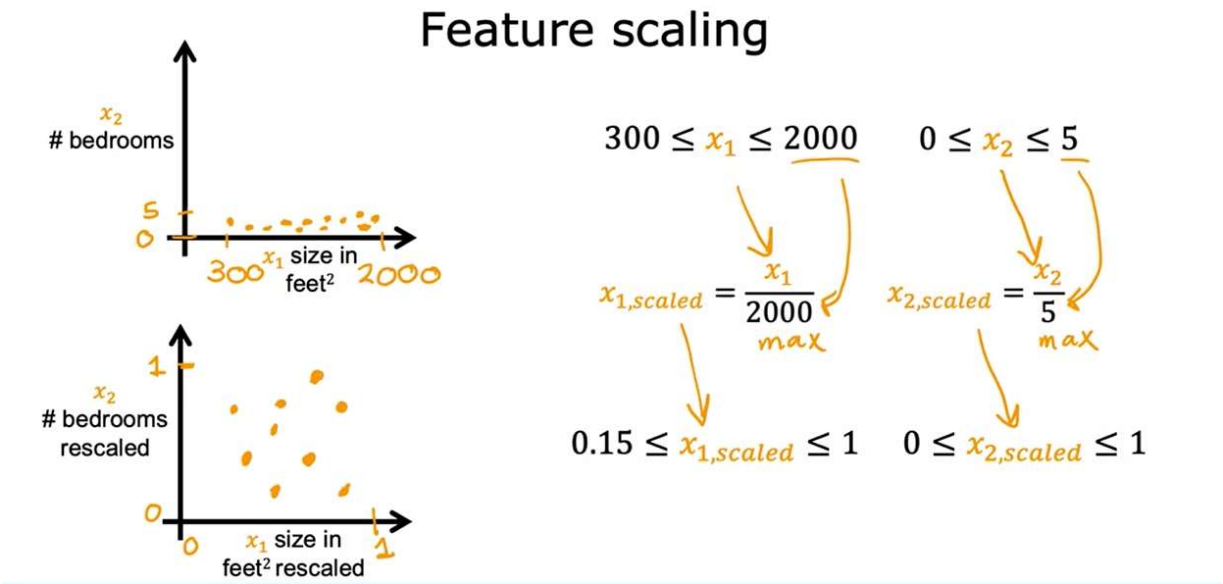
entscheidende Punkt ist, dass die Neuskalierung x1 und x2 jetzt beide vergleichbare Wertebereiche annehmen.



Und wenn Sie einen Gradientenabstieg für eine Kostenfunktion ausführen, um mithilfe dieser transformierten Daten x1 und x2 neu zu skalieren, sehen die Konturen eher so aus, eher wie Kreise und weniger hoch und dünn. Und der Gradientenabstieg kann einen viel direkteren Weg zum globalen Minimum finden. Um es noch einmal zusammenzufassen: Wenn Sie verschiedene Features haben , die sehr unterschiedliche Wertebereiche annehmen, kann es dazu führen, dass der Gradientenabfall langsam verläuft, die verschiedenen Features jedoch neu skaliert werden, sodass sie alle vergleichbare Wertebereiche annehmen. weil Geschwindigkeit, Upgrade und Dissens deutlich. Wie macht man das eigentlich? Schauen wir uns das im nächsten Video an.

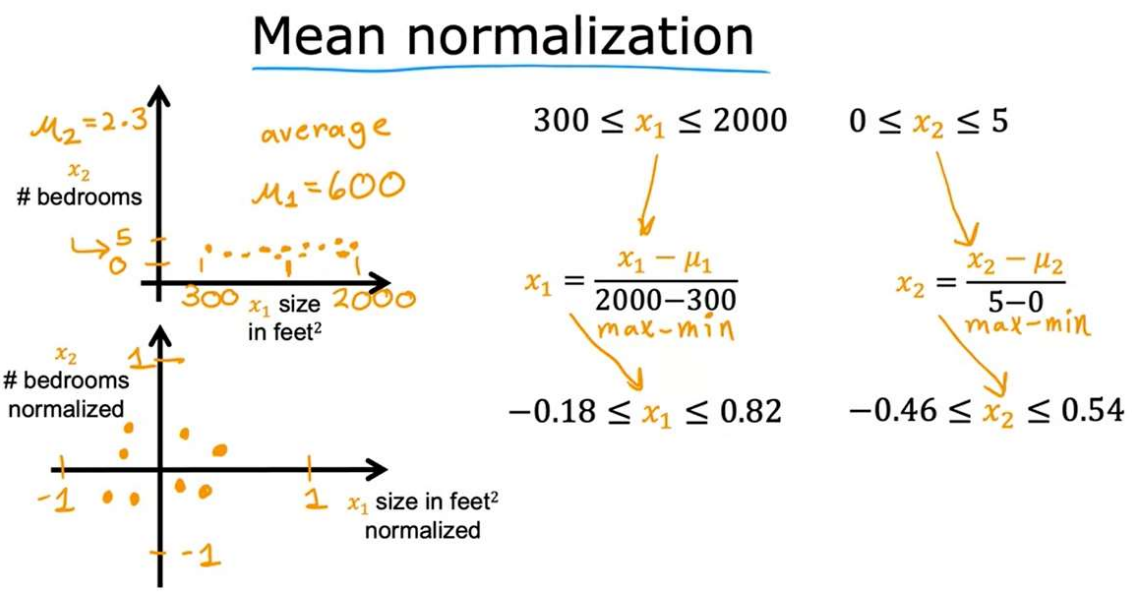
Feature-Skalierung Teil 2

Sehen wir uns an, wie Sie die Feature-Skalierung implementieren können, um Features, die sehr unterschiedliche Wertebereiche annehmen, so zu konzipieren, dass sie untereinander vergleichbare Wertebereiche aufweisen. Wie skaliert man Features eigentlich? Nun, wenn x_1 zwischen 3 und 2.000 liegt, besteht eine Möglichkeit, eine skalierte Version von x_1 zu erhalten, darin, jeden ursprünglichen x1_-Wert zu nehmen und durch 2.000, das Maximum des Bereichs, zu dividieren. Die Skala x_1 reicht von 0,15 bis eins. Da x_2 im Bereich von 0 bis 5 liegt, können Sie auf ähnliche Weise eine skalierte Version von x_2 berechnen, indem Sie jedes ursprüngliche x_2 nehmen und durch fünf dividieren, was wiederum das Maximum darstellt. Die Skala ist also x_2 und reicht nun von 0-1. Wenn Sie den Maßstab auf x_1 und x_2 in einem Diagramm darstellen, könnte das so aussehen.



Neben der Division durch das Maximum können Sie auch eine sogenannte Mittelwertnormalisierung durchführen. Dies sieht so aus, dass Sie mit den ursprünglichen Features beginnen und diese dann neu skalieren, sodass beide zentriert sind um Null. Während sie früher nur Werte größer als Null hatten, haben sie jetzt sowohl negative als auch positive Werte, die normalerweise zwischen minus eins und plus eins liegen können. Um die mittlere Normalisierung von x_1 zu berechnen, ermitteln Sie zunächst den Durchschnitt, auch Mittelwert von x_1 genannt, in Ihrem Trainingssatz, und nennen wir diesen Mittelwert Mu_1, wobei dies das griechische Alphabet Mu ist.

Beispielsweise stellen Sie möglicherweise fest, dass der Durchschnitt von Feature 1, Mu_1, 600 Quadratfuß beträgt. Nehmen wir jedes x_1, subtrahieren den Mittelwert Mu_1 und dividieren dann durch die Differenz 2.000 minus 300, wobei 2.000 das Maximum und 300 das Minimum ist.



Wenn Sie dies tun, erhalten Sie ein normalisiertes x_1 im Bereich von minus 0,18 bis 0,82. Um den normalisierten x_2-Mittelwert zu ermitteln, können Sie in ähnlicher Weise den Durchschnitt von Merkmal 2 berechnen. Beispielsweise kann Mu_2 2,3 betragen. Dann können Sie jedes x_2 nehmen, Mu_2 subtrahieren und durch 5 minus 0 dividieren. Auch hier ist das Maximum 5 minus dem Mittelwert, der 0 ist. Der normalisierte Mittelwert x_2 liegt jetzt im Bereich von minus 0,46 bis 0 54. Wenn Sie die Trainingsdaten mit dem zeichnen bedeuten normalisierte x_1 und x_2, es könnte so aussehen.

Es gibt eine letzte gängige Neuskalierungsmethode namens Z-Score-Normalisierung. Um die Z-Score-Normalisierung zu implementieren, müssen Sie die sogenannte Standardabweichung jedes Merkmals berechnen. Wenn Sie die Standardabweichung nicht kennen, machen Sie sich keine Sorgen, Sie müssen sie für diesen Kurs nicht kennen. Oder wenn Sie von der Normalverteilung oder der glockenförmigen Kurve, manchmal auch Gauß-Verteilung genannt, gehört haben, sehen Sie hier, wie die Standardabweichung für die Normalverteilung aussieht. Aber wenn Sie davon noch nichts gehört haben, brauchen Sie sich darüber auch keine Sorgen zu machen.

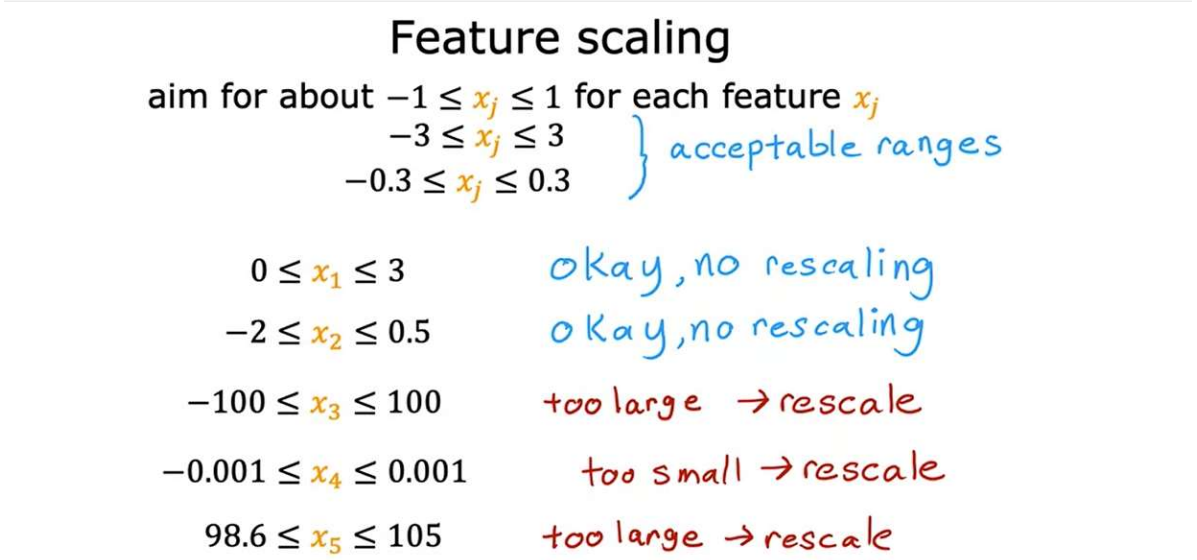
Wenn Sie jedoch wissen, was die Standardabweichung ist, berechnen Sie zum Implementieren einer Z-Score-Normalisierung zunächst den Mittelwert Mu sowie die Standardabweichung, die häufig mit dem griechischen Kleinbuchstaben Sigma für jedes Merkmal angegeben wird. Beispielsweise hat Merkmal 1 möglicherweise eine Standardabweichung von 450 und einen Mittelwert von 600. Um dann den Z-Score zu normalisieren, nehmen Sie x_1, subtrahieren Sie Mu_1 und dividieren Sie dann durch die Standardabweichung, die ich als Sigma 1 bezeichnen werde. Möglicherweise stellen Sie fest, dass der normalisierte Z-Score x_1 jetzt im Bereich von minus 0,67 bis 3,1 liegt.

Wenn Sie in ähnlicher Weise die Standardabweichung des zweiten Merkmals mit 1,4 und den Mittelwert mit 2,3 berechnen, können Sie x_2 minus Mu_2 dividiert durch Sigma_2 berechnen. In diesem Fall könnte der durch x_2 normalisierte Z-Score nun im Bereich von minus 1,6 bis 1,9 liegen. Wenn Sie die Trainingsdaten für die normalisierten x_1 und x_2 in einem Diagramm darstellen, könnte das so aussehen. Als Faustregel gilt: Wenn Sie eine Feature-Skalierung durchführen, sollten Sie darauf abzielen, dass die Features für jedes Feature x im Bereich von etwa minus eins bis etwa plus eins liegen. Aber diese Werte, minus eins und plus eins, können etwas locker sein. Liegen die Merkmale im Bereich von minus drei bis plus drei oder minus 0,3 bis plus 0,3, sind diese völlig in Ordnung.

Wenn Sie ein Merkmal x_1 haben, das am Ende zwischen null und drei liegt, ist das kein Problem. Sie können es bei Bedarf neu skalieren, aber wenn Sie es nicht neu skalieren, sollte es auch einwandfrei funktionieren. Oder wenn Sie eine andere Funktion haben, x_2, deren Werte zwischen -2 und plus 0,5 liegen, ist das wiederum in Ordnung, eine Neuskalierung kann nicht schaden, aber es könnte in Ordnung sein, wenn Sie es auch in Ruhe lassen. Aber wenn ein anderes Merkmal, wie hier x_3, von minus 100 bis plus 100 reicht, dann nimmt dies einen ganz anderen Wertebereich an, sagen wir etwas von etwa minus eins bis plus eins.

Wahrscheinlich ist es besser, diese Funktion x_3 neu zu skalieren, sodass sie von etwas näher an minus eins bis plus eins reicht. Wenn Sie ein Merkmal x_4 haben, das sehr kleine

Werte annimmt, beispielsweise zwischen minus 0,001 und plus 0,001, dann sind diese Werte entsprechend klein. Das bedeutet, dass Sie es möglicherweise auch neu skalieren möchten.

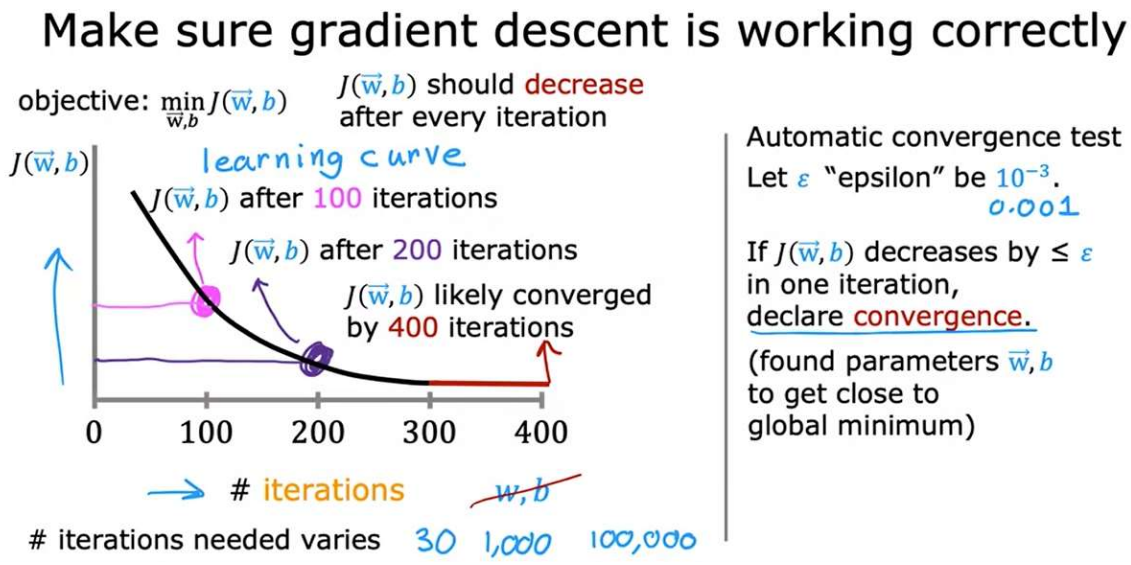


Was passiert schließlich, wenn Ihre Funktion x_5, z. B. die Messung der Temperatur eines Krankenhauspatienten, im Bereich von 98,6 bis 105 Grad Fahrenheit liegt? In diesem Fall liegen diese Werte bei etwa 100, was im Vergleich zu anderen Maßstabsmerkmalen tatsächlich ziemlich groß ist und dazu führt, dass der Gradientenabstieg tatsächlich langsamer verläuft. In diesem Fall wird wahrscheinlich eine Neuskalierung der Funktionen hilfreich sein.

Es schadet fast nie, eine Neuskalierung der Funktionen durchzuführen. Im Zweifelsfall ermutige ich Sie, es einfach auszuführen. Das war's mit der Feature-Skalierung. Mit dieser kleinen Technik gelingt es Ihnen oft, den Gefälleabstieg viel schneller zu bewältigen. Das ist die Skalierung von Funktionen. Wie können Sie beim Ausführen des Gradientenabstiegs mit oder ohne Feature-Skalierung wissen und überprüfen, ob der Gradientenabstieg wirklich funktioniert? Wenn Sie das globale Minimum oder etwas in der Nähe davon finden. Schauen wir uns im nächsten Video an, wie man erkennt, ob der Gradientenabstieg konvergiert, und im darauffolgenden Video wird dies zu einer Diskussion darüber führen, wie man eine gute Lernrate für den Gradientenabstieg wählt.

Überprüfen des Gradientenabstiegs auf Konvergenz

Wie können Sie beim Ausführen des Gradientenabstiegs feststellen, ob er konvergiert? Das heißt, ob es Ihnen hilft, Parameter zu finden, die nahe am globalen Minimum der Kostenfunktion liegen. Indem wir lernen zu erkennen, wie eine gut laufende Implementierung des Gradientenabstiegs aussieht, werden wir in einem späteren Video auch besser in der Lage sein, eine gute Lernrate Alpha auszuwählen. Lass uns einen Blick darauf werfen.



Zur Erinnerung: Hier ist die Gradientenabstiegsregel. Eine der wichtigsten Entscheidungen ist die Wahl der Lernrate Alpha. Folgendes mache ich oft, um sicherzustellen, dass der Gradientenabstieg gut funktioniert. Denken Sie daran, dass die Aufgabe des Gradientenabstiegs darin besteht, die Parameter w und b zu finden, die hoffentlich die Kostenfunktion J minimieren. Was ich oft mache, ist, die Kostenfunktion J, die auf dem Trainingssatz berechnet wird, zu zeichnen, und ich trage den Wert von J ein bei jeder Iteration des Gradientenabstiegs.

Denken Sie daran, dass jede Iteration nach jeder gleichzeitigen Aktualisierung der Parameter w und b bedeutet. In diesem Diagramm ist die horizontale Achse die Anzahl der Iterationen des Gradientenabfalls, die Sie bisher ausgeführt haben. Möglicherweise erhalten Sie eine Kurve, die so aussieht. Beachten Sie, dass die horizontale Achse die Anzahl der Iterationen des Gradientenabstiegs ist und kein Parameter wie w oder b. Dies unterscheidet sich von den vorherigen Diagrammen, die Sie gesehen haben, wo die vertikale Achse die Kosten J und die horizontale Achse ein einzelner Parameter wie w oder b war. Diese Kurve wird auch Lernkurve genannt. Beachten Sie, dass beim maschinellen Lernen einige verschiedene Arten von Lernkurven verwendet werden. Einige dieser Arten werden Sie auch später in diesem Kurs sehen. Konkret bedeutet dies, wenn Sie hier auf diesen Punkt der Kurve schauen, dass Sie, nachdem Sie den Gradientenabstieg für 100 Iterationen ausgeführt haben, d. h. 100 gleichzeitige Aktualisierungen der Parameter, einige gelernte Werte für w und b haben.

Wenn Sie die Kosten J, w, b für die Werte von w und b berechnen, die Sie nach 100 Iterationen erhalten haben, erhalten Sie diesen Wert für die Kosten J. Das ist dieser Punkt auf der vertikalen Achse. Dieser Punkt entspricht hier dem Wert von J für die Parameter, den Sie nach 200 Iterationen des Gradientenabstiegs erhalten haben. Wenn Sie sich dieses Diagramm ansehen, können Sie sehen, wie sich Ihre Kosten J nach jeder Iteration des Gradientenabfalls ändern. Wenn der Gradientenabstieg ordnungsgemäß funktioniert, sollten die Kosten J nach jeder einzelnen Iteration sinken.

Wenn J nach einer Iteration jemals ansteigt, bedeutet das entweder, dass Alpha schlecht gewählt wurde, was normalerweise bedeutet, dass Alpha zu groß ist, oder dass ein Fehler im

Code vorliegen könnte. Eine weitere nützliche Sache, die Ihnen dieser Teil sagen kann, ist, dass, wenn Sie sich diese Kurve ansehen, bis Sie etwa 300 Iterationen erreichen, die Kosten J sich einpendeln und nicht mehr stark sinken. Nach 400 Iterationen scheint sich die Kurve abgeflacht zu haben.

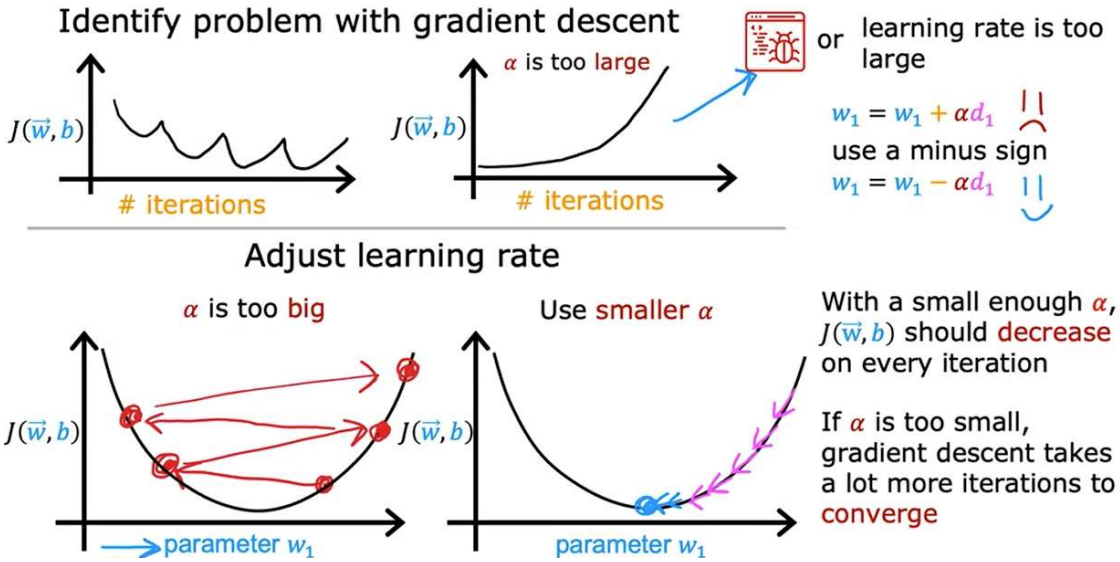
Dies bedeutet, dass der Gradientenabfall mehr oder weniger konvergiert ist, da die Kurve nicht mehr abnimmt. Wenn Sie sich diese Lernkurve ansehen, können Sie versuchen zu erkennen, ob der Gradientenabfall konvergiert oder nicht. Übrigens kann die Anzahl der Iterationen, die der Gradientenabstieg für eine Konvertierung benötigt, zwischen verschiedenen Anwendungen stark variieren. In einer Anwendung kann es bereits nach 30 Iterationen zu einer Konvergenz kommen. Für eine andere Anwendung könnten 1.000 oder 100.000 Iterationen erforderlich sein.

Es erweist sich als sehr schwierig, im Voraus zu sagen, wie viele Iterationen der Gradientenabstieg zur Konvergenz benötigt, weshalb Sie ein Diagramm wie dieses, eine Lernkurve, erstellen können. Versuchen Sie herauszufinden, wann Sie mit dem Training Ihres speziellen Modells beginnen können. Eine weitere Möglichkeit zu entscheiden, wann das Training Ihres Modells abgeschlossen ist, ist ein automatischer Konvergenztest. Hier ist das griechische Alphabet Epsilon. Nehmen wir an, Epsilon sei eine Variable, die eine kleine Zahl darstellt, beispielsweise 0,001 oder 10^-3. Wenn die Kosten J bei einer Iteration um weniger als diese Zahl Epsilon sinken, dann befinden Sie sich wahrscheinlich in diesem abgeflachten Teil der Kurve, die Sie links sehen, und Sie können Konvergenz erklären. Denken Sie daran, Konvergenz, hoffentlich für den Fall, dass Sie die Parameter w und b gefunden haben, die nahe am minimal möglichen Wert von J liegen.

Normalerweise finde ich, dass die Wahl des richtigen Schwellenwerts Epsilon ziemlich schwierig ist. Eigentlich neige ich dazu, mir Diagramme wie dieses links anzuschauen, anstatt mich auf automatische Konvergenztests zu verlassen. Ein Blick auf die solide Abbildung kann Ihnen Aufschluss darüber geben, ob der Gefälleabstieg möglicherweise auch nicht ordnungsgemäß funktioniert. Sie haben nun gesehen, wie die Lernkurve aussehen sollte, wenn der Gefälleabstieg gut verläuft. Nehmen wir diese Erkenntnisse und schauen wir uns im nächsten Video an, wie man eine geeignete Lernrate wählt.

Wahl der Lernrate

Ihr Lernalgorithmus läuft viel besser, wenn Sie die Lernrate richtig wählen. Wenn es zu klein ist, läuft es sehr langsam und wenn es zu groß ist, konvergiert es möglicherweise nicht einmal. Sehen wir uns an, wie Sie eine gute Lernrate für Ihr Modell auswählen können.

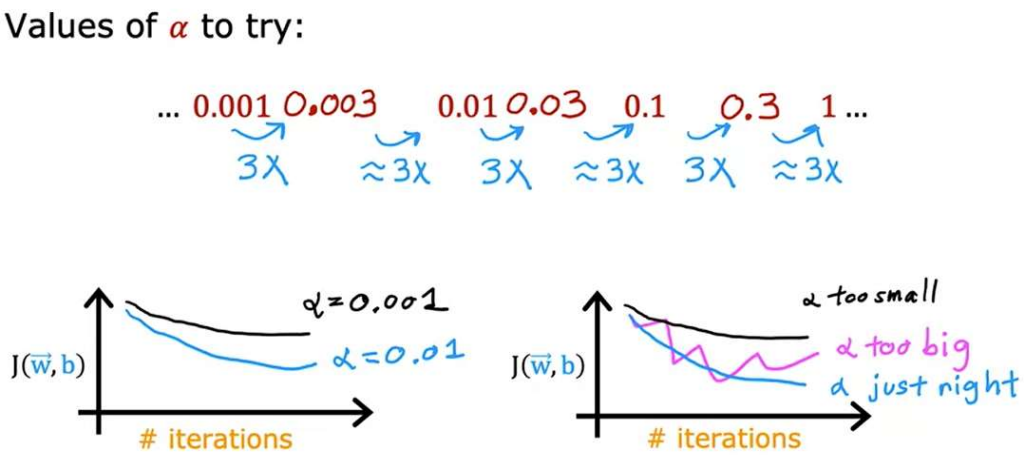


Konkret: Wenn Sie die Kosten für mehrere Iterationen grafisch darstellen und feststellen, dass die Kosten manchmal steigen und manchmal sinken, sollten Sie dies als klares Zeichen dafür werten, dass der Gradientenabstieg nicht richtig funktioniert. Dies könnte bedeuten, dass ein Fehler im Code vorliegt. Oder manchmal könnte es bedeuten, dass Ihre Lernrate zu hoch ist. Hier ist eine Veranschaulichung dessen, was passieren könnte. Hier ist die vertikale Achse eine Kostenfunktion J und die horizontale Achse stellt einen Parameter wie etwa w_1 dar.

Wenn die Lernrate zu groß ist, kann es sein, dass Ihr Aktualisierungsschritt, wenn Sie hier beginnen, das Minimum überschreitet und hier und dort endet Der nächste Update-Schritt hier, Ihr Gewinn übersteigt, sodass Sie hier landen und so weiter. Aus diesem Grund können die Kosten manchmal steigen, anstatt zu sinken. Um dies zu beheben, können Sie eine kleinere Lernrate verwenden. Dann können Ihre Aktualisierungen hier beginnen und immer weiter abnehmen, und wir werden hoffentlich kontinuierlich abnehmen, bis sie das globale Minimum erreichen. Manchmal stellen Sie möglicherweise fest, dass die Kosten nach jeder Iteration kontinuierlich steigen, wie in dieser Kurve hier. Dies ist wahrscheinlich auch auf eine zu große Lernrate zurückzuführen und könnte durch die Wahl einer kleineren Lernrate behoben werden.

Solche Lernraten könnten aber auch ein Zeichen für einen möglicherweise fehlerhaften Code sein. Wenn ich beispielsweise meinen Code so schreibe, dass w_1 als w_1 plus Alpha mal diesem Ableitungsterm aktualisiert wird, könnte dies dazu führen, dass die Kosten bei jeder Iteration stetig steigen. Dies liegt daran, dass sich Ihre Kosten J durch den Ableitungsterm weiter vom globalen Minimum entfernen, anstatt sich ihm anzunähern. Denken Sie also daran, dass Sie ein Minuszeichen verwenden möchten, daher sollte der Code w_1 aktualisiert werden, indem w_1 minus Alpha mal der Ableitungsterm aktualisiert wird. Ein Debugging-Tipp für eine korrekte Implementierung des Gradientenabstiegs ist, dass bei einer ausreichend kleinen Lernrate die Kostenfunktion bei jeder einzelnen Iteration sinken sollte. Wenn also der Gradientenabstieg nicht funktioniert, mache ich das oft und ich hoffe, dass Sie diesen Tipp auch nützlich finden.

Eine Sache, die ich oft mache, ist, einfach Alpha auf eine sehr kleine Zahl zu setzen und zu prüfen, ob dadurch die Kosten sinken bei jeder Iteration. Wenn J nicht bei jeder einzelnen Iteration abnimmt, sondern manchmal zunimmt, selbst wenn Alpha auf eine sehr kleine Zahl eingestellt ist, bedeutet das normalerweise, dass irgendwo im Code ein Fehler vorliegt. Beachten Sie, dass die Einstellung von Alpha auf einen wirklich kleinen Wert hier als Debugging-Schritt gedacht ist und ein sehr kleiner Wert von Alpha nicht die effizienteste Wahl für das tatsächliche Training Ihres Lernalgorithmus ist.



Ein wichtiger Kompromiss besteht darin, dass bei zu geringer Lernrate Gradientenabstiege viele Iterationen erfordern können, um zu konvergieren. Wenn ich also einen Gradientenabstieg durchführe, probiere ich normalerweise einen Wertebereich für die Lernrate α aus. Ich kann damit beginnen, eine Lernrate von 0,001 auszuprobieren, und ich kann auch eine zehnmal so große Lernrate versuchen, beispielsweise 0,01 und 0,1 und so weiter. Für jede Wahl von Alpha können Sie einen Gradientenabstieg nur für eine Handvoll Iterationen ausführen und die Kostenfunktion J als Funktion der Anzahl der Iterationen darstellen.

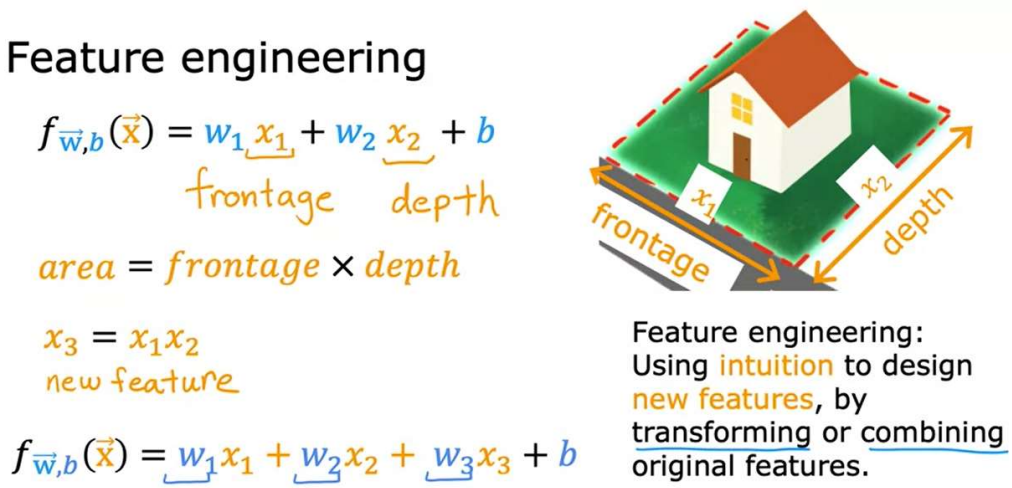
Nachdem Sie einige verschiedene Werte ausprobiert haben, können Sie dann den Wert von Alpha auswählen, der Ihnen erscheint Verringern Sie die Lernrate schnell, aber auch kontinuierlich. Tatsächlich probiere ich eine Reihe solcher Werte aus. Nachdem ich 0,001 ausprobiert habe, erhöhe ich die Lernrate um das Dreifache auf 0,003. Danach versuche ich es mit 0,01, was wiederum etwa dreimal so groß ist wie 0,003. Es handelt sich also grob um das Ausprobieren von Gradientenabstiegen, wobei jeder Wert von Alpha ungefähr dreimal so groß ist wie der vorherige Wert. Was ich tun werde, ist, eine Reihe von Werten auszuprobieren, bis ich herausgefunden habe, dass der Wert zu klein ist, und dann auch sicherzustellen, dass ich einen Wert gefunden habe, der zu groß ist. Ich werde langsam versuchen, die größtmögliche Lernrate auszuwählen, oder einfach etwas, das etwas kleiner ist als der größte vernünftige Wert, den ich gefunden habe. Wenn ich das mache, erhalte ich normalerweise eine gute Lernrate für mein Modell. Ich hoffe, dass diese Technik auch für Sie nützlich sein wird, um eine gute Lernrate für Ihre Implementierung des Gradientenabstiegs auszuwählen.

In der kommenden optionalen Übung können Sie sich auch ansehen, wie die Feature-Skalierung im Code erfolgt, und sehen, wie unterschiedliche Auswahlmöglichkeiten der Lernrate α zu einem besseren oder schlechteren Training Ihres Modells führen können. Ich wünsche Ihnen viel Spaß beim Spielen mit dem Wert von Alpha und beim Betrachten der Ergebnisse verschiedener Alpha-Entscheidungen. Bitte werfen Sie einen Blick darauf und führen Sie den Code im optionalen Labor aus, um einen tieferen Einblick in die Feature-Skalierung sowie die Lernrate α zu erhalten.

Die Auswahl der Lernraten ist ein wichtiger Teil des Trainings vieler Lernalgorithmen und ich hoffe, dass Ihnen dieses Video einen Einblick in die verschiedenen Auswahlmöglichkeiten gibt und Ihnen zeigt, wie Sie einen guten Wert für Alpha auswählen. Nun gibt es noch ein paar weitere Ideen, mit denen Sie die multiple lineare Regression viel leistungsfähiger machen können. Dabei handelt es sich um die Auswahl benutzerdefinierter Funktionen, mit denen Sie auch Kurven und nicht nur eine gerade Linie an Ihre Daten anpassen können. Schauen wir uns das im nächsten Video an.

Feature-Engineering

Die Auswahl der Features kann einen großen Einfluss auf die Leistung Ihres Lernalgorithmus haben. Tatsächlich ist die Auswahl oder Eingabe der richtigen Funktionen für viele praktische Anwendungen ein entscheidender Schritt für die gute Funktion des Algorithmus. Sehen wir uns in diesem Video an, wie Sie die am besten geeigneten Funktionen für Ihren Lernalgorithmus auswählen oder entwickeln können. Werfen wir einen Blick auf das Feature-Engineering, indem wir uns das Beispiel der Vorhersage des Hauspreises noch einmal ansehen. Angenommen, Sie haben zwei Funktionen für jedes Haus. X_1 ist die Breite der Grundstücksgröße des Grundstücks, auf dem das Haus bebaut ist. Dies wird im realen Zustand auch Grundstücksfront genannt, und das zweite Merkmal, x_2, ist die Tiefe der Grundstücksgröße, nehmen wir an, das rechteckige Grundstück, auf dem das Haus gebaut wurde. Angesichts dieser beiden Merkmale x_1 und x_2 könnten Sie ein Modell wie dieses erstellen, bei dem f(x) w_1x_1 plus w_2x_2 plus b ist, wobei x_1 die Front oder Breite und x_2 die Tiefe ist. Dieses Modell könnte gut funktionieren. Aber hier ist eine weitere Option, wie Sie diese Funktionen im Modell auf andere Weise nutzen können, die noch effektiver sein könnte.

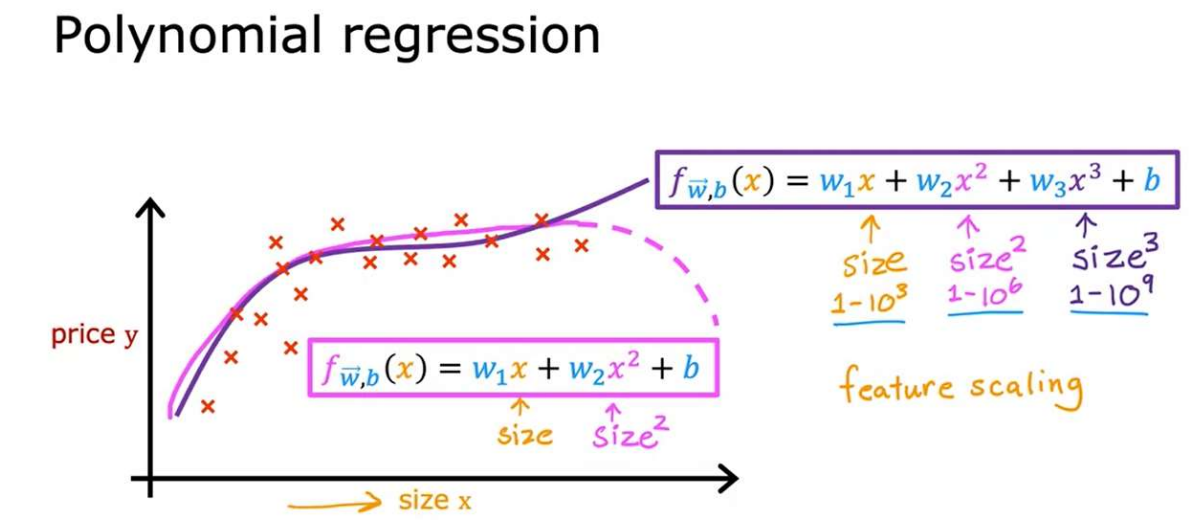


Möglicherweise ist Ihnen aufgefallen, dass die Fläche des Grundstücks als Frontfläche oder Breite multipliziert mit der Tiefe berechnet werden kann. Möglicherweise haben Sie den Eindruck, dass die Fläche des Grundstücks einen größeren Einfluss auf den Preis hat als die Lage und die Tiefe als separate Merkmale. Sie könnten ein neues Feature, x_3, als x_1 mal x_2 definieren. Dieses neue Merkmal x_3 entspricht der Fläche des Grundstücks. Mit dieser Funktion können Sie dann ein Modell f_w haben, wobei b von x gleich w_1x_1 plus w_2x_2 plus w_3x_3 plus b ist, sodass das Modell nun die Parameter w_1, w_2 und w_3 auswählen kann, je nachdem, ob die Daten die Front oder die Tiefe zeigen oder die Fläche x_3 des Grundstücks erweist sich als das wichtigste Kriterium für die Vorhersage des Hauspreises.

Was wir gerade gemacht haben, das Erstellen eines neuen Features, ist ein Beispiel für das sogenannte Feature Engineering, bei dem Sie Ihr Wissen oder Ihre Intuition über das Problem nutzen können, um neue Features zu entwerfen, in der Regel durch Transformation oder Kombination der ursprünglichen Features des Problems, um es zu erstellen Dies erleichtert es dem Lernalgorithmus, genaue Vorhersagen zu treffen. Abhängig davon, welche Einblicke Sie in die Anwendung haben, können Sie möglicherweise ein viel besseres Modell erhalten, anstatt nur die Funktionen zu übernehmen, mit denen Sie zufällig begonnen haben, und manchmal neue Funktionen zu definieren. Das ist Feature Engineering. Es stellt sich heraus, dass dies eine Variante des Feature-Engineerings ist, die es Ihnen ermöglicht, nicht nur gerade Linien, sondern auch Kurven und nichtlineare Funktionen an Ihre Daten anzupassen. Schauen wir uns im nächsten Video an, wie Sie das machen können.

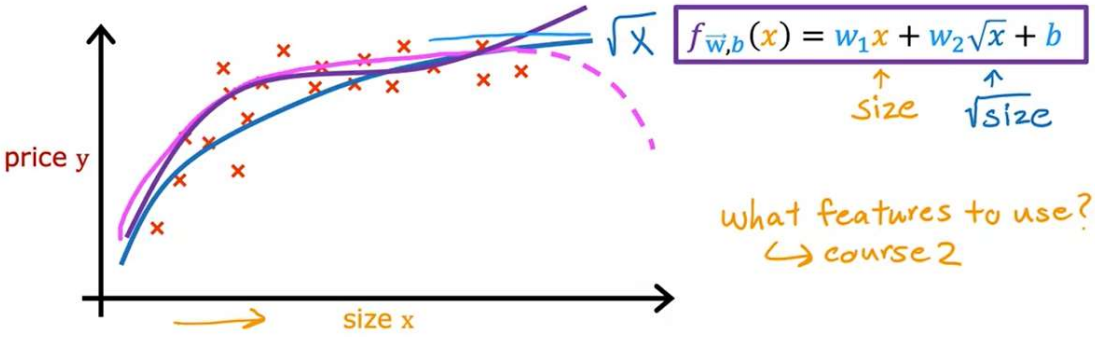
Polynomische Regression

Bisher haben wir nur gerade Linien an unsere Daten angepasst. Nehmen wir die Ideen der multiplen linearen Regression und des Feature-Engineerings, um einen neuen Algorithmus namens Polynomregression zu entwickeln, mit dem Sie Kurven und nichtlineare Funktionen an Ihre Daten anpassen können. Nehmen wir an, Sie haben einen Wohnungsdatensatz, der so aussieht, wobei Merkmal x die Größe in Quadratfuß ist. Es sieht nicht so aus, als ob eine gerade Linie sehr gut zu diesem Datensatz passt.



Vielleicht möchten Sie eine Kurve, vielleicht eine quadratische Funktion, an die Daten wie diese anpassen, die eine Größe x und auch x im Quadrat enthält, also die Größe hoch zwei. Vielleicht erhalten Sie dadurch eine bessere Übereinstimmung mit den Daten. Aber dann kommen Sie vielleicht zu dem Schluss, dass Ihr quadratisches Modell keinen Sinn ergibt, weil eine quadratische Funktion irgendwann wieder fehlschlägt. Nun, wir würden nicht wirklich erwarten, dass die Immobilienpreise sinken, wenn die Größe zunimmt.

Große Häuser scheinen normalerweise mehr zu kosten. Dann können Sie eine kubische Funktion wählen, bei der wir jetzt nicht nur x quadriert, sondern x kubisch haben. Vielleicht erzeugt dieses Modell hier diese Kurve, die etwas besser zu den Daten passt, da die Größe schließlich mit zunehmender Größe wieder ansteigt. Dies sind beides Beispiele einer polynomialen Regression, da Sie Ihr optionales Merkmal x genommen und es auf die Potenz von zwei oder drei oder einer anderen Potenz erhöht haben. Bei der kubischen Funktion ist das erste Merkmal die Größe, das zweite Merkmal die quadrierte Größe und das dritte Merkmal die kubische Größe. Ich möchte nur noch auf eine Sache hinweisen: Wenn Sie Features erstellen, die solche Kräfte wie das Quadrat der ursprünglichen Features haben, wird die Feature-Skalierung immer wichtiger. Wenn die Größe des Hauses beispielsweise zwischen 1 und 1.000 Quadratfuß liegt, dann reicht das zweite Merkmal, das die Größe zum Quadrat angibt, von eins bis zu einer Million, und das dritte Merkmal, das die Größe zum Quadrat misst, reicht von eins bis a Milliarde.



Diese beiden Merkmale, x im Quadrat und x kubisch, nehmen im Vergleich zum ursprünglichen Merkmal x sehr unterschiedliche Wertebereiche an. Wenn Sie den Gradientenabstieg verwenden, ist es wichtig, die Feature-Skalierung anzuwenden, um Ihre Features in vergleichbare Wertebereiche zu bringen. Abschließend noch ein letztes Beispiel dafür, wie vielfältig die Auswahl an Funktionen ist, die Sie nutzen können. Eine weitere sinnvolle Alternative zum Quadrat der Größe und zur Kubikgröße besteht darin, beispielsweise die Quadratwurzel von x zu verwenden. Ihr Modell könnte wie folgt aussehen: w_1 mal x plus w_2 mal die Quadratwurzel von x plus b.

Die Quadratwurzelfunktion sieht so aus und wird mit zunehmendem x etwas weniger steil, aber sie flacht nie vollständig ab und fällt ganz bestimmt nie wieder ab. Dies wäre eine weitere Auswahl an Funktionen, die auch für diesen Datensatz gut funktionieren könnten. Sie fragen sich vielleicht: Wie entscheide ich, welche Funktionen ich verwenden möchte? Später im zweiten Kurs dieser Spezialisierung erfahren Sie, wie Sie verschiedene Funktionen und Modelle auswählen können, die diese Funktionen enthalten oder nicht, und Sie verfügen über einen Prozess zum Messen der Leistung dieser verschiedenen Modelle, um Ihnen bei der Entscheidung zu helfen, welche Funktionen Sie verwenden möchten einschließen oder nicht einschließen.

Im Moment möchte ich Sie nur darauf hinweisen, dass Sie die Wahl haben, welche Funktionen Sie nutzen möchten. Durch den Einsatz von Feature Engineering und Polynomfunktionen können Sie möglicherweise ein viel besseres Modell für Ihre Daten erhalten. In der optionalen Übung, die diesem Video folgt, sehen Sie Code, der eine Polynomregression unter Verwendung von Funktionen wie x, x-Quadrat und x-Würfel implementiert. Bitte werfen Sie einen Blick darauf, führen Sie den Code aus und sehen Sie, wie er funktioniert.

Danach gibt es noch eine weitere optionale Übung, die zeigt, wie man ein beliebtes Open-Source-Toolkit verwendet, das lineare Regression implementiert. Scikit-learn ist eine sehr weit verbreitete Open-Source-Bibliothek für maschinelles Lernen, die von vielen Praktikern in vielen der führenden KI-, Internet- und maschinellen Lernunternehmen der Welt verwendet wird. Wenn Sie jetzt oder in Zukunft maschinelles Lernen in Ihrem Beruf

einsetzen, besteht eine sehr gute Chance, dass Sie Tools wie Scikit-learn verwenden, um Ihre Modelle zu trainieren. Wenn Sie dieses optionale Labor durcharbeiten, haben Sie nicht nur die Möglichkeit, die lineare Regression besser zu verstehen, sondern auch zu sehen, wie dies mithilfe einer Bibliothek wie Scikit-learn in nur wenigen Codezeilen durchgeführt werden kann.

Damit Sie ein solides Verständnis dieser Algorithmen haben und sie anwenden können, ist es meiner Meinung nach wichtig, dass Sie wissen, wie Sie die lineare Regression selbst implementieren und nicht einfach eine Scikit-Learn-Funktion aufrufen, die eine Blackbox darstellt . Aber Scikit-learn spielt auch eine wichtige Rolle bei der Art und Weise, wie maschinelles Lernen heute in der Praxis durchgeführt wird. Wir sind fast am Ende dieser Woche angelangt.

Tools

You will utilize functions from scikit-learn as well as matplotlib and NumPy.

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import SGDRegressor
from sklearn.preprocessing import StandardScaler
from lab_utils_multi import  load_house_data
from lab_utils_common import dlc
np.set_printoptions(precision=2)
plt.style.use('./deeplearning.mplstyle')
```

Gradient Descent

Scikit-learn has a gradient descent regression model sklearn.linear_model.SGDRegressor. Like your previous implementation of gradient descent, this model performs best with normalized inputs. sklearn.preprocessing.StandardScaler will perform z-score normalization as in a previous lab. Here it is referred to as 'standard score'.

Load the data set

```
X_train, y_train = load_house_data()
X_features = ['size(sqft)','bedrooms','floors','age']
```

Scale/normalize the training data

```
scaler = StandardScaler()
X_norm = scaler.fit_transform(X_train)
print(f"Peak to Peak range by column in Raw        X:{np.ptp(X_train,axis=0)}")
print(f"Peak to Peak range by column in Normalized X:{np.ptp(X_norm,axis=0)}")
```

```
Peak to Peak range by column in Raw        X:[2.41e+03 4.00e+00 1.00e+00 9.50e+01]
Peak to Peak range by column in Normalized X:[5.85 6.14 2.06 3.69]
```

Create and fit the regression model

```
sgdr = SGDRegressor(max_iter=1000)
sgdr.fit(X_norm, y_train)
print(sgdr)
print(f"number of iterations completed: {sgdr.n_iter_}, number of weight updates: {sgdr.t_}")
```

```
SGDRegressor()
number of iterations completed: 127, number of weight updates: 12574.0
```

View parameters

Note, the parameters are associated with the *normalized* input data. The fit parameters are very close to those found in the previous lab with this data.

```
b_norm = sgdr.intercept_
w_norm = sgdr.coef_
print(f"model parameters:                   w: {w_norm}, b:{b_norm}")
print( "model parameters from previous lab: w: [110.56 -21.27 -32.71 -37.97], b: 363.16")
```

```
model parameters:                   w: [110.19 -21.08 -32.49 -38.03], b:[363.16]
model parameters from previous lab: w: [110.56 -21.27 -32.71 -37.97], b: 363.16
```

Make predictions

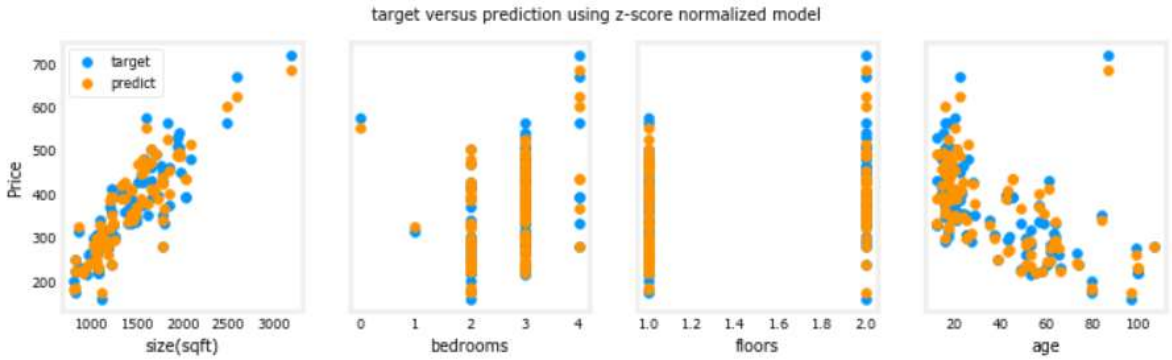
Predict the targets of the training data. Use both the `predict` routine and compute using w and b.

```
# make a prediction using sgdr.predict()
y_pred_sgd = sgdr.predict(X_norm)
# make a prediction using w,b.
y_pred = np.dot(X_norm, w_norm) + b_norm
```

Plot Results

Let's plot the predictions versus the target values.

```
# plot predictions and targets vs original features
fig,ax=plt.subplots(1,4,figsize=(12,3),sharey=True)
for i in range(len(ax)):
    ax[i].scatter(X_train[:,i],y_train, label = 'target')
    ax[i].set_xlabel(X_features[i])
    ax[i].scatter(X_train[:,i],y_pred,color=dlc["dlorange"], label = 'predict')
ax[0].set_ylabel("Price"); ax[0].legend();
fig.suptitle("target versus prediction using z-score normalized model")
plt.show()
```



Diese Woche lernen Sie die andere Art des überwachten Lernens kennen: die Klassifizierung. Sie erfahren, wie Sie Kategorien mithilfe des logistischen Regressionsmodells vorhersagen. Sie lernen das Problem der Überanpassung kennen und erfahren, wie Sie dieses Problem mit einer Methode namens Regularisierung lösen können. Am Ende dieser Woche können Sie die Implementierung der logistischen Regression mit Regularisierung üben!

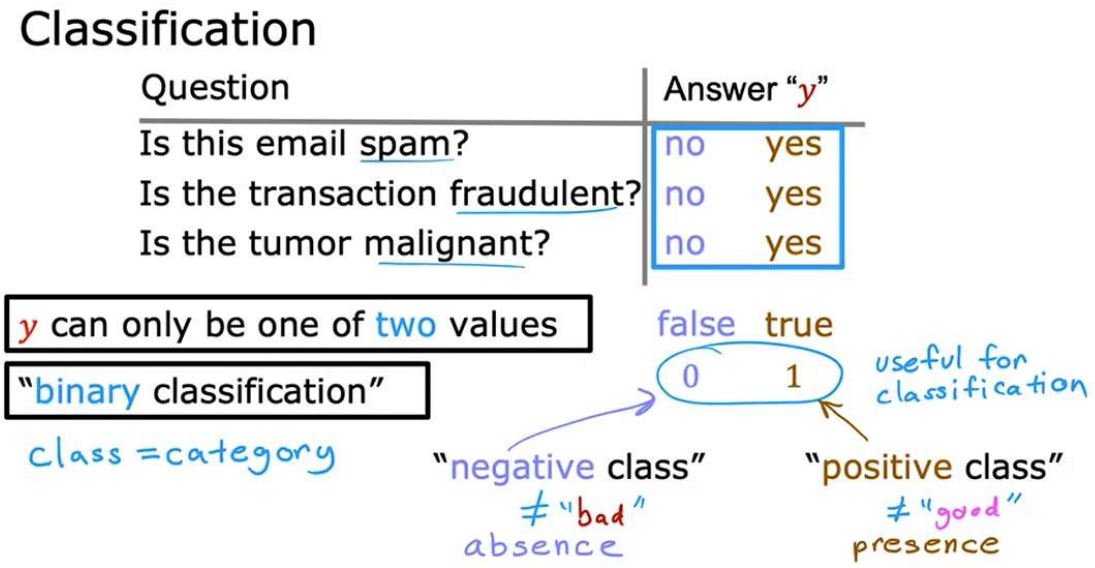
Lernziele

- Verwenden Sie die logistische Regression für die binäre Klassifizierung
- Implementieren Sie eine logistische Regression für die binäre Klassifizierung
- Beheben Sie Überanpassung mithilfe von Regularisierung, um die Modelllleistung zu verbessern
- **Motivation**

Diese Woche lernen Sie etwas über die Klassifizierung, bei der Ihre Ausgabevariable y nur einen von wenigen möglichen Werten annehmen kann, statt einer beliebigen Zahl in einem unendlichen Zahlenbereich. Es stellt sich heraus, dass die lineare Regression kein guter Algorithmus für Klassifizierungsprobleme ist. Werfen wir einen Blick darauf, warum das so ist, und das führt uns zu einem anderen Algorithmus namens logistischer Regression. Dies ist heute einer der beliebtesten und am weitesten verbreiteten Lernalgorithmen.

Hier sind einige Beispiele für Klassifizierungsprobleme. Erinnern Sie sich an das Beispiel des Versuchs herauszufinden, ob es sich bei einer E-Mail um Spam handelt. Die Antwort, die Sie ausgeben möchten, wird also entweder „Nein“ oder „Ja“ sein. Ein weiteres Beispiel wäre die Feststellung, ob eine Online-Finanztransaktion betrügerisch ist. Ich habe einmal an der Bekämpfung von Online-Finanzbetrug gearbeitet und es war seltsam aufregend. Weil ich wusste, dass da draußen Kräfte waren, die versuchten, Geld zu stehlen, und die Aufgabe meines Teams darin bestand, sie aufzuhalten. Das Problem ist also eine Finanztransaktion.

Kann Ihr Lernalgorithmus herausfinden, ob es sich bei dieser Transaktion um einen Betrug handelt, beispielsweise um die gestohlene Kreditkarte? Ein weiteres Beispiel, das wir zuvor angesprochen haben, war der Versuch, einen Tumor als bösartig und nicht als bösartig zu klassifizieren. Bei jedem dieser Probleme kann die Variable, die Sie vorhersagen möchten, nur einen von zwei möglichen Werten annehmen. Nein oder Ja. Diese Art von Klassifizierungsproblem, bei dem es nur zwei mögliche Ausgaben gibt, wird als binäre Klassifizierung bezeichnet. Wobei sich das Wort „binär“ darauf bezieht, dass es nur zwei mögliche Klassen oder zwei mögliche Kategorien gibt.



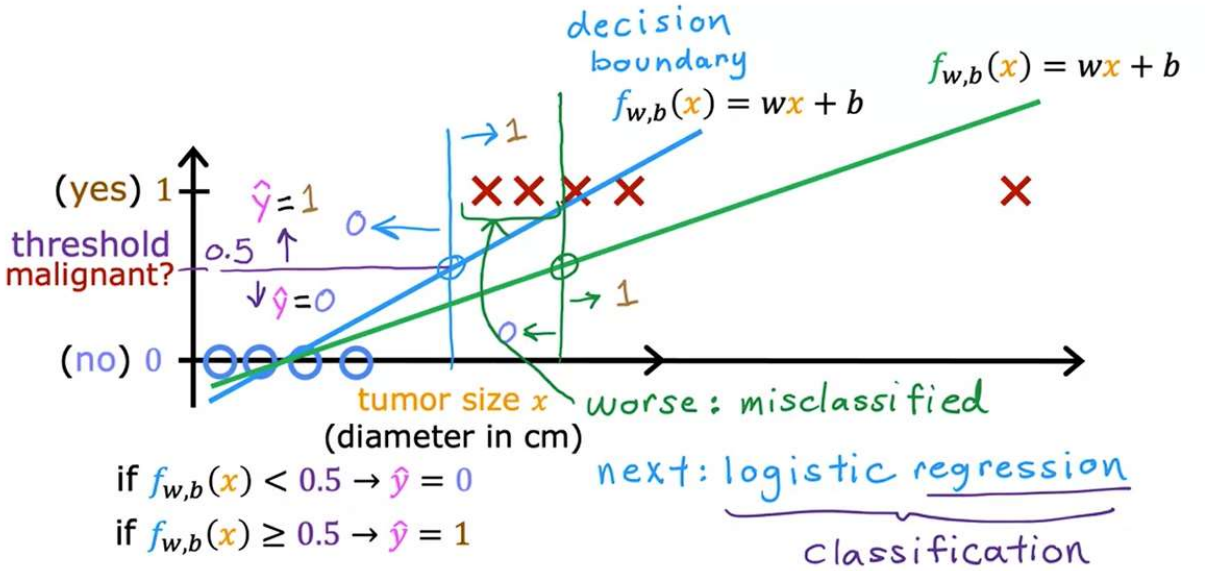
Bei diesen Problemen werde ich die Begriffe Klasse und Kategorie relativ austauschbar verwenden. Sie meinen im Grunde das Gleiche. Konventionell können wir auf einige gängige Arten auf diese beiden Klassen oder Kategorien verweisen. Wir bezeichnen Klauseln oft als „nein“ oder „ja“ oder manchmal als gleichwertig „falsch“ oder „wahr“ oder verwenden sehr häufig die Zahlen Null oder Eins. In Anlehnung an die in der Informatik übliche Konvention, dass Null für Stürze und Eins für Wahr steht.

Normalerweise verwende ich die Zahlen Null und Eins, um die Antwort y darzustellen. Denn das passt am besten zu den Arten von Lernalgorithmen, die wir implementieren wollen. Aber wenn wir darüber reden, sagen wir oft nein oder ja oder auch falsch oder wahr. Eine der häufig verwendeten Technologien besteht darin, die Klasse „Falsch“ oder „Null“ aufzurufen. Die negative Klasse und die wahre oder die eine Klasse, die positive Klasse. Beispielsweise kann bei der Spam-Klassifizierung eine E-Mail, die kein Spam ist, als Negativbeispiel bezeichnet werden. Weil die Ausgabe auf die Frage ein Spam ist. Die Ausgabe ist nein oder null. Im Gegensatz dazu könnte eine E-Mail mit Spam als positives Trainingsbeispiel bezeichnet werden. Denn die Antwort auf die Frage, ob es sich um Spam handelt, ist „Ja“ oder „Wahr“ oder eine klare, negative und positive Antwort.

Bedeutet nicht unbedingt Böse gegen Gut oder Böse gegen Gut. Es ist nur so, dass negative und positive Beispiele verwendet werden, um die Konzepte der Abwesenheit oder Null oder falsch gegenüber der Anwesenheit oder wahr oder einem von etwas zu vermitteln, nach dem Sie suchen könnten. Zum Beispiel das Fehlen oder Vorhandensein der Spam-Krankheit oder der Spam-Eigenschaft einer E-Mail oder das Fehlen einer Ausbreitungsaktivität oder das Fehlen einer Bösartigkeit des Tumors. Zwischen Nicht-Spam- und Spam-E-Mails. Welches Sie falsch oder Null nennen und welches Sie wahr oder Eins nennen, ist ein wenig willkürlich. Oftmals könnte beides funktionieren. Ein anderer Ingenieur könnte es also tatsächlich austauschen und die positive Klasse B erhalten. Das Vorhandensein einer guten E-Mail oder die möglichen Ursachen sind das Vorhandensein einer echten Finanztransaktion oder eines gesunden Patienten.

Wie erstellt man also einen Klassifizierungsalgorithmus? Hier ist das Beispiel eines Trainingssatzes zur Klassifizierung, ob der Tumor bösartig ist. Eine Klasse Eins, eine positive Klasse, eine Ja-Klasse oder eine harmlose Klasse, eine Klasse Null oder eine negative Klasse. Ich habe sowohl die Tumorgröße auf der horizontalen Achse als auch die Beschriftung Y auf der vertikalen Achse aufgetragen. Übrigens, in der ersten Woche, als wir zum ersten Mal über die Klassifizierung gesprochen haben. So haben wir es uns zuvor auf dem Zahlenstrahl vorgestellt, nur dass wir die Klassen jetzt Null nennen. Und eins und sie auf der vertikalen Achse darstellen. Nun können Sie mit diesem Trainingsset versuchen, das Album anzuwenden, das Sie bereits kennen.

Führen Sie eine lineare Regression durch und versuchen Sie, eine gerade Linie an die Daten anzupassen. Wenn Sie das tun, sieht die gerade Linie vielleicht so aus, oder? Und das sind Ihre F-Effekte. Die lineare Regression sagt nicht nur die Werte Null und Eins voraus. Sondern alle Zahlen zwischen Null und Eins oder sogar kleiner als Null oder größer als Eins. Aber hier wollen wir Kategorien vorhersagen. Sie könnten versuchen, einen Schwellenwert von beispielsweise 0,5 festzulegen.



Wenn das Modell also einen Wert unter 0,5 ausgibt, können Sie vorhersagen, warum gleich Null oder nicht bösartig. Und wenn das Modell eine Zahl gleich oder größer als 0,5 ausgibt, dann sagen Sie voraus, dass Y gleich eins oder bösartig ist. Beachten Sie, dass dieser Schwellenwert von 0,5 an diesem Punkt die beste Anpassungsgerade schneidet. Wenn Sie also hier diese vertikale Linie zeichnen, ergibt sich für alles links die Vorhersage, dass y gleich Null ist. Und alles auf der rechten Seite endet mit der Vorhersage, dass y gleich eins ist. Nun sieht es so aus, als könnte die lineare Regression für diesen speziellen Datensatz etwas Vernünftiges bewirken.

Aber jetzt wollen wir sehen, was passiert, wenn Ihr Datensatz ein weiteres Trainingsbeispiel enthält. Dieser Weg hier rechts. Lassen Sie uns auch die horizontale Achse verlängern. Beachten Sie, dass dieses Trainingsbeispiel nicht wirklich die Art und Weise ändern sollte, wie Sie die Datenpunkte klassifizieren. Diese vertikale Trennlinie, die wir gerade gezogen haben, ist immer noch sinnvoll als Grenze, bei der Tumoren, die kleiner sind, als Null

klassifiziert werden sollten. Und größere Tumoren sollten als ein einziger Tumor klassifiziert werden. Aber sobald Sie dieses zusätzliche Trainingsbeispiel rechts hinzugefügt haben. Die beste Anpassungslinie für die lineare Regression verschiebt sich auf diese Weise. Und wenn Sie weiterhin den Schwellenwert von 0,5 verwenden, werden Sie feststellen, dass alles links von diesem Punkt als nicht bösartig Null vorhergesagt wird. Und alles rechts von diesem Punkt wird als eins oder bösartig vorhergesagt. Dies ist nicht das, was wir wollen, da das Hinzufügen dieses Beispiels rechts keine unserer Schlussfolgerungen zur Klassifizierung bösartiger und gutartiger Tumoren ändern sollte. Aber wenn Sie versuchen, dies mit linearer Regression zu erreichen, sollte das Hinzufügen dieses einen Beispiels scheinbar nichts ändern. Am Ende lernen wir eine viel schlechtere Funktion für dieses Klassifizierungsproblem.

Wenn der Tumor groß ist, möchten wir natürlich, dass der Algorithmus ihn als bösartig einstuft. Was wir gerade gesehen haben, ist, dass die lineare Regression die beste Anpassungslinie verursacht. Als wir rechts ein weiteres Beispiel hinzugefügt haben, um es zu verschieben. Und wird die Trennlinie auch Entscheidungsgrenze genannt, um sich nach rechts zu verschieben? Im nächsten Video erfahren Sie mehr über die Entscheidungsgrenze und einen Algorithmus namens logistische Regression. Wobei der Ausgabewert des Ergebnisses immer zwischen Null und Eins liegt. Und der Durchschnitt wird diese Probleme, die wir auf dieser Folie sehen, vermeiden. Was an dem Namen „Logistische Regression“ übrigens verwirrend ist, ist, dass er tatsächlich zur Klassifizierung verwendet wird, obwohl er das Wort „Regression“ enthält.

Lassen Sie sich nicht durch den Namen verwirren, der aus historischen Gründen vergeben wurde. Es wird tatsächlich verwendet, um binäre Klassifizierungsprobleme zu lösen, bei denen die Ausgabebezeichnung y entweder Null oder Eins ist. In der kommenden optionalen Übung können Sie auch einen Blick darauf werfen, was passiert, wenn Sie versuchen, lineare Regression zur Klassifizierung zu verwenden. Manchmal hat man Glück und es funktioniert vielleicht, aber oft funktioniert es nicht gut. Deshalb verwende ich selbst keine lineare Regression zur Klassifizierung. Im optionalen Labor sehen Sie einen interaktiven Plot, der versucht, zwischen zwei Kategorien zu klassifizieren. Und hoffentlich merkt man, dass das oft nicht so gut funktioniert. Das ist in Ordnung, da dies die Notwendigkeit eines anderen Modells für die Durchführung von Klassifizierungsgesprächen begründet. Schauen Sie sich also bitte dieses optionale Labor an. Anschließend schauen wir uns das nächste Video an.

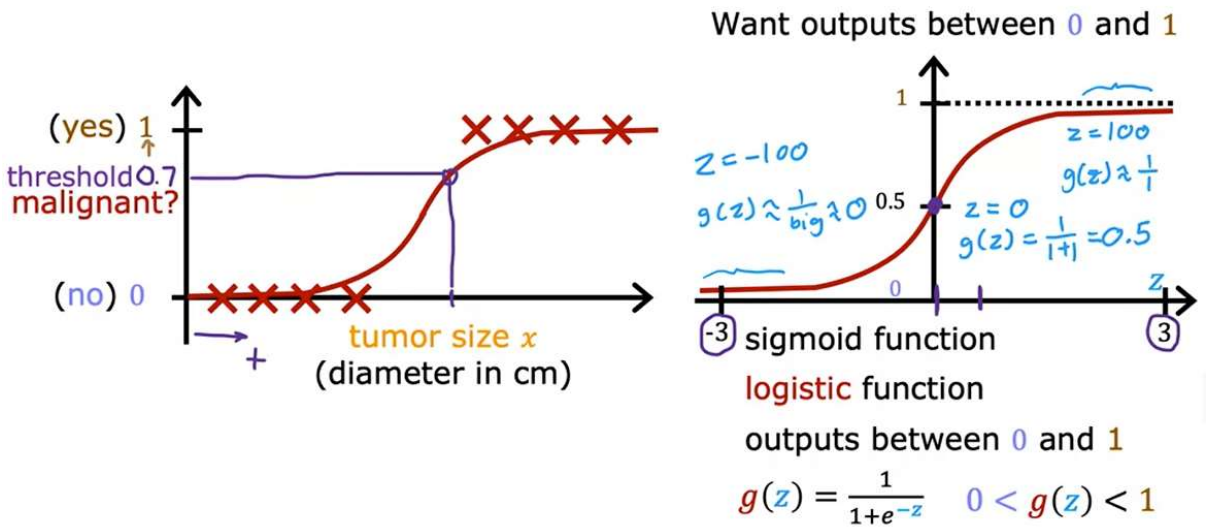
Logistische Regression zur Klassifizierung

Lassen Sie uns über die logistische Regression sprechen, den wahrscheinlich am häufigsten verwendeten Klassifizierungsalgorithmus der Welt. Das ist etwas, das ich bei meiner Arbeit ständig nutze. Fahren wir mit dem Beispiel der Klassifizierung fort, ob ein Tumor bösartig ist. Zuvor verwenden wir die Bezeichnung „1“ oder „Ja“ für die positive Klasse, um bösartige Tumoren darzustellen, und „null“ oder „Nein“ und negative Beispiele, um gutartige Tumoren darzustellen.

Hier ist ein Diagramm des Datensatzes, bei dem die horizontale Achse die Tumorgröße darstellt und die vertikale Achse nur die Werte 0 und 1 annimmt, da es sich um ein Klassifizierungsproblem handelt.

Sie haben im letzten Video gesehen, dass die lineare Regression kein guter Algorithmus für dieses Problem ist. Im Gegensatz dazu passen wir bei der logistischen Regression am Ende eine Kurve an, die so aussieht: eine S-förmige Kurve, an diesen Datensatz. Wenn in diesem Beispiel ein Patient mit einem Tumor dieser Größe kommt, den ich auf der x-Achse zeige, dann gibt der Algorithmus 0,7 aus, was darauf hindeutet, dass dieser eher bösartig oder gutartig oder möglicherweise eher bösartig ist. Ich werde später mehr sagen, was 0,7 in diesem Zusammenhang tatsächlich bedeutet. Aber die Ausgabebezeichnung y ist niemals 0,7, sondern immer nur 0 oder 1.

Um den logistischen Regressionsalgorithmus zu erweitern, gibt es eine wichtige mathematische Funktion, die ich gerne beschreibe, die Sigmoidfunktion genannt wird, manchmal auch als logistische Funktion bezeichnet. Die Sigmoid-Funktion sieht so aus. Beachten Sie, dass die x-Achse des Diagramms links und rechts unterschiedlich ist.



In der Grafik links auf der x-Achse ist die Tumorgröße angegeben, ebenso alle positiven Zahlen.

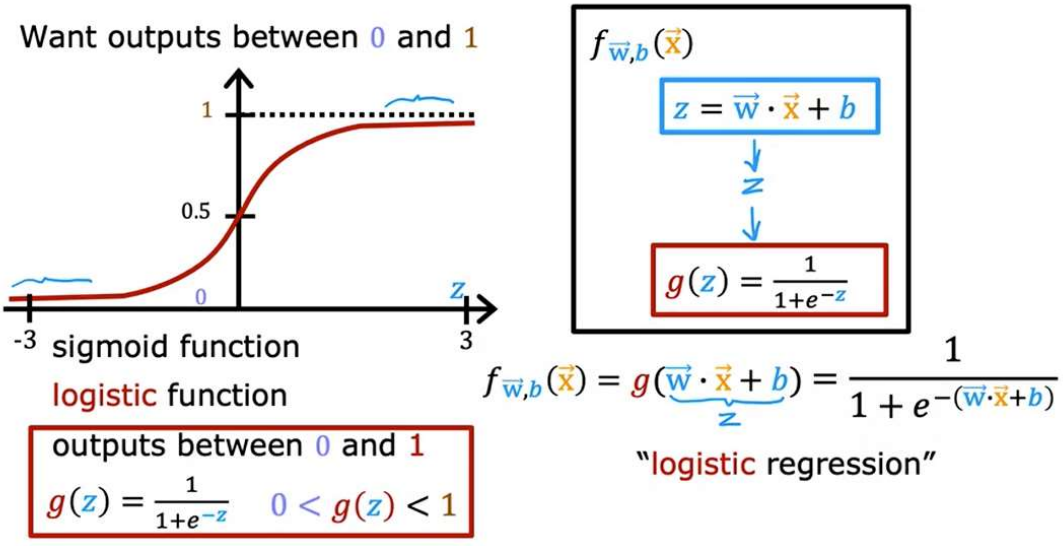
In der Grafik rechts hingegen haben Sie hier unten 0 und die horizontale Achse nimmt sowohl negative als auch positive Werte an und haben die horizontale Achse mit Z bezeichnet. Ich zeige hier nur einen Bereich von minus 3 bis plus 3. Der Ausgabewert der Sigmoidfunktion liegt zwischen 0 und 1.

Wenn ich g von z verwende, um diese Funktion zu bezeichnen, dann ist die Formel von g(z) gleich 1 über 1 plus e zum negativen z. Dabei ist e eine mathematische Konstante, die einen Wert von etwa **2,7** annimmt, und somit ist e hoch - z hoch negativ z. Beachten Sie, dass wenn z wirklich ist, sagen wir a 100, e zum negativen z gleich e zum negativen 100 ist, was eine winzige Zahl ist. Am Ende ist das also 1 über 1 plus einer winzigen kleinen Zahl, und der Nenner liegt im Grunde genommen sehr nahe bei 1. Deshalb wird g von z, das eine Sigmoidfunktion von z ist, sehr nahe bei z liegen, wenn z groß ist zu 1. Umgekehrt können Sie

auch selbst überprüfen, dass, wenn z eine sehr große negative Zahl ist, g von z über einer Riesenzahl zu 1 wird, weshalb g von z sehr nahe bei 0 liegt. Aus diesem Grund gilt die Sigmoidfunktion Diese Form beginnt sehr nahe bei Null und steigert sich langsam bis zum Wert Eins. Wenn in der Sigmoidfunktion z gleich 0 ist, dann ist e zum negativen z gleich e zum negativen 0, was gleich 1 ist, und daher ist g von z gleich 1 über 1 plus 1, was 0,5 ist. Das ist also so warum es die vertikale Achse bei 0,5 passiert.

Lassen Sie uns dies nun nutzen, um den logistischen Regressionsalgorithmus aufzubauen. Wir werden dies in zwei Schritten tun. Ich hoffe, Sie erinnern sich im ersten Schritt daran, dass eine gerade Linienfunktion wie eine lineare Regressionsfunktion als w definiert werden kann. Produkt von x plus b. Speichern wir diesen Wert in einer Variablen, die ich z nennen werde, und es wird sich herausstellen, dass dies dasselbe z ist wie das, das Sie auf der vorherigen Folie gesehen haben, aber dazu kommen wir gleich.

Der nächste Schritt besteht dann darin, diesen Wert von z zu nehmen und ihn an die Sigmoidfunktion, auch Logistikfunktion genannt, g zu übergeben. Nun gibt g von z einen durch diese Formel berechneten Wert aus: 1 über 1 plus e zum negativen z. Der Wert liegt zwischen 0 und 1. Wenn Sie diese beiden Gleichungen zusammensetzen, erhalten Sie das logistische Regressionsmodell f von x, das gleich g von wx + b ist. Oder äquivalent g von z, was dieser Formel hier entspricht. Dies ist das logistische Regressionsmodell, und es gibt ein Merkmal oder einen Satz von Merkmalen X ein und gibt eine Zahl zwischen 0 und 1 aus.

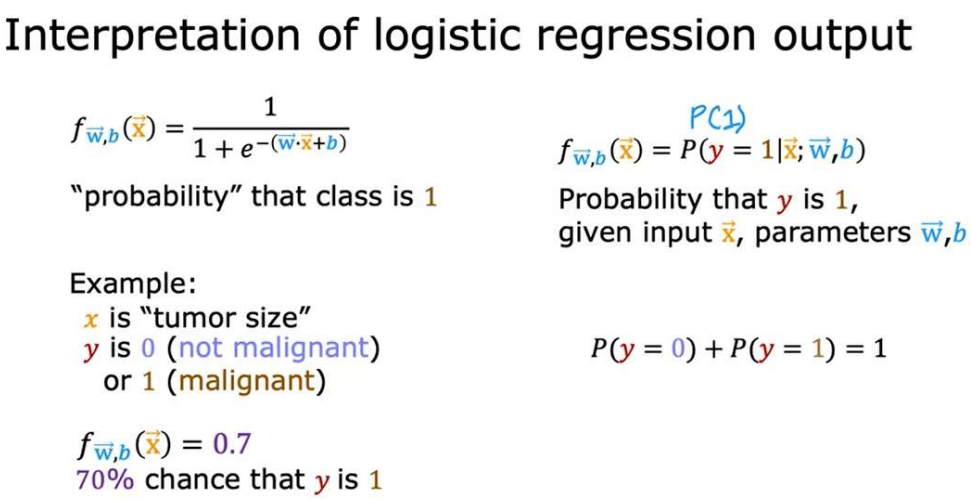


Als Nächstes werfen wir einen Blick darauf, wie die Ausgabe der logistischen Regression zu interpretieren ist. Wir kehren zum Beispiel der Tumorklassifizierung zurück. Ich ermutige Sie, sich die Ausgabe logistischer Regressionen so vorzustellen, dass sie die Wahrscheinlichkeit ausgibt, dass die Klasse oder die Bezeichnung y bei einer bestimmten Eingabe x gleich 1 ist.

In dieser Anwendung ist beispielsweise x die Tumorgröße und y entweder 0 oder 1, wenn eine Patientin hereinkommt und sie einen Tumor mit einer bestimmten Größe x hat, und wenn auf dieser Eingabe x basiert, das Modell I ll plus 0,7, dann bedeutet das, dass das Modell eine Vorhersage macht oder davon ausgeht, **dass eine 70-prozentige Wahrscheinlichkeit besteht, dass die wahre Bezeichnung y für diesen Patienten gleich 1**

wäre. Mit anderen Worten: Das Modell sagt uns, dass es davon ausgeht, dass der Patient eine 70-prozentige Wahrscheinlichkeit hat, dass der Tumor bösartig ist. Lassen Sie mich Ihnen jetzt eine Frage stellen. Sehen Sie, ob Sie das richtig machen können.

Wir wissen, dass y entweder 0 oder 1 sein muss. Wenn y also eine 70-prozentige Chance hat, 1 zu sein, wie hoch ist dann die Chance, dass es 0 ist? Also muss y entweder 0 oder 1 sein, und daher muss die Wahrscheinlichkeit, dass diese beiden Zahlen 0 oder 1 sind, eins oder eine 100-prozentige Wahrscheinlichkeit ergeben. Wenn also die Wahrscheinlichkeit, dass y 1 ist, 0,7 oder 70 Prozent beträgt, muss die Wahrscheinlichkeit, dass es 0 ist, 0,3 oder 30 Prozent betragen. Wenn Sie eines Tages Forschungsarbeiten oder Blog-Beiträge zu allen logistischen Regressionen lesen, sehen Sie manchmal die Notation, dass f(x) gleich p von y gleich 1 ist, wenn die Eingabemerkmale x und die Parameter w und b gegeben sind. Was das Semikolon hier verwendet, ist lediglich, dass w und b Parameter sind, die sich auf diese Berechnung auswirken. Wie groß ist die Wahrscheinlichkeit, dass y bei gegebenem Eingabemerkmal x gleich 1 ist?



Machen Sie sich für diesen Kurs keine allzu großen Gedanken darüber, was diese vertikale Linie und das Semikolon bedeuten. Sie müssen sich für diesen Kurs keine dieser mathematischen Notationen merken oder befolgen. Ich erwähne dies nur, weil Sie es möglicherweise an anderen Orten sehen. In der optionalen Übung, die diesem Video folgt, können Sie auch sehen, wie die Sigmoid-Funktion im Code implementiert wird. Sie können ein Diagramm sehen, das die Sigmoid-Funktion verwendet, um die Klassifizierungsaufgaben, die Sie in der vorherigen optionalen Übung gesehen haben, besser zu bewältigen.

```
def sigmoid(z):
    """
    Compute the sigmoid of z

    Args:
        z (ndarray): A scalar, numpy array of any size.

    Returns:
        g (ndarray): sigmoid(z), with the same shape as z

    """

    g = 1/(1+np.exp(-z))

    return g
```

Denken Sie daran, dass Ihnen der Code zur Verfügung gestellt wird und Sie ihn nur ausführen müssen. Ich hoffe, Sie werfen einen Blick darauf und machen sich mit dem Code vertraut. Sie wissen jetzt, was das logistische Regressionsmodell ist und welche mathematische Formel die logistische Regression definiert. Lange Zeit wurde ein Großteil der Internetwerbung im Wesentlichen durch eine leichte Variation der logistischen Regression gesteuert. Dies war für einige große Unternehmen sehr lukrativ, und im Grunde ist dies der Algorithmus, der entschieden hat, welche Anzeige Ihnen und vielen anderen auf einigen großen Websites gezeigt wurde. Jetzt gibt es noch mehr über diesen Algorithmus zu erfahren. Im nächsten Video werfen wir einen Blick auf die Details der logistischen Regression. Wir werden uns einige Visualisierungen ansehen und auch etwas untersuchen, das als Entscheidungsgrenze bezeichnet wird. Dadurch erhalten Sie verschiedene Möglichkeiten, die von diesem Modell ausgegebenen Zahlen, z. B. 0,3, 0,7 oder 0,65, einer Vorhersage zuzuordnen, ob y tatsächlich 0 oder 1 ist.

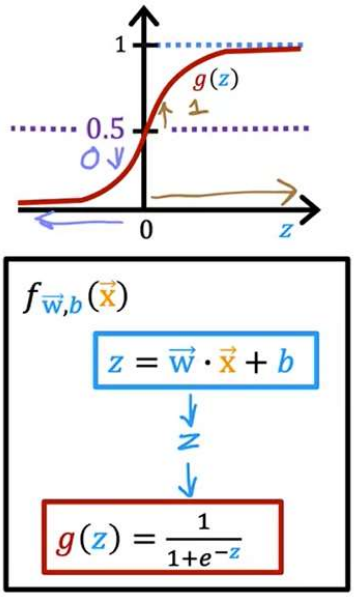
Entscheidungsgrenze

Im letzten Video haben Sie etwas über das logistische Regressionsmodell gelernt. Werfen wir nun einen Blick auf die Entscheidungsgrenze, um ein besseres Gefühl dafür zu bekommen, wie die logistische Regression diese Vorhersagen berechnet. Um es noch einmal zusammenzufassen: Hier erfahren Sie, wie die Ergebnisse des logistischen Regressionsmodells in zwei Schritten berechnet werden.

Im ersten Schritt berechnen Sie z als wx + b. Dann wenden Sie die Sigmoidfunktion g auf diesen Wert z an. Hier ist noch einmal die Formel für die Sigmoid-Funktion. Anders ausgedrückt: Wir können sagen, dass f(x) gleich g ist, die Sigmoidfunktion, auch logistische Funktion genannt, angewendet auf wx + b, wobei dies natürlich der Wert von z ist. Wenn Sie die Definition der Sigmoidfunktion nehmen und die Definition von z einfügen, dann finden Sie, dass f(x) gleich dieser Formel hier ist, 1 über 1 plus e zum negativen z, wobei z wx + b ist. Sie erinnern sich vielleicht, dass wir im vorherigen Video gesagt haben, dass wir dies als die Wahrscheinlichkeit interpretieren, dass y bei gegebenem x und mit den Parametern w und b gleich 1 ist. Das wird eine Zahl sein, etwa 0,7 oder 0,3.

Was ist nun, wenn Sie den Algorithmus zur Vorhersage erlernen möchten? Wird der Wert von y Null oder Eins sein? Nun, eine Sache, die Sie tun könnten, wäre, einen Schwellenwert festzulegen, über dem Sie vorhersagen, dass y eins ist, oder Sie setzen ŷ so, dass die Vorhersage gleich eins ist, und unter diesem Wert könnten Sie sagen, y hat, meine Vorhersage wird gleich Null sein. Eine übliche Wahl wäre, einen Schwellenwert von 0,5 zu wählen, sodass, wenn f(x) größer oder gleich 0,5 ist, y als eins vorhergesagt wird. Wir schreiben diese Vorhersage so, dass ŷ gleich 1 ist, oder wenn f(x) kleiner als 0,5 ist, dann ist

y die Vorhersage 0, oder mit anderen Worten, die Vorhersage ŷ ist gleich 0. Lassen Sie uns nun tiefer in die Frage eintauchen, wann das Modell dies tun würde einen vorhersagen.



$$f_{\vec{w},b}(\vec{x}) = g(\underbrace{\vec{w} \cdot \vec{x} + b}_{z}) = \frac{1}{1+e^{-(\vec{w} \cdot \vec{x} + b)}}$$

$$= P(y = 1 | x; \vec{w}, b) \quad 0.7 \quad 0.3$$

0 or 1? threshold

Is $f_{\vec{w},b}(\vec{x}) \geq 0.5$?

Yes: $\hat{y} = 1$ No: $\hat{y} = 0$

When is $f_{\vec{w},b}(\vec{x}) \geq 0.5$?

$g(z) \geq 0.5$

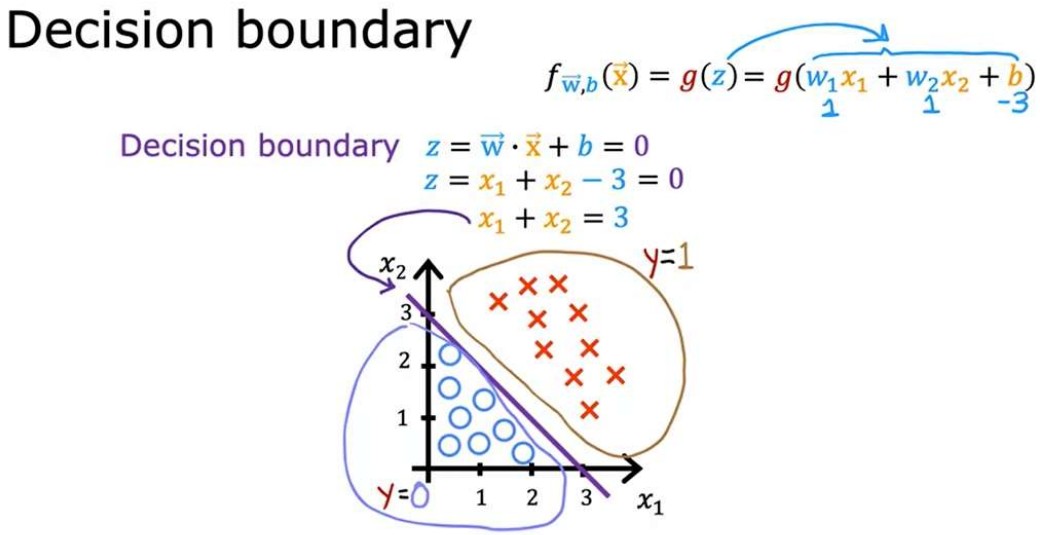
$z \geq 0$

$\vec{w} \cdot \vec{x} + b \geq 0$ $\quad \vec{w} \cdot \vec{x} + b < 0$

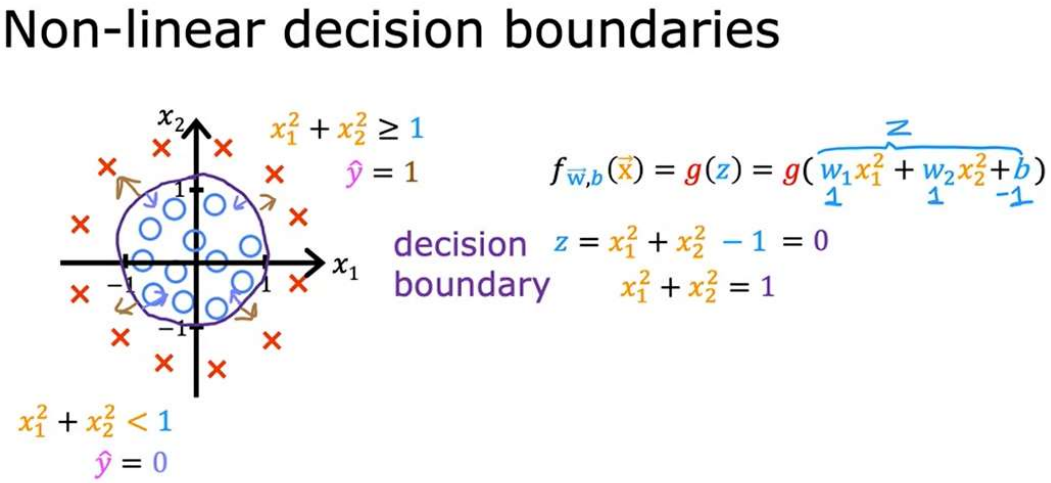
$\hat{y} = 1$ $\quad \hat{y} = 0$

Mit anderen Worten, wenn f(x) größer oder gleich 0,5 ist. Wir erinnern uns, dass f(x) genau gleich g von z ist. f ist also immer dann größer oder gleich 0,5, wenn g von z größer oder gleich 0,5 ist. Aber wann ist g von z größer oder gleich 0,5? Nun, hier ist eine Sigmoid-Funktion. Also ist g von z immer dann größer oder gleich 0,5, wenn z größer oder gleich 0 ist. Das heißt, immer wenn z auf der rechten Hälfte dieser Achse liegt. Wann ist z schließlich größer oder gleich Null? Nun, z ist gleich wx + b, also ist z immer dann größer oder gleich Null, wenn wx + b größer oder gleich Null ist. Um es noch einmal zusammenzufassen: Was Sie hier gesehen haben, ist, dass das Modell immer dann 1 vorhersagt, wenn wx + b größer oder gleich 0 ist.

Umgekehrt sagt der Algorithmus voraus, dass y 0 ist, wenn wx + b kleiner als Null ist Visualisieren Sie, wie das Modell Vorhersagen trifft. Ich werde ein Beispiel für ein Klassifizierungsproblem nehmen, bei dem es zwei Features gibt, x1 und x2, statt nur eines Features. Hier ist ein Trainingssatz, bei dem die kleinen roten Kreuze die positiven Beispiele und die kleinen blauen Kreise die negativen Beispiele kennzeichnen. Die roten Kreuze entsprechen y gleich 1 und die blauen Kreise entsprechen y gleich 0. Das logistische Regressionsmodell wird mithilfe dieser Funktion Vorhersagen treffen: f(x) gleich g von z, wobei z nun dieser Ausdruck hier ist, w1x1 plus w2x2 plus b, weil wir zwei Merkmale x1 und x2 haben. Nehmen wir für dieses Beispiel einfach an, dass die Werte der Parameter w1 gleich 1, w2 gleich 1 und b gleich minus 3 sind.



Schauen wir uns nun an, wie die logistische Regression Vorhersagen trifft. Lassen Sie uns insbesondere herausfinden, wann wx + b größer als gleich 0 ist und wann wx + b kleiner als 0 ist. Um das herauszufinden, müssen wir uns eine sehr interessante Linie ansehen, nämlich wann wx + b genau gleich ist 0. Es stellt sich heraus, dass diese Linie auch als Entscheidungsgrenze bezeichnet wird, da es sich dabei um die Linie handelt, bei der Sie fast neutral darüber sind, ob y 0 oder y 1 ist. Nun zu den Werten der Parameter w_1, w_2 und b Wie wir oben aufgeschrieben haben, ist diese Entscheidungsgrenze einfach x_1 plus x_2 minus 3. Wann ist x_1 plus x_2 minus 3 gleich 0? Nun, das entspricht der Linie x_1 plus x_2 gleich 3, und das ist diese Linie, die hier gezeigt wird. Diese Linie erweist sich als Entscheidungsgrenze. Wenn sich die Merkmale x rechts von dieser Linie befinden, würde die logistische Regression 1 vorhersagen und links von dieser Linie würde die logistische Regression 0 vorhersagen.



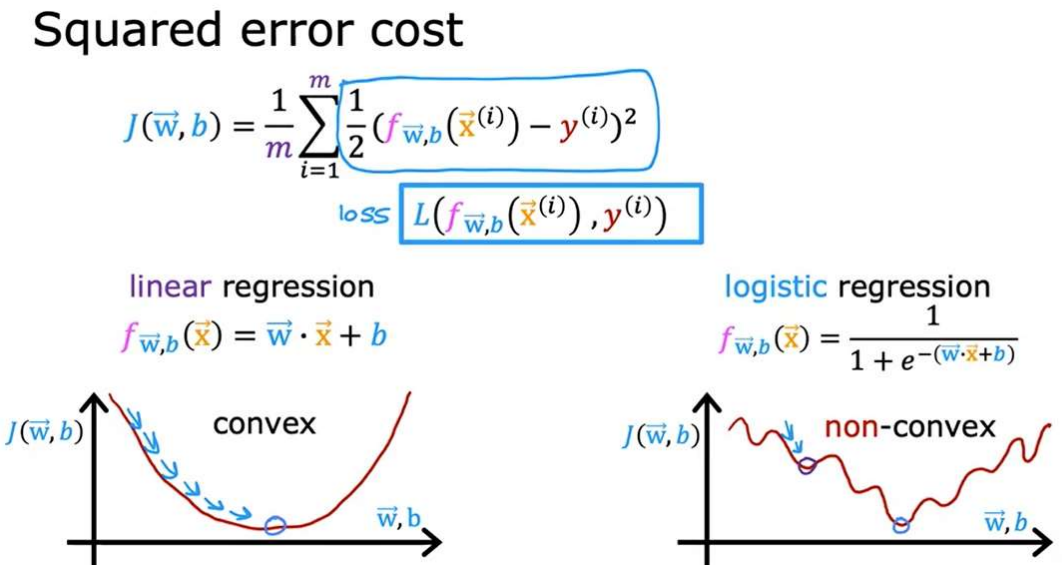
Mit anderen Worten, was wir gerade haben visualize ist die Entscheidungsgrenze für die logistische Regression, wenn die Parameter w_1, w_2 und b 1,1 und negativ 3 sind. Wenn Sie eine andere Wahl der Parameter hätten, wäre die Entscheidungsgrenze natürlich eine

andere Linie. Schauen wir uns nun ein komplexeres Beispiel an, bei dem die Entscheidungsgrenze keine gerade Linie mehr ist. Wie zuvor bezeichnen Kreuze die Klasse y gleich 1 und die kleinen Kreise bezeichnen die Klasse y gleich 0. Anfang letzter Woche haben Sie gesehen, wie man Polynome in der linearen Regression verwendet, und Sie können dasselbe in der logistischen Regression tun. Dies setzt z auf w_1, x_1 zum Quadrat plus w_2, x_2 zum Quadrat plus b. Mit dieser Auswahl an Merkmalen werden Polynommerkmale in eine logistische Regression umgewandelt. F von x, das gleich g von z ist, ist jetzt g dieses Ausdrucks hier. Nehmen wir an, wir haben am Ende w_1 und w_2 als 1 und b als negativ 1 gewählt. Z ist gleich 1 mal x_1 zum Quadrat plus 1 mal x_2 zum Quadrat minus 1. Die Entscheidungsgrenze entspricht wie zuvor dem Fall, dass z gleich ist auf 0. Dieser Ausdruck ist gleich 0, wenn x_1 zum Quadrat plus x_2 zum Quadrat gleich 1 ist. Wenn Sie im Diagramm links zeichnen, ist die Kurve, die x_1 zum Quadrat plus x_2 zum Quadrat entspricht, gleich 1, dies stellt sich als Kreis heraus . Wenn x_1 zum Quadrat plus x_2 zum Quadrat größer oder gleich 1 ist, ist das dieser Bereich außerhalb des Kreises und das ist der Zeitpunkt, an dem Sie y als 1 vorhersagen. Wenn umgekehrt x_1 zum Quadrat plus x_2 zum Quadrat kleiner als 1 ist, ist das dieser Bereich innerhalb des Kreises und Dann sagen Sie voraus, dass y 0 ist. Können wir noch komplexere Entscheidungsgrenzen als diese finden? Ja, du kannst. Sie können dies erreichen, indem Sie Polynomterme noch höherer Ordnung verwenden. Angenommen, z ist w_1, x_1 plus w_2, x_2 plus w_3, x_1 im Quadrat plus w_4, x_1, x_2 plus w_5, x_2 im Quadrat. Dann ist es möglich, dass Sie noch komplexere Entscheidungsgrenzen erhalten. Das Modell kann Entscheidungsgrenzen definieren, wie in diesem Beispiel, eine Ellipse wie diese oder mit einer anderen Wahl der Parameter. Sie können sogar komplexere Entscheidungsgrenzen erhalten, die wie Funktionen aussehen können, die möglicherweise so aussehen. Dies ist also ein Beispiel für eine noch komplexere Entscheidungsgrenze als die, die wir zuvor gesehen haben. Diese Implementierung der logistischen Regression wird y gleich 1 innerhalb dieser Form vorhersagen und außerhalb der Form y gleich 0 vorhersagen. Mit diesen Polynomfunktionen können Sie sehr komplexe Entscheidungsgrenzen erhalten. Mit anderen Worten: Die logistische Regression kann lernen, ziemlich komplexe Daten anzupassen. Wenn Sie jedoch keines dieser Polynome höherer Ordnung einbeziehen würden, also die einzigen Features, die Sie verwenden, x_1, x_2, x_3 usw. sind, wäre die Entscheidungsgrenze für die logistische Regression immer linear und immer eine Gerade Linie. Im kommenden optionalen Labor können Sie auch die Code-Implementierung der Entscheidungsgrenze sehen. Im Beispiel im Labor gibt es zwei Features, sodass Sie diese Entscheidungsgrenze als Linie sehen können. Ich hoffe, dass Sie mit dieser Visualisierung nun einen Eindruck von der Bandbreite der möglichen Modelle bekommen, die Sie mit der logistischen Regression erhalten können. Nachdem Sie nun gesehen haben, was f(x) potenziell berechnen kann, werfen wir einen Blick darauf, wie Sie tatsächlich ein logistisches Regressionsmodell trainieren können. Wir beginnen mit der Betrachtung der Kostenfunktion für die logistische Regression und finden anschließend heraus, wie man den Gradientenabstieg darauf anwendet. Kommen wir zum nächsten Video.

Kostenfunktion für die logistische Regression

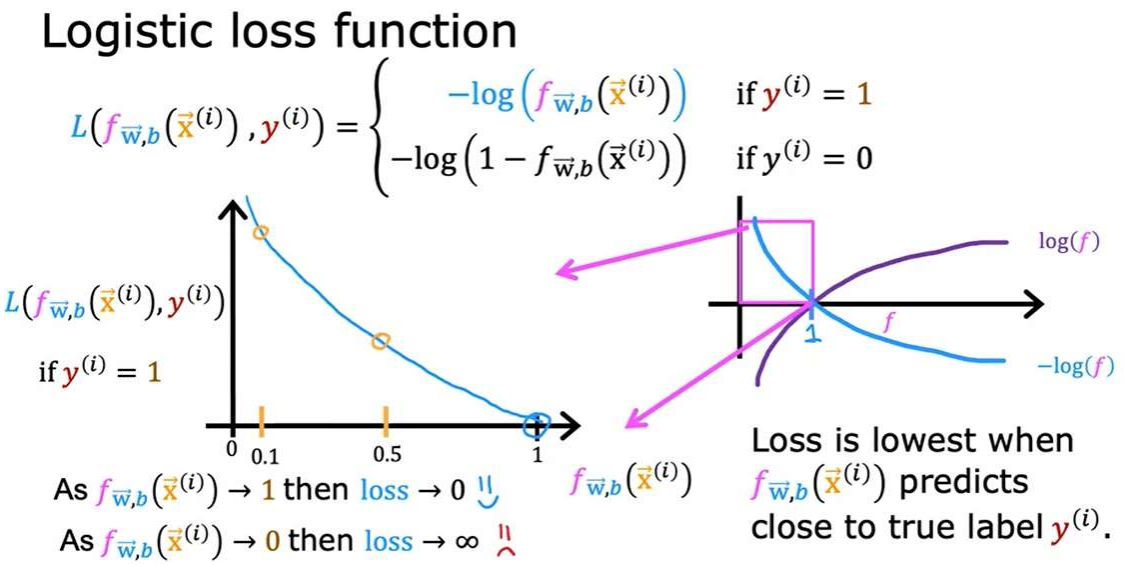
Denken Sie daran, dass Sie mit der Kostenfunktion messen können, wie gut ein bestimmter Parametersatz zu den Trainingsdaten passt. Dadurch haben Sie die Möglichkeit, bessere

Parameter auszuwählen. In diesem Video sehen wir uns an, dass die quadrierte Fehlerkostenfunktion keine ideale Kostenfunktion für die logistische Regression ist. Wir werfen einen Blick auf eine andere Kostenfunktion, die uns dabei helfen kann, bessere Parameter für die logistische Regression auszuwählen. So könnte der Trainingssatz für unser logistisches Regressionsmodell aussehen. Wobei hier jede Zeile Patienten entsprechen könnte, die einen Arztbesuch abstatteten, und eine Zeile sich mit einer Diagnose befasste. Wie zuvor verwenden wir m, um die Anzahl der Trainingsbeispiele anzugeben. Jedes Trainingsbeispiel verfügt über ein oder mehrere Merkmale, z. B. die Tumorgröße, das Alter des Patienten usw. für insgesamt n Merkmale. Nennen wir die Features X_1 bis X_n. Da es sich um eine binäre Klassifizierungsaufgabe handelt, nimmt die Zielbezeichnung y nur zwei Werte an, entweder 0 oder 1. Schließlich wird das logistische Regressionsmodell durch diese Gleichung definiert. Die Frage, die Sie beantworten möchten, lautet: Wie können Sie angesichts dieses Trainingssatzes die Parameter w und b auswählen? Zur Erinnerung: Bei der linearen Regression handelt es sich um die quadratische Fehlerkostenfunktion. Das Einzige , was ich geändert habe, ist, dass ich die eine Hälfte innerhalb der Summierung statt außerhalb der Summierung platziert habe. Sie erinnern sich vielleicht daran, dass im Fall der linearen Regression f(x) die lineare Funktion ist, w Punkt x plus b. Die Kostenfunktion sieht so aus, ist eine konvexe Funktion oder eine Schüsselform oder Hammerform. Der Gradientenabstieg sieht so aus, dass man einen Schritt, einen Schritt usw. macht, um am globalen Minimum zu konvergieren. Jetzt könnten Sie versuchen, dieselbe Kostenfunktion für die logistische Regression zu verwenden.



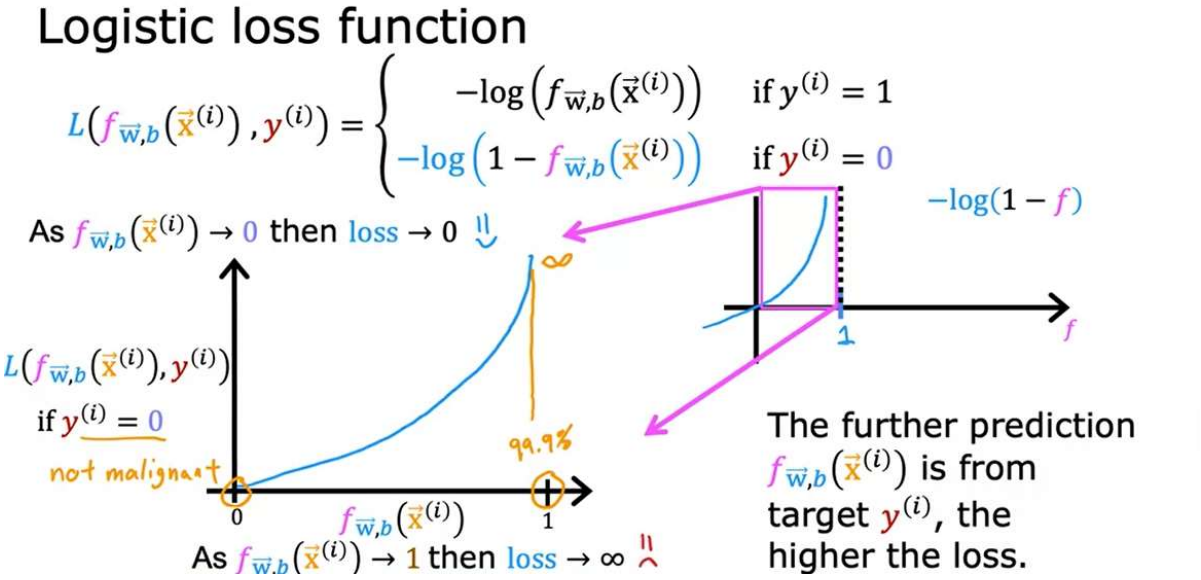
Aber es stellt sich heraus, dass, wenn ich f(x) gleich 1 über 1 plus e zum negativen wx + b schreiben würde und die Kostenfunktion unter Verwendung dieses Werts von f(x) grafisch darstellen würde, die Kosten so aussehen würden. Dies wird zu einer sogenannten nicht konvexen Kostenfunktion, die nicht konvex ist. Dies bedeutet, dass Sie versuchen würden, den Gradientenabstieg zu verwenden. Es gibt viele lokale Minima, die man erreichen kann. Es stellt sich heraus, dass diese quadratische Fehlerkostenfunktion für die logistische Regression keine gute Wahl ist. Stattdessen wird es eine andere Kostenfunktion geben, die die Kostenfunktion wieder konvex machen kann. Es kann garantiert werden, dass der Gradientenabstieg zum globalen Minimum konvergiert. Das Einzige, was ich geändert habe,

ist, dass ich die eine Hälfte innerhalb der Summierung statt außerhalb der Summierung platziert habe. Dadurch wird die Mathematik, die Sie später auf dieser Folie sehen, etwas einfacher. Um eine neue Kostenfunktion zu erstellen, die wir für die logistische Regression verwenden werden. Ich werde die Definition der Kostenfunktion J von w und b ein wenig ändern. Wenn Sie sich diese Zusammenfassung genauer ansehen, nennen wir diesen Begriff innerhalb des Verlusts für ein einzelnes Trainingsbeispiel. Ich werde den Verlust mit diesem großen L und als Funktion der Vorhersage des Lernalgorithmus, f(x) sowie der wahren Bezeichnung y bezeichnen. Der Verlust angesichts des Prädiktors f(x) und der wahren Bezeichnung y beträgt in diesem Fall 1,5 der quadrierten Differenz. Wir werden gleich sehen, dass durch die Wahl einer anderen Form für diese Verlustfunktion die Gesamtkostenfunktion, die 1 über dem n-fachen der Summe dieser Verlustfunktionen beträgt, als konvexe Funktion beibehalten werden kann. Nun gibt die Verlustfunktion f(x) und die wahre Bezeichnung y ein und sagt uns, wie gut wir in diesem Beispiel abschneiden.



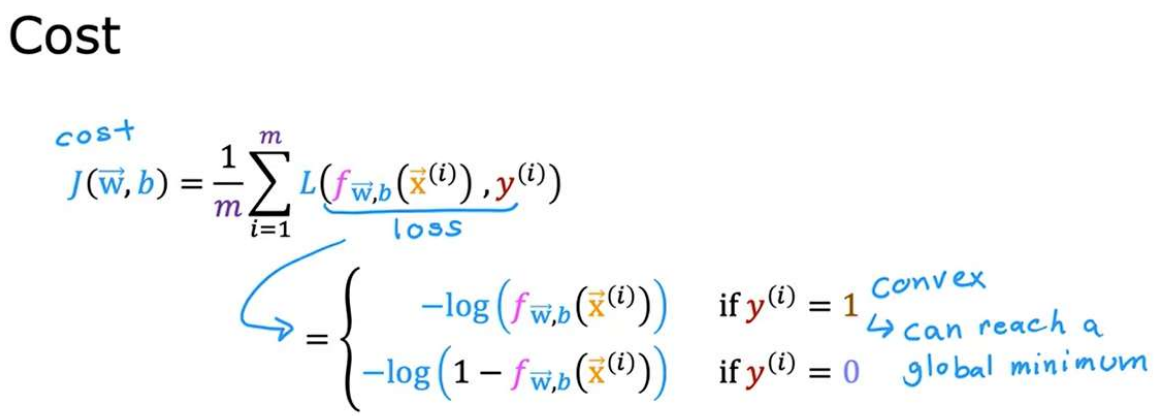
Ich werde hier einfach die Definition der Verlustfunktion aufschreiben, die wir für die logistische Regression verwenden werden. Wenn die Bezeichnung y gleich 1 ist, beträgt der Verlust einen negativen Logarithmus von f von x, und wenn die Bezeichnung y gleich 0 ist, beträgt der Verlust einen negativen Logarithmus von 1 minus f von x. Werfen wir einen Blick darauf, warum diese Verlustfunktion hoffentlich Sinn macht. Betrachten wir zunächst den Fall, dass y gleich 1 ist, und zeichnen wir auf, wie diese Funktion aussieht, um eine Vorstellung davon zu bekommen, was diese Verlustfunktion bewirkt. Denken Sie daran, dass die Verlustfunktion misst, wie gut Sie bei einem Trainingsbeispiel abschneiden. Durch Summieren der Verluste bei allen Trainingsbeispielen, die Sie dann erhalten, ergibt sich die Kostenfunktion, die misst, wie gut Sie bei dem gesamten Training abschneiden Satz. Wenn Sie den Logarithmus von f zeichnen, sieht es wie diese Kurve hier aus, wobei f hier auf der horizontalen Achse liegt. Ein Diagramm des Negativs des Logarithmus von f sieht folgendermaßen aus, wobei wir die Kurve einfach entlang der horizontalen Achse umdrehen. Beachten Sie, dass es die horizontale Achse bei f gleich 1 schneidet und von dort aus weiter nach unten verläuft. Nun ist f die Ausgabe der logistischen Regression. Daher liegt f immer zwischen null und eins, da die Ausgabe der logistischen Regression immer zwischen null und eins liegt. Der einzige Teil der Funktion, der relevant ist, ist daher dieser Teil hier, der f

zwischen 0 und 1 entspricht. Schauen wir uns diesen Teil des Diagramms genauer an. Wenn der Algorithmus eine Wahrscheinlichkeit nahe 1 vorhersagt und die wahre Bezeichnung 1 ist, ist der Verlust sehr gering. Es ist so ziemlich 0, weil Sie der richtigen Antwort sehr nahe sind. Fahren Sie nun mit dem Beispiel fort, bei dem die wahre Bezeichnung y 1 ist, sagen wir, alles sei ein bösartiger Tumor. Wenn der Algorithmus 0,5 vorhersagt, dann liegt der Verlust an diesem Punkt, der etwas höher, aber nicht so hoch ist. Wenn der Algorithmus im Gegensatz dazu einen Ausgang von 0,1 haben würde, wenn er davon ausgeht, dass die Wahrscheinlichkeit, dass der Tumor bösartig ist, nur bei 10 Prozent liegt, y aber tatsächlich 1 ist. Wenn er wirklich bösartig ist, dann ist der Verlust hier um diesen viel höheren Wert . Wenn y gleich 1 ist, gibt die Verlustfunktion einen Anreiz oder fördert oder hilft dem Algorithmus, genauere Vorhersagen zu treffen, da der Verlust am geringsten ist, wenn Werte nahe 1 vorhergesagt werden.



Auf dieser Folie schauen wir uns nun an, was Der Verlust liegt vor, wenn y gleich 1 ist. Schauen wir uns auf dieser Folie den zweiten Teil der Verlustfunktion an, der entspricht, wenn y gleich 0 ist. In diesem Fall ist der Verlust ein negativer Logarithmus von 1 minus f von x. Wenn diese Funktion dargestellt wird, sieht sie tatsächlich so aus. Der Bereich von f ist auf 0 bis 1 begrenzt, da die logistische Regression nur Werte zwischen 0 und 1 ausgibt. Wenn wir hineinzoomen, sieht es so aus. In diesem Diagramm zeigt die vertikale Achse, wenn y gleich 0 ist, den Wert des Verlusts für verschiedene Werte von f von x. Wenn f 0 oder sehr nahe bei 0 ist, wird der Verlust ebenfalls sehr gering sein. Das heißt, wenn die wahre Bezeichnung 0 ist und die Vorhersage des Modells sehr nahe bei 0 liegt, haben Sie fast alles richtig gemacht, sodass der Verlust angemessen ist sehr nahe bei 0. Je größer der Wert von f(x) wird, desto größer ist der Verlust, da die Vorhersage weiter von der wahren Bezeichnung 0 entfernt ist. Tatsächlich geht der Verlust tatsächlich gegen Unendlich, wenn sich diese Vorhersage 1 nähert. Um noch einmal auf das Beispiel der Tumorvorhersage zurückzukommen: Wenn das Modell vorhersagt, dass der Tumor des Patienten mit an Sicherheit grenzender Wahrscheinlichkeit bösartig ist, beispielsweise mit einer Wahrscheinlichkeit von 99,9 Prozent, dass er tatsächlich nicht bösartig ist, also y gleich 0 ist, dann bestrafen wir das Modell mit einem sehr hohen Verlust. In diesem Fall ist y gleich 0. Dies gilt auch für den Fall,

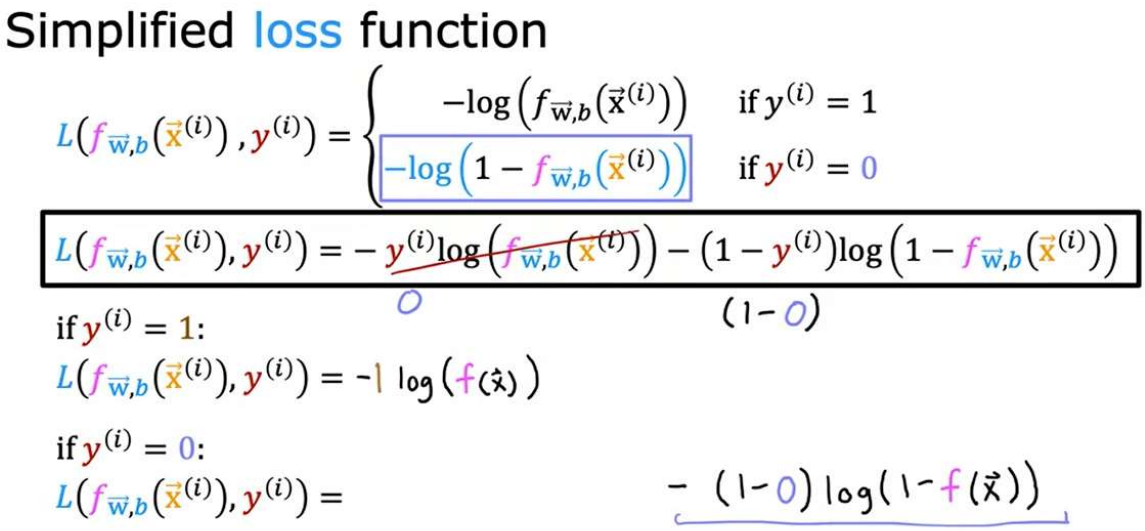
dass y gleich 1 auf der vorherigen Folie ist. Je weiter die Vorhersage f(x) vom wahren Wert von y entfernt ist, desto höher ist der Verlust. Wenn f(x) gegen 0 geht, ist der Verlust tatsächlich sehr groß und nähert sich tatsächlich der Unendlichkeit. Wenn die wahre Bezeichnung 1 ist, besteht ein starker Anreiz für den Algorithmus, nichts vorherzusagen, das zu nahe bei 0 liegt. In diesem Video haben Sie gesehen, warum die Funktion „Quadratfehlerkosten“ für die logistische Regression nicht gut funktioniert. Wir haben auch den Verlust für ein einzelnes Trainingsbeispiel definiert und eine neue Definition für die Verlustfunktion für die logistische Regression erstellt. Es stellt sich heraus, dass bei dieser Wahl der Verlustfunktion die Gesamtkostenfunktion konvex ist und Sie daher zuverlässig den Gradientenabstieg verwenden können, um zum globalen Minimum zu gelangen. Der Beweis, dass diese Funktion konvex ist, liegt außerhalb des Rahmens dieser Kosten.



Sie erinnern sich vielleicht, dass die Kostenfunktion eine Funktion des gesamten Trainingssatzes ist und daher der Durchschnitt oder 1 über m mal die Summe der Verlustfunktion der einzelnen Trainingsbeispiele ist. Die Kosten für einen bestimmten Satz von Parametern, w und b, betragen 1 über das m-fache der Summe aller Trainingsbeispiele des Verlusts an den Trainingsbeispielen. Wenn Sie den Wert der Parameter w und b finden, der dies minimiert, dann hätten Sie einen ziemlich guten Wertesatz für die Parameter w und b für die logistische Regression. In der kommenden optionalen Übung werden Sie einen Blick darauf werfen, warum die Kostenquadratfunktion für die Klassifizierung nicht besonders gut funktioniert, da Sie sehen, dass das Oberflächendiagramm zu einer sehr schwankenden Kostenoberfläche mit vielen lokalen Minima führt. Dann werfen Sie einen Blick auf die neue Logistikverlustfunktion. Wie Sie hier sehen können, entsteht dadurch ein schönes und glattes konvexes Oberflächendiagramm, das nicht alle diese lokalen Minima aufweist. Bitte werfen Sie einen Blick auf die Kosten und die Grundstücke nach diesem Video. Wir haben in diesem Video viel gesehen. Gehen wir im nächsten Video zurück und nehmen wir die Verlustfunktion für ein einzelnes Zugbeispiel und definieren damit die Gesamtkostenfunktion für den gesamten Trainingssatz. Wir werden auch eine einfachere Möglichkeit finden, die Kostenfunktion zu schreiben, die es uns später ermöglicht, einen Gradientenabstieg durchzuführen, um gute Parameter für die logistische Regression zu finden. Kommen wir zum nächsten Video.

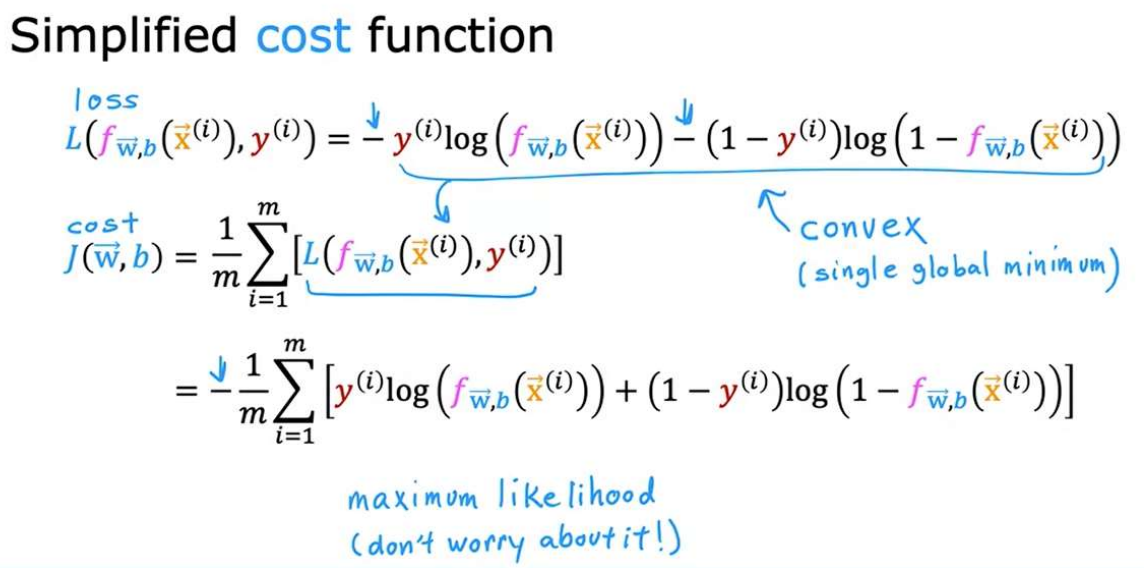
Vereinfachte Kostenfunktion für die logistische Regression

Im letzten Video haben Sie die Verlustfunktion und die Kostenfunktion für die logistische Regression gesehen. In diesem Video sehen Sie eine etwas einfachere Möglichkeit, die Verlust- und Kostenfunktionen auszuschreiben, sodass die Implementierung etwas einfacher sein kann, wenn wir zum Gradientenabstieg zur Anpassung der Parameter eines logistischen Regressionsmodells kommen. Lass uns einen Blick darauf werfen. Zur Erinnerung: Hier ist die Verlustfunktion, die wir im vorherigen Video für die logistische Regression definiert haben. Da wir immer noch an einem binären Klassifizierungsproblem arbeiten, ist y entweder Null oder Eins. Da y entweder Null oder Eins ist und keinen anderen Wert als Null oder Eins annehmen kann, können wir eine einfachere Möglichkeit finden, diese Verlustfunktion zu schreiben. Sie können die Verlustfunktion wie folgt schreiben. Bei einer Vorhersage f(x) und der Zielbezeichnung y beträgt der Verlust negativ y mal logarithmisch von f minus 1 minus y mal logarithmisch von 1 minus f. Es stellt sich heraus, dass diese Gleichung, die wir gerade in einer Zeile geschrieben haben, völlig äquivalent zu dieser komplexeren Formel hier oben ist.



Mal sehen, warum das so ist. Denken Sie daran, dass y nur die Werte Eins oder Null annehmen kann. Nehmen wir im ersten Fall an, dass y gleich 1 ist. Das erste y hier ist eins und diese 1 minus y ist 1 minus 1, was daher gleich 0 ist. Der Verlust wird also negativ 1 mal logarithmisch von f(x) minus 0 mal eine Menge Zeug. Das wird Null und verschwindet. Wenn y gleich 1 ist, ist der Verlust tatsächlich der erste Term oben, der negative Logarithmus von f von x. Schauen wir uns den zweiten Fall an, wenn y gleich 0 ist. In diesem Fall ist dieses y hier gleich 0, sodass dieser erste Term wegfällt und der zweite Term 1 minus 0 mal dieser logarithmische Term ist. Der Verlust wird zu diesem negativen 1-fachen Logarithmus von 1 minus f von x. Das entspricht genau dieser zweiten Amtszeit hier oben. Für den Fall, dass y gleich 0 ist, erhalten wir auch die ursprüngliche Verlustfunktion wie oben definiert zurück. Was Sie sehen ist, dass dieser einzelne Ausdruck hier unabhängig davon, ob y eins oder null ist, dem komplexeren Ausdruck hier oben entspricht , weshalb wir hiermit eine einfachere Möglichkeit haben, den Verlust mit nur einer Gleichung zu schreiben, ohne diese beiden Fälle zu trennen. wie wir es oben getan haben. Lassen Sie uns unter Verwendung dieser vereinfachten Verlustfunktion zurückgehen und die Kostenfunktion für die logistische

Regression schreiben. Auch hier ist die vereinfachte Verlustfunktion. Denken Sie daran, dass die Kosten J nur der durchschnittliche Verlust sind, der über den gesamten Trainingssatz von m Beispielen durchschnittlich ist. Es ist also 1 über das n-fache der Summe der Verluste von i gleich 1 bis m. Wenn Sie die Definition für den vereinfachten Verlust von oben einfügen, dann sieht es so aus: 1 über m mal die Summe dieses Termes oben. Wenn Sie die negativen Vorzeichen nach außen verschieben, erhalten Sie hier diesen Ausdruck, und das ist die Kostenfunktion. Die Kostenfunktion, die so ziemlich jeder zum Trainieren der logistischen Regression verwendet. Sie fragen sich vielleicht: Warum wählen wir diese spezielle Funktion, wenn wir doch jede Menge andere Kostenfunktionen hätten wählen können? Auch wenn wir in diesem Kurs keine Zeit haben, ausführlicher darauf einzugehen, möchte ich nur erwähnen, dass diese spezielle Kostenfunktion aus der Statistik abgeleitet wird und dabei ein statistisches Prinzip namens Maximum-Likelihood-Schätzung verwendet, eine Idee aus der Statistik wie man effizient Parameter für verschiedene Modelle findet.

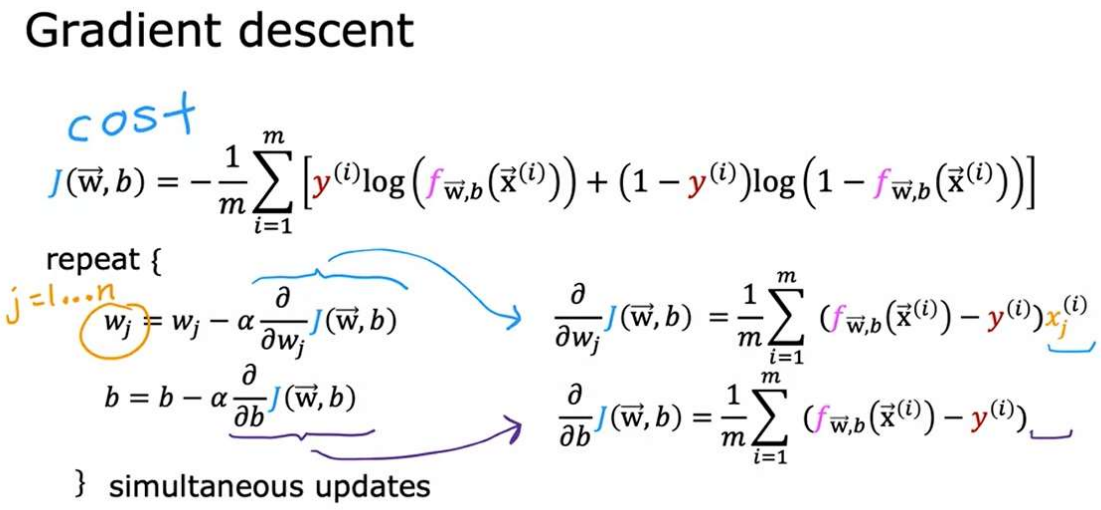


Diese Kostenfunktion hat die schöne Eigenschaft, dass sie konvex ist. Aber machen Sie sich keine Sorgen darüber, die Details der maximalen Wahrscheinlichkeit zu erfahren. Es handelt sich lediglich um eine tiefere Begründung und Rechtfertigung hinter dieser speziellen Kostenfunktion. In der kommenden optionalen Übung erfahren Sie, wie die Logistikkostenfunktion im Code implementiert wird. Ich empfehle, einen Blick darauf zu werfen, da Sie dies später am Ende der Woche im Praxislabor umsetzen. Diese bevorstehende optionale Übung zeigt Ihnen auch, wie zwei unterschiedliche Auswahlen der Parameter zu unterschiedlichen Kostenberechnungen führen. Im Diagramm können Sie sehen, dass die besser passende blaue Entscheidungsgrenze im Vergleich zur magentafarbenen Entscheidungsgrenze geringere Kosten verursacht. Mit der vereinfachten Kostenfunktion können wir nun mit der Anwendung des Gradientenabstiegs auf die logistische Regression beginnen. Schauen wir uns das im nächsten Video an.

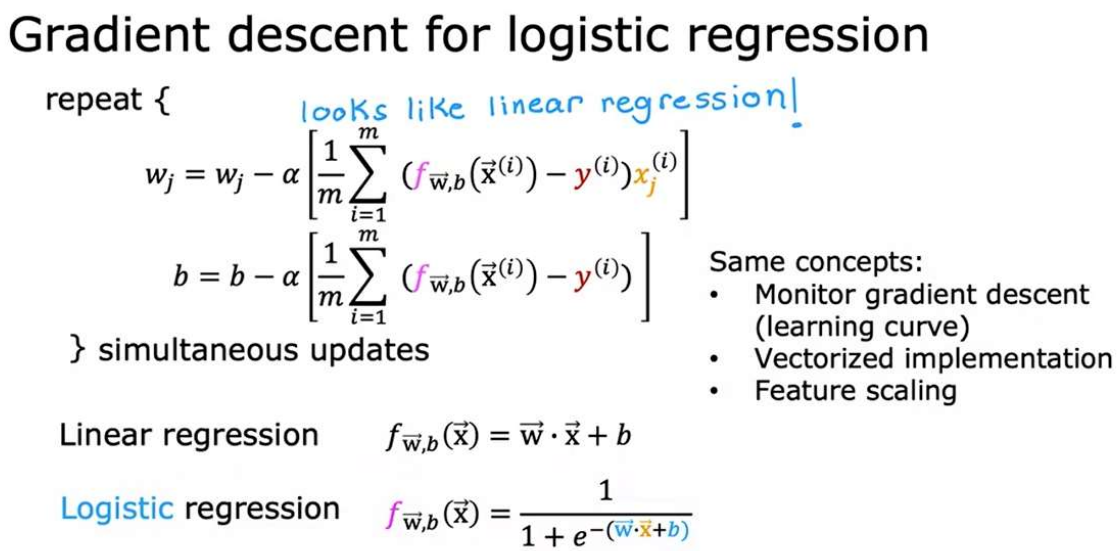
Implementierung des Gradientenabstiegs

Um die Parameter eines logistischen Regressionsmodells anzupassen, werden wir versuchen, die Werte der Parameter w und b zu finden, die die Kostenfunktion J von w und b

minimieren, und wir werden dazu erneut den Gradientenabstieg anwenden. Werfen wir einen Blick darauf, wie. In diesem Video konzentrieren wir uns darauf, wie man eine gute Wahl der Parameter w und b findet. Wenn Sie dem Modell anschließend eine neue Eingabe x geben, beispielsweise einen neuen Patienten im Krankenhaus mit einer bestimmten Tumorgröße und einem bestimmten Alter, dann handelt es sich um eine Diagnose. Das Modell kann dann eine Vorhersage treffen oder versuchen, die Wahrscheinlichkeit abzuschätzen, dass die Bezeichnung y eins ist.



Der Durchschnitt, den Sie zur Minimierung der Kostenfunktion verwenden können, ist der Gradientenabstieg. Hier ist wieder die Kostenfunktion. Wenn Sie die Kosten j als Funktion von w und b minimieren möchten, finden Sie hier den üblichen Gradientenabstiegsalgorithmus, bei dem Sie jeden Parameter wiederholt als 0-Wert minus Alpha aktualisieren, wobei die Lernrate diesen Ableitungsterm multipliziert. Werfen wir einen Blick auf die Ableitung von j nach w_j. Dieser Begriff steht hier oben, wobei j wie üblich von eins bis n reicht, wobei n die Anzahl der Features ist. Wenn jemand die Regeln der Infinitesimalrechnung anwenden würde, könnte man zeigen, dass die Ableitung nach w_j der Kostenfunktion Kapital J gleich diesem Ausdruck hier ist, nämlich 1 über m mal der Summe von 1 bis m dieses Fehlerterms. Das ist f minus der Bezeichnung y mal x_j. Hier sind nur x I j die j-Funktion des Trainingsbeispiels i. Schauen wir uns nun auch die Ableitung von j nach dem Parameter b an. Es stellt sich heraus, dass es dieser Ausdruck hier ist. Es ist dem obigen Ausdruck ziemlich ähnlich, außer dass er nicht mit diesem x hochgestellten i tiefgestellten j am Ende multipliziert wird. Zur Erinnerung: Ähnlich wie Sie es bei der linearen Regression gesehen haben, besteht die Möglichkeit, diese Aktualisierungen durchzuführen, darin, gleichzeitige Aktualisierungen zu verwenden. Das bedeutet, dass Sie zunächst die rechte Seite für alle diese Aktualisierungen berechnen und dann alle Werte gleichzeitig überschreiben gleichzeitig links. Lassen Sie mich diese abgeleiteten Ausdrücke hier nehmen und sie in diese Begriffe hier einfügen. Dadurch erhalten Sie einen Gradientenabstieg für die logistische Regression.

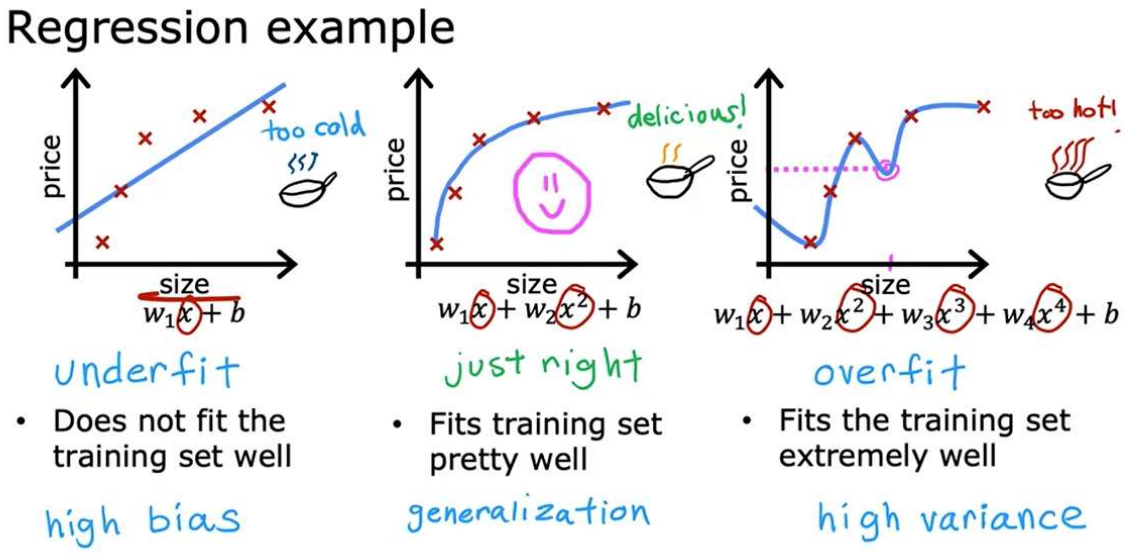


Nun, eine lustige Sache, die Sie sich vielleicht fragen, ist, dass das seltsam ist. Diese beiden Gleichungen sehen genauso aus wie der Durchschnitt, den wir zuvor für die lineare Regression ermittelt haben. Sie fragen sich also vielleicht: Ist die lineare Regression insgeheim tatsächlich dasselbe wie die logistische Regression? Nun, auch wenn diese Gleichungen gleich aussehen, liegt der Grund dafür, dass es sich nicht um eine lineare Regression handelt, darin, dass sich die Definition für die Funktion f(x) geändert hat. In der linearen Regression ist f von x, das ist wx + b. Aber in der logistischen Regression ist f(x) als die Sigmoidfunktion definiert, die auf wx + b angewendet wird. Obwohl der geschriebene Algorithmus für die lineare Regression und die logistische Regression gleich aussah, handelt es sich tatsächlich um zwei sehr unterschiedliche Algorithmen, da die Definition für f(x) nicht dieselbe ist. Als wir zuvor über den Gradientenabfall für die lineare Regression gesprochen haben, haben Sie gesehen, wie Sie einen Gradientenabfall überwachen können, um sicherzustellen, dass er konvergiert. Sie können dieselbe Methode auch für die logistische Regression anwenden, um sicherzustellen, dass sie auch konvergiert. Ich habe diese Aktualisierungen so geschrieben, als ob Sie die Parameter w_j Parameter für Parameter aktualisieren würden. Ähnlich wie bei der Diskussion über vektorisierte Implementierungen der linearen Regression können Sie die Vektorisierung auch verwenden, um den Gradientenabstieg für die logistische Regression zu beschleunigen. Ich werde in diesem Video nicht auf die Details der vektorisierten Implementierung eingehen. Sie können aber auch mehr darüber erfahren und den Code in den optionalen Labs sehen. Jetzt wissen Sie, wie Sie einen Gradientenabstieg für die logistische Regression implementieren. Vielleicht erinnern Sie sich auch an die Feature-Skalierung, als wir die lineare Regression verwendeten. Sie haben gesehen, wie die Feature-Skalierung, also die Skalierung aller Features, um ähnliche Wertebereiche anzunehmen, beispielsweise zwischen minus 1 und plus 1, dazu beitragen kann, dass der Gradientenabfall schneller konvergiert. Wenn die Feature-Skalierung auf die gleiche Weise angewendet wird, um die verschiedenen Features so zu skalieren, dass sie ähnliche Wertebereiche annehmen, kann dies auch den Gradientenabfall für die logistische Regression beschleunigen. In der kommenden optionalen Übung erfahren Sie außerdem, wie der Gradient für die logistische Regression im Code berechnet werden kann. Dies wird nützlich sein, da Sie es am Ende dieser Woche auch im Übungslabor

umsetzen werden. Nachdem Sie in diesem Labor den Gradientenabstieg ausgeführt haben, gibt es eine Reihe schöner animierter Plots, die den Gradientenabstieg in Aktion zeigen. Sie sehen die Sigmoidfunktion, das Konturdiagramm der Kosten, das 3D-Oberflächendiagramm der Kosten und die Lernkurve oder entwickeln sich als Gradientenabstiegsläufe. Danach gibt es eine weitere optionale Übung, die kurz und bündig, aber auch sehr nützlich ist, weil sie Ihnen zeigt, wie Sie die beliebte Scikit-Learn-Bibliothek verwenden, um das logistische Regressionsmodell für die Klassifizierung zu trainieren. Viele Praktiker des maschinellen Lernens in vielen Unternehmen nutzen Scikit-Learn heute regelmäßig als Teil ihrer Arbeit. Ich hoffe, dass Sie sich auch die Scikit-Learn-Funktion ansehen und einen Blick darauf werfen, wie diese verwendet wird. Das ist es. Sie sollten jetzt wissen, wie Sie die logistische Regression implementieren. Dies ist ein sehr leistungsfähiger und sehr weit verbreiteter Lernalgorithmus, und Sie wissen jetzt, wie Sie ihn selbst zum Laufen bringen können. Glückwunsch.

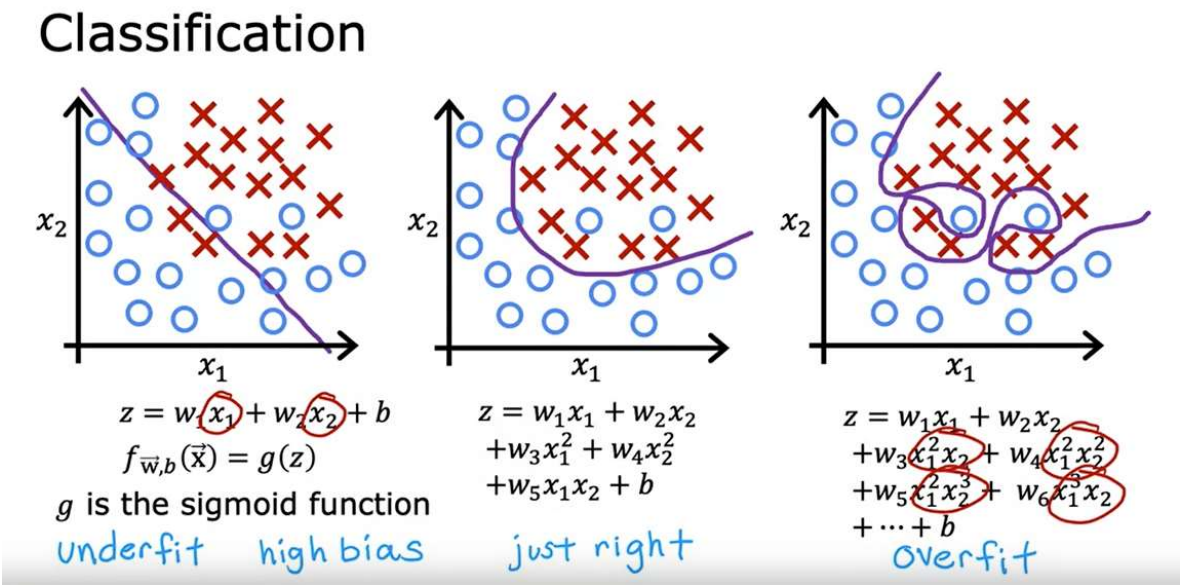
Das Problem der Überanpassung

Jetzt haben Sie einige verschiedene Lernalgorithmen gesehen, die lineare Regression und die logistische Regression. Sie eignen sich gut für viele Aufgaben. Aber manchmal kann der Algorithmus in einer Anwendung auf ein Problem namens Überanpassung stoßen, das zu einer schlechten Leistung führen kann. In diesem Video möchte ich Ihnen gerne zeigen, was Überanpassung ist, sowie ein eng damit verbundenes, fast entgegengesetztes Problem namens Unteranpassung. In den nächsten Videos werde ich Ihnen einige Techniken zur Genauigkeitsüberanpassung vorstellen. Insbesondere gibt es eine Methode namens Regularisierung. Sehr nützliche Technik. Ich benutze es die ganze Zeit. Dann hilft Ihnen die Regularisierung dabei, dieses Überanpassungsproblem zu minimieren und Ihre Lernalgorithmen viel besser funktionieren zu lassen. Werfen wir einen Blick darauf, was Überanpassung ist. Um uns zu helfen zu verstehen, was Überanpassung ist. Schauen wir uns ein paar Beispiele an. Kehren wir zu unserem ursprünglichen Beispiel der Vorhersage von Immobilienpreisen mit linearer Regression zurück. Wo Sie den Preis als Funktion der Größe eines Hauses vorhersagen möchten. Um zu verstehen, was Überanpassung ist, schauen wir uns ein Beispiel einer linearen Regression an.



Ich werde auf unser ursprüngliches laufendes Beispiel der Vorhersage von Immobilienpreisen mit linearer Regression zurückkommen. Angenommen, Ihr Datensatz sieht so aus, wobei das Eingabemerkmal x die Größe des Hauses ist und der Wert y, mit dem Sie den Preis des Hauses vorhersagen möchten. Sie könnten beispielsweise eine lineare Funktion an diese Daten anpassen. Wenn Sie das tun, erhalten Sie eine gerade Linienanpassung an die Daten, die möglicherweise so aussieht. Aber das ist kein sehr gutes Modell. Wenn man sich die Daten ansieht, scheint es ziemlich klar zu sein, dass der Wohnprozess mit zunehmender Größe des Hauses abflachte. Dieser Algorithmus passt nicht sehr gut zu den Trainingsdaten. Der Fachausdruck dafür ist, dass das Modell nicht ausreichend mit den Trainingsdaten übereinstimmt. Ein anderer Begriff ist, dass der Algorithmus eine hohe Voreingenommenheit aufweist. Möglicherweise haben Sie in den Nachrichten gelesen, dass einige Lernalgorithmen leider tatsächlich eine Voreingenommenheit gegenüber bestimmten Ethnien oder bestimmten Geschlechtern aufweisen. Beim maschinellen Lernen hat der Begriff Voreingenommenheit mehrere Bedeutungen. Die Überprüfung von Lernalgorithmen auf Verzerrungen aufgrund von Merkmalen wie Geschlecht oder ethnischer Zugehörigkeit ist absolut entscheidend. Aber der Begriff „Bias“ hat auch eine zweite technische Bedeutung, die ich hier verwende, nämlich wenn der Algorithmus nicht ausreichend an die Daten angepasst ist, was bedeutet, dass er einfach nicht einmal in der Lage ist, den Trainingssatz so gut anzupassen. Es gibt ein klares Muster in den Trainingsdaten, das der Algorithmus einfach nicht erfassen kann. Eine andere Möglichkeit, sich diese Form der Verzerrung vorzustellen, besteht darin, dass der Lernalgorithmus eine sehr starke Voreingenommenheit hat, oder wir sagen eine sehr starke Verzerrung, dass die Immobilienpreise trotz gegenteiliger Daten eine völlig lineare Funktion der Größe sein werden. Dieses Vorurteil, dass die Daten linear sind, führt dazu, dass sie an eine gerade Linie angepasst werden, die schlecht zu den Daten passt, was zu unzureichend angepassten Daten führt. Schauen wir uns nun eine zweite Variante eines Modells an: Wenn Sie für eine quadratische Funktion die Daten mit zwei Merkmalen x und x^2 einfügen, können Sie durch Anpassen der Parameter W1 und W2 eine Kurve erhalten passt etwas besser zu den Daten. Vielleicht sieht es so aus. Auch wenn Sie ein neues Haus kaufen würden, ist das nicht in diesem Satz von fünf Trainingsbeispielen enthalten. Dieses Modell

würde in diesem neuen Haus wahrscheinlich ganz gut funktionieren. Wenn Sie Immobilienmakler sind, nennen Sie die Idee, dass Ihr Lernalgorithmus auch bei Beispielen, die nicht im Trainingssatz enthalten sind, gut funktionieren soll, Generalisierung. Technisch gesehen sagen wir, dass Sie möchten, dass Ihr Lernalgorithmus gut generalisiert, was bedeutet, dass er selbst bei brandneuen Beispielen, die er noch nie zuvor gesehen hat, gute Vorhersagen trifft.



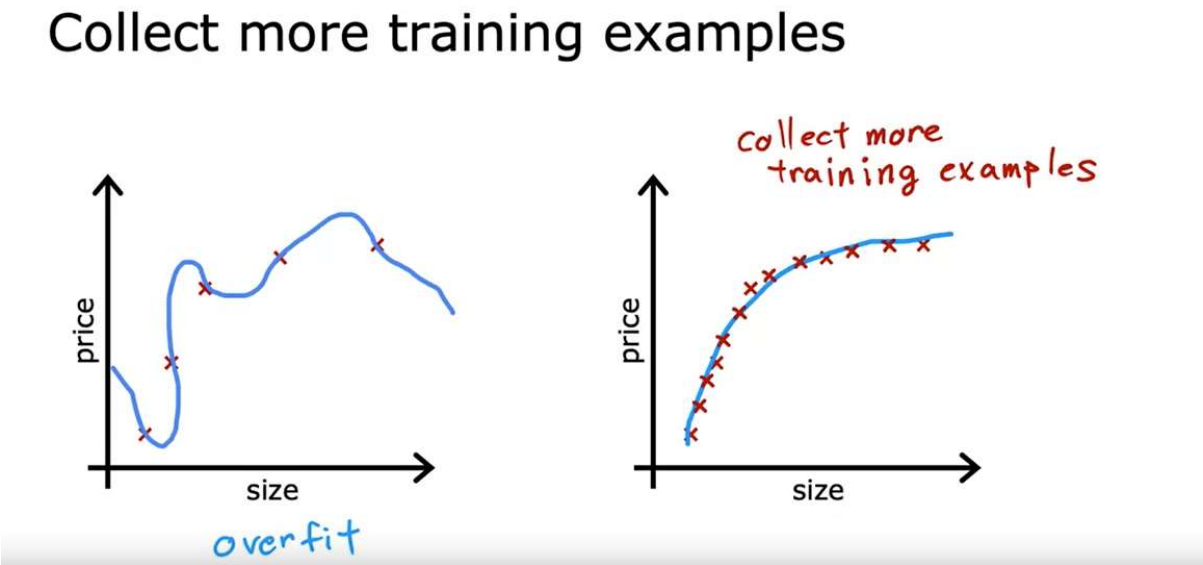
Diese quadratischen Modelle scheinen nicht perfekt, aber ziemlich gut zum Trainingssatz zu passen. Ich denke, dass es sich gut auf neue Beispiele übertragen lässt. Schauen wir uns nun das andere Extrem an. Was wäre, wenn Sie ein Polynom vierter Ordnung an die Daten anpassen würden? Sie haben x, x^2, x^3 und x^4 alle als Features. Mit diesem vierten für das Polynom können Sie die Kurve, die durch alle fünf Trainingsbeispiele verläuft, tatsächlich genau anpassen. Möglicherweise erhalten Sie eine Kurve, die so aussieht. Einerseits scheint die Anpassung der Trainingsdaten äußerst gut zu funktionieren, da alle Trainingsdaten perfekt verarbeitet werden. Tatsächlich könnten Sie Parameter auswählen, die dazu führen, dass die Kostenfunktion genau gleich Null ist, da die Fehler in allen fünf Trainingsbeispielen Null sind. Aber das ist eine sehr schlängelnde Kurve, die überall auf und ab geht. Wenn Sie diese ganze Größe hier haben, würde das Modell vorhersagen, dass dieses Haus billiger ist als Häuser, die kleiner sind. Wir glauben nicht, dass dies ein besonders gutes Modell zur Vorhersage von Immobilienpreisen ist. Der Fachausdruck lautet, dass wir sagen, dass dieses Modell eine Überanpassung an die Daten vorgenommen hat oder dass bei diesem Modell ein Überanpassungsproblem vorliegt. Denn obwohl es sehr gut zum Trainingssatz passt, hat es fast zu gut zu den Daten gepasst und ist daher überpasst. Es sieht nicht so aus, als ob sich dieses Modell auf neue, noch nie dagewesene Beispiele verallgemeinern lässt. Ein anderer Begriff dafür ist, dass der Algorithmus eine hohe Varianz aufweist. Beim maschinellen Lernen verwenden viele Menschen die Begriffe „Überanpassung“ und „Hohe Varianz“ fast synonym. Wir werden die Begriffe „Unteranpassung“ und „Hoher Bias“ fast synonym verwenden. Die Intuition hinter Überanpassung oder hoher Varianz liegt darin, dass der Algorithmus sich sehr bemüht, jedes einzelne Trainingsbeispiel anzupassen. Es stellt sich heraus, dass die Funktion, die der Algorithmus anpasst, am Ende völlig anders sein könnte, wenn Ihr Trainingssatz auch nur ein bisschen anders wäre, sagen wir, der Preis für ein Loch wäre nur

ein bisschen höher oder ein bisschen niedriger. Wenn zwei verschiedene Ingenieure für maschinelles Lernen dieses Polynommodell vierter Ordnung an nur geringfügig unterschiedliche Datensätze anpassen würden, könnten sie nicht zu völlig unterschiedlichen Vorhersagen oder sehr variablen Vorhersagen kommen. Deshalb sagen wir, dass der Algorithmus eine hohe Varianz aufweist. Wenn man dieses Modell ganz rechts mit dem Modell in der Mitte für dasselbe Haus vergleicht, scheint es, dass das mittlere Modell eine viel vernünftigere Preisvorhersage liefert. Es gibt keinen wirklichen Namen für diesen Fall in der Mitte, aber ich nenne ihn einfach genau richtig, weil er weder „Underfit“ noch „Overfit“ ist. Man kann sagen, dass das Ziel des maschinellen Lernens darin besteht, ein Modell zu finden, das hoffentlich weder unter- noch überangepasst ist. Mit anderen Worten, hoffentlich ein Modell, das weder eine hohe Verzerrung noch eine hohe Varianz aufweist. Wenn ich an Unter- und Überanpassung, hohe Verzerrung und hohe Varianz denke. Manchmal erinnere ich mich an die Kindergeschichte von Goldlöckchen und den drei Bären. In diesem Kindermärchen besucht ein Mädchen namens Goldlöckchen das Haus einer Bärenfamilie. Es gibt eine Schüssel Haferbrei, die zu kalt ist, um sie zu schmecken, und daher nicht gut ist. Es gibt auch eine Schüssel Haferbrei, die zum Essen zu heiß ist. Das ist auch nicht gut. Aber es gibt eine Schüssel Porridge, die weder zu kalt noch zu heiß ist. Die Temperatur liegt im mittleren Bereich, was genau richtig zum Essen ist. Um es noch einmal zusammenzufassen: Wenn Sie zu viele Funktionen wie das Polynom vierter Ordnung auf der rechten Seite haben, passt das Modell möglicherweise gut zum Trainingssatz, aber fast zu gut oder überpasst und weist eine hohe Varianz auf. Wenn Sie hingegen über zu wenige Funktionen verfügen, ist die Anpassung in diesem Beispiel, wie auch im linken, unzureichend und weist eine hohe Voreingenommenheit auf. In diesem Beispiel scheint die Verwendung quadratischer Merkmale x und x im Quadrat genau richtig zu sein. Bisher haben wir uns mit Unteranpassung und Überanpassung für das lineare Regressionsmodell befasst. In ähnlicher Weise wird auch bei der Überanpassung eine Klassifizierung angewendet. Hier ist ein Klassifizierungsbeispiel mit zwei Merkmalen, x_1 und x_2, wobei x_1 möglicherweise die Tumorgröße und x_2 das Alter des Patienten ist. Wir versuchen zu klassifizieren, ob ein Tumor bösartig oder gutartig ist, wie durch diese Kreuze und Kreise angezeigt. Sie könnten beispielsweise ein logistisches Regressionsmodell anpassen. Nur ein einfaches Modell wie dieses, bei dem g wie üblich die Sigmoidfunktion ist und dieser Term hier drinnen z ist. Wenn Sie das tun, erhalten Sie am Ende eine gerade Linie als Entscheidungsgrenze. Dies ist die Linie, bei der z gleich Null ist und die die positiven und negativen Beispiele trennt. Diese gerade Linie sieht nicht schrecklich aus. Es sieht in Ordnung aus, aber es scheint auch nicht sehr gut zu den Daten zu passen. Dies ist ein Beispiel für eine Unteranpassung oder einen hohen Bias. Schauen wir uns ein anderes Beispiel an. Wenn Sie diese quadratischen Terme zu Ihren Features hinzufügen, wird z zu diesem neuen Term in der Mitte und die Entscheidungsgrenze, d. h. wenn z gleich Null ist, kann eher so aussehen, eher wie eine Ellipse oder ein Teil einer Ellipse. Dies passt ziemlich gut zu den Daten, auch wenn nicht jedes einzelne Trainingsbeispiel im Trainingssatz perfekt klassifiziert wird. Beachten Sie, wie einige dieser Kreuze den Kreisen zugeordnet werden. Aber dieses Modell sieht ziemlich gut aus. Ich werde es genau richtig nennen. Es sieht so aus, als ob sich dies ziemlich gut auf neue Patienten übertragen lässt. Im anderen Extremfall schließlich: Wenn Sie ein Polynom sehr hoher Ordnung mit vielen Merkmalen wie diesen anpassen würden, würde sich das Modell möglicherweise sehr anstrengen und sich selbst konturieren oder verdrehen, um eine

Entscheidungsgrenze zu finden, die perfekt zu Ihren Trainingsdaten passt. Mit all diesen Polynommerkmalen höherer Ordnung kann der Algorithmus dies tatsächlich über die komplexe Entscheidungsgrenze hinweg auswählen. Wenn es sich bei den Merkmalen um die Größe des Tumors nach Alter handelt und Sie versuchen, Tumore als bösartig oder gutartig zu klassifizieren, dann scheint dies kein wirklich gutes Modell für Vorhersagen zu sein. Dies ist wiederum ein Fall von Überanpassung und hoher Varianz, da das Modell, obwohl es im Trainingssatz sehr gut abschneidet, nicht so aussieht, als ob es sich gut auf neue Beispiele übertragen lässt. Jetzt haben Sie gesehen, wie ein Algorithmus eine Unteranpassung oder eine hohe Verzerrung oder eine Überanpassung und eine hohe Varianz aufweisen kann. Vielleicht möchten Sie wissen, wie Sie ein Modell erhalten, das genau richtig ist. Im nächsten Video sehen wir uns einige Möglichkeiten an, wie Sie das Problem der Überanpassung angehen können. Wir gehen auch auf einige Ideen ein, die für die Verwendung von Underfitting relevant sind. Kommen wir zum nächsten Video.

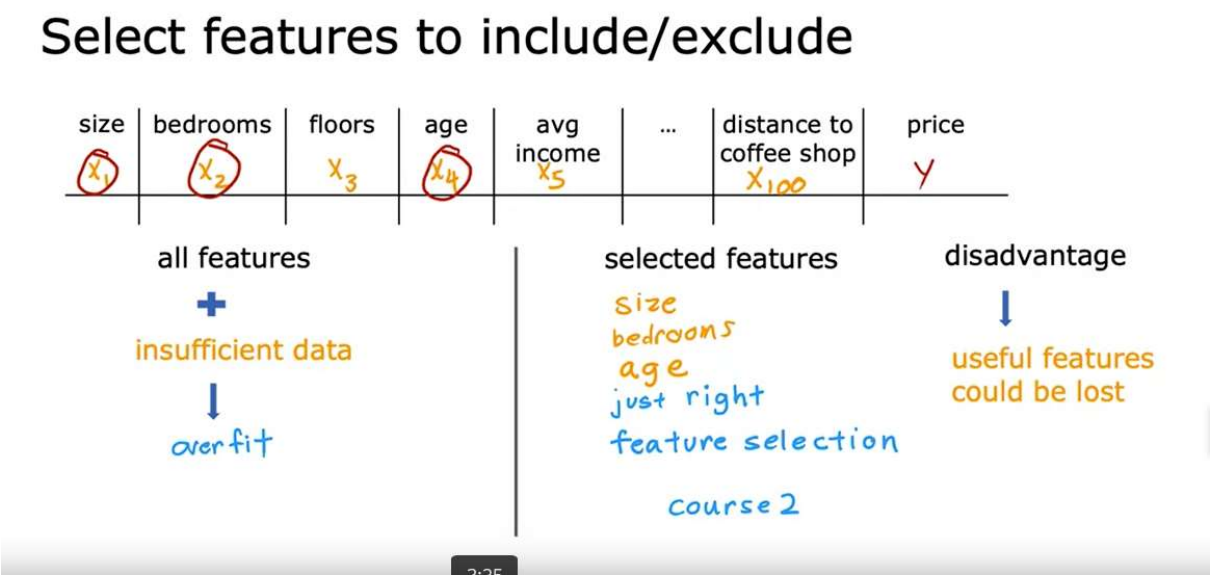
Überanpassung angehen

Später in dieser Spezialisierung werden wir über das Debuggen und Diagnostizieren von Dingen sprechen, die bei Lernalgorithmen schief gehen können. Außerdem lernen Sie spezielle Tools kennen, mit denen Sie erkennen können, wann eine Über- oder Unteranpassung auftreten kann. Aber wenn Sie glauben, dass eine Überanpassung vorliegt, sprechen wir zunächst darüber, was Sie tun können, um dagegen vorzugehen. Nehmen wir an, Sie passen ein Modell an und es weist eine hohe Varianz auf, ist also überangepasst.



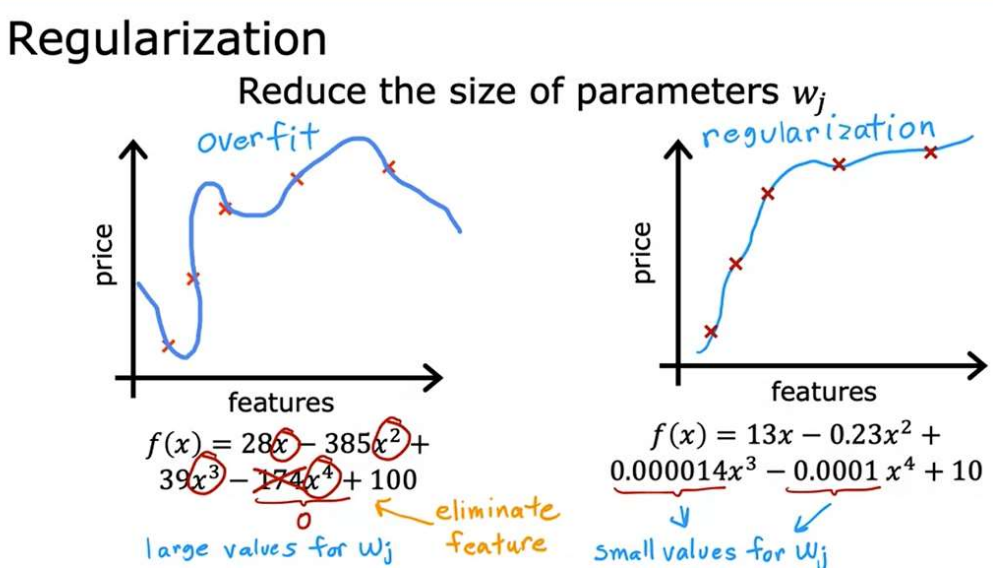
Hier ist unser Overfit-Hauspreisvorhersagemodell. Eine Möglichkeit, dieses Problem anzugehen, besteht darin, mehr Trainingsdaten zu sammeln, das ist eine Option. Wenn Sie in der Lage sind, mehr Daten zu erhalten, das heißt mehr Trainingsbeispiele zu Größen und Preisen von Häusern, dann lernt der Lernalgorithmus mit dem größeren Trainingssatz, eine Funktion anzupassen, die weniger wackelig ist. Sie können weiterhin ein Polynom höherer Ordnung oder einen Teil der Funktion mit vielen Funktionen anpassen, und wenn Sie über genügend Trainingsbeispiele verfügen, funktioniert es immer noch einwandfrei. Zusammenfassend lässt sich sagen, dass das wichtigste Mittel gegen Überanpassung darin

besteht, mehr Trainingsdaten zu erhalten. Nun ist es nicht immer eine Option, mehr Daten zu erhalten. Möglicherweise wurden an diesem Standort nur eine begrenzte Anzahl von Häusern verkauft, sodass möglicherweise keine weiteren Daten hinzugefügt werden können. Aber wenn die Daten vorliegen, kann das wirklich gut funktionieren. Eine zweite Möglichkeit, der Überanpassung zu begegnen, besteht darin, zu prüfen, ob Sie weniger Funktionen verwenden können. Im vorherigen Video umfassten die Funktionen unserer Modelle die Größe x sowie die Größe im Quadrat, und zwar x im Quadrat, x in Würfelform und x^4 und so weiter. Das waren viele Polynommerkmale. In diesem Fall besteht eine Möglichkeit zur Reduzierung der Überanpassung darin, einfach nicht so viele dieser Polynommerkmale zu verwenden. Aber schauen wir uns nun ein anderes Beispiel an. Vielleicht haben Sie viele verschiedene Merkmale eines Hauses, deren Preis Sie vorhersagen möchten, angefangen bei der Größe, der Anzahl der Schlafzimmer, der Anzahl der Stockwerke, dem Alter, dem durchschnittlichen Einkommen der Nachbarschaft usw. bis hin zum Gesamtwert Entfernung zum nächsten Café. Es stellt sich heraus, dass Ihr Lernalgorithmus möglicherweise auch zu stark an Ihren Trainingssatz angepasst ist, wenn Sie viele Funktionen wie diese, aber nicht genügend Trainingsdaten haben . Anstatt alle 100 Funktionen zu nutzen, würden wir nur eine Teilmenge der nützlichsten auswählen, vielleicht Größe, Schlafzimmer und das Alter des Hauses.

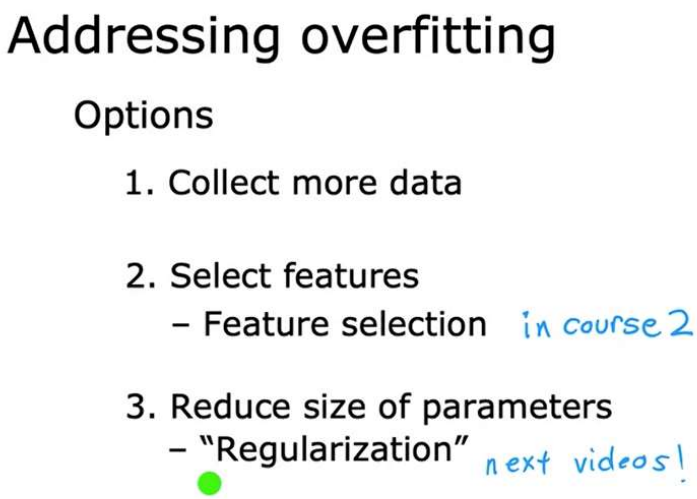


Wenn Sie der Meinung sind, dass dies die relevantesten Funktionen sind, werden Sie möglicherweise feststellen, dass Ihr Modell nicht mehr so stark überpasst, wenn Sie nur diese kleinste Teilmenge von Funktionen verwenden. Die Auswahl des am besten geeigneten Funktionssatzes wird manchmal auch als Funktionsauswahl bezeichnet. Eine Möglichkeit, dies zu tun, besteht darin, Ihre Intuition zu nutzen und auszuwählen, was Ihrer Meinung nach die besten Funktionen sind, die für die Vorhersage des Preises am relevantesten sind. Ein Nachteil der Feature-Auswahl besteht darin, dass der Algorithmus durch die Verwendung nur einer Teilmenge der Features einige der Informationen, die Sie über die Häuser haben, verwirft. Vielleicht sind zum Beispiel alle diese Funktionen, alle 100 davon, tatsächlich nützlich, um den Preis eines Hauses vorherzusagen. Vielleicht möchten Sie nicht einige Informationen wegwerfen, indem Sie einige Funktionen wegwerfen. Später in Kurs 2 werden

Sie auch einige Algorithmen sehen, mit denen Sie automatisch den am besten geeigneten Funktionssatz für unsere Vorhersageaufgabe auswählen. Damit kommen wir nun zur dritten Möglichkeit zur Reduzierung der Überanpassung. Diese Technik, die wir im nächsten Video noch genauer betrachten werden, nennt sich Regularisierung. Wenn Sie sich ein Overfit-Modell ansehen, sehen Sie hier ein Modell, das Polynomfunktionen verwendet: x, x im Quadrat, x kubisch und so weiter. Sie stellen fest, dass die Parameter oft relativ groß sind. Wenn Sie nun einige dieser Features eliminieren würden, beispielsweise wenn Sie das Feature x4 eliminieren würden, entspricht das dem Setzen dieses Parameters auf 0. Das Setzen eines Parameters auf 0 ist also gleichbedeutend mit dem Eliminieren eines Features, was wir gesehen haben die vorherige Folie. Es stellt sich heraus, dass die Regularisierung eine Möglichkeit ist, die Auswirkungen einiger Funktionen sanfter zu reduzieren, ohne sie ganz zu eliminieren. Durch die Regularisierung wird der Lernalgorithmus dazu angeregt, die Werte der Parameter zu verkleinern, ohne unbedingt zu verlangen, dass der Parameter genau auf 0 gesetzt wird. Es stellt sich heraus, dass selbst wenn Sie ein Polynom höherer Ordnung wie dieses anpassen, solange Sie den Algorithmus erhalten um kleinere Parameterwerte zu verwenden: w1, w2, w3, w4. Am Ende erhalten Sie eine Kurve, die viel besser zu den Trainingsdaten passt. Was die Regularisierung also bewirkt, ist, dass Sie alle Ihre Features behalten können, aber sie verhindert lediglich, dass die Features einen übermäßig großen Effekt haben, was manchmal zu einer Überanpassung führen kann. Übrigens reduzieren wir vereinbarungsgemäß normalerweise einfach die Größe der wj-Parameter, also w1 bis wn. Es macht keinen großen Unterschied, ob Sie auch den Parameter b regulieren. Sie können dies tun, wenn Sie möchten, oder nicht, wenn Sie dies nicht tun. Normalerweise tue ich das nicht und es ist völlig in Ordnung, w1, w2 bis hin zu wn zu regulieren, aber b nicht wirklich dazu zu ermutigen, kleiner zu werden. In der Praxis sollte es kaum einen Unterschied machen, ob Sie auch b regulieren oder nicht. Um es noch einmal zusammenzufassen: Dies sind die drei Möglichkeiten, die Sie in diesem Video gesehen haben, um der Überanpassung zu begegnen. Erstens: Sammeln Sie mehr Daten. Wenn Sie mehr Daten erhalten, kann dies wirklich dazu beitragen, die Überanpassung zu reduzieren. Manchmal ist das nicht möglich. In diesem Fall gibt es einige der Optionen: Zweitens: Versuchen Sie, nur eine Teilmenge der Funktionen auszuwählen und zu verwenden. In Kurs 2 erfahren Sie mehr über die Feature-Auswahl. Die dritte Möglichkeit besteht darin, die Größe der Parameter mithilfe der Regularisierung zu reduzieren. Dies wird auch Thema des nächsten Videos sein. Nur für mich selbst verwende ich ständig die Regularisierung.



Dies ist also eine sehr nützliche Technik zum Trainieren von Lernalgorithmen, insbesondere neuronalen Netzen, was Sie später in dieser Spezialisierung ebenfalls sehen werden. Ich hoffe, dass Sie sich auch das optionale Labor zum Thema Überanpassung ansehen. Im Labor können Sie verschiedene Beispiele für Überanpassung sehen und diese Beispiele anpassen, indem Sie in den Diagrammen auf Optionen klicken. Sie können auch Ihre eigenen Datenpunkte hinzufügen, indem Sie auf das Diagramm klicken und sehen, wie sich dadurch die angepasste Kurve ändert. Sie können auch Beispiele für Regression und Klassifizierung ausprobieren und den Grad des Polynoms in x, x quadriert, x kubisch usw. ändern. Im Labor können Sie außerdem mit zwei verschiedenen Optionen zur Behebung von Überanpassungen experimentieren. Sie können zusätzliche Trainingsdaten hinzufügen, um die Überanpassung zu reduzieren, und Sie können auch auswählen, welche Features ein- oder ausgeschlossen werden sollen, um eine weitere Möglichkeit zu finden, die Überanpassung zu reduzieren.

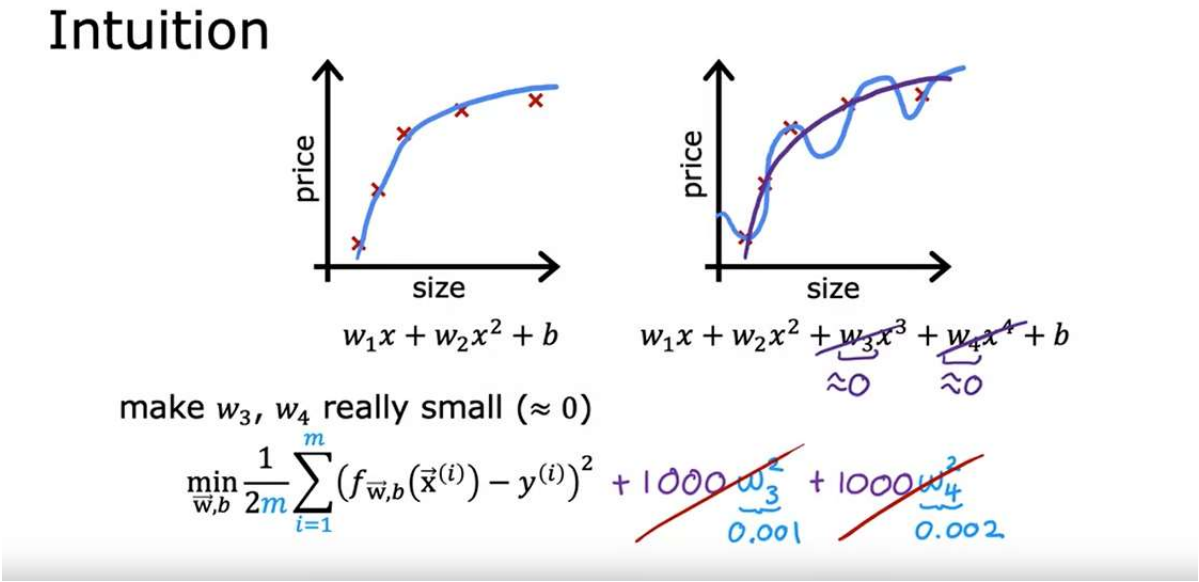


Bitte schauen Sie sich ein Labor an, das Ihnen hoffentlich dabei helfen wird, Ihr Gespür für Überanpassung zu entwickeln und einige Methoden zu finden, wie Sie dagegen vorgehen können. In diesem Video haben Sie auch die Idee der Regularisierung auf einem relativ

hohen Niveau gesehen. Ich bin mir darüber im Klaren, dass all diese Details zur Regularisierung für Sie möglicherweise noch nicht vollständig nachvollziehbar sind. Aber im nächsten Video beginnen wir damit, genau zu formulieren, wie die Regularisierung angewendet wird und was Regularisierung genau bedeutet. Dann beginnen wir herauszufinden, wie wir dies mit unseren Lernalgorithmen zum Laufen bringen können, um eine lineare Regression und eine logistische Regression durchzuführen , und in Zukunft werden auch andere Algorithmen eine Überanpassung vermeiden. Schauen wir uns das im nächsten Video an.

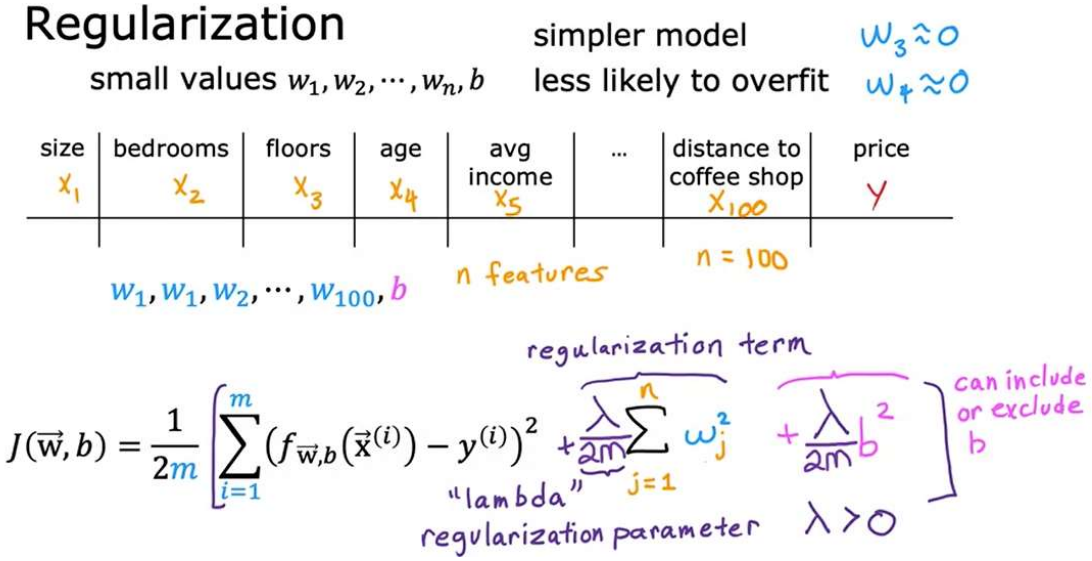
Kostenfunktion mit Regularisierung

Im letzten Video haben wir gesehen, dass die Regularisierung versucht, die Elternwerte W1 bis WN klein zu machen, um eine Überanpassung zu reduzieren. In diesem Video bauen wir auf dieser Intuition auf und entwickeln eine modifizierte Kostenfunktion für Ihren Lernalgorithmus, mit der Sie die Regularisierung tatsächlich anwenden können. Lassen Sie uns einsteigen und sich an dieses Beispiel aus dem vorherigen Video erinnern, in dem wir gesehen haben, dass die Anpassung einer quadratischen Funktion an diese Daten eine ziemlich gute Anpassung ergibt. Wenn Sie jedoch ein Polynom sehr hoher Ordnung anpassen, erhalten Sie eine Kurve, die den Daten besser entspricht. Aber bedenken Sie nun Folgendes: Nehmen wir an, Sie hätten die Möglichkeit, die Parameter W3 und W4 wirklich, wirklich klein zu machen.



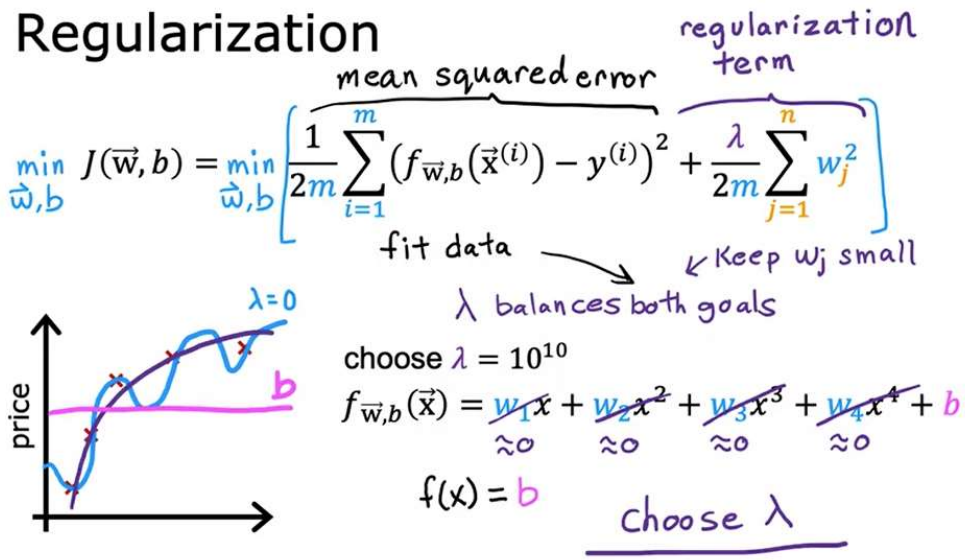
Sagen wir nahe 0. Hier ist, was ich meine. Anstatt diese Zielfunktion zu minimieren, handelt es sich hierbei um eine Kostenfunktion für die lineare Regression. Nehmen wir an, Sie ändern die Kostenfunktion und addieren dazu 1000 Mal W3 zum Quadrat plus 1000 Mal W4 zum Quadrat. Und hier wähle ich einfach 1000, weil es eine große Zahl ist, aber jede andere wirklich große Zahl wäre in Ordnung. Mit dieser modifizierten Kostenfunktion könnten Sie also tatsächlich das Modell benachteiligen, wenn W3 und W4 groß sind. Denn wenn Sie diese Funktion minimieren möchten, besteht die einzige Möglichkeit, diese neue Kostenfunktion klein zu machen, darin, dass W3 und W4 beide klein sind, oder? Denn sonst

werden diese 1000-mal W3-Quadrat- und 1000-mal W4-Quadrat-Terme wirklich sehr, sehr groß. Wenn Sie diese Funktion also minimieren, wird W3 nahe bei 0 und W4 nahe bei 0 liegen. Wir heben also praktisch die Auswirkungen der Funktionen „Execute“ und „Extra Power of 4“ auf und beseitigen diese beiden Begriffe hier. Und wenn wir das tun, erhalten wir am Ende eine Anpassung an die Daten, die viel näher an der quadratischen Funktion liegt, einschließlich vielleicht nur winziger Beiträge der Merkmale x kubisch und zusätzlich 4. Und das ist gut, weil es viel besser an die Funktion angepasst ist Daten im Vergleich dazu, ob alle Parameter groß sein könnten und Sie am Ende diese wöchentliche quadratische Funktion allgemeiner erhalten würden. Hier ist die Idee hinter der Regularisierung. Die Idee dahinter ist: Wenn es kleinere Werte für die Parameter gibt, ist das ein bisschen so, als hätte man ein einfacheres Modell. Vielleicht eines mit weniger Funktionen, das daher weniger anfällig für Überanpassung ist. Auf der letzten Folie bestrafen wir oder wir sagen, wir haben nur W3 und W4 reguliert.



Aber allgemeiner gesagt wird die Regularisierung in der Regel so implementiert, dass Sie bei vielen Funktionen, beispielsweise 100 Funktionen, möglicherweise nicht wissen, welche Funktionen am wichtigsten sind und welche bestraft werden müssen. Die Art und Weise, wie die Regularisierung typischerweise implementiert wird, besteht darin, alle Funktionen zu bestrafen, oder genauer gesagt, alle WJ-Parameter zu bestrafen, und es lässt sich zeigen, dass dies normalerweise dazu führt, dass eine glattere, einfachere, weniger wöchentliche Funktion angepasst wird, die weniger anfällig für Überanpassung ist. Wenn Sie also in diesem Beispiel Daten mit 100 Merkmalen für jedes Haus haben, kann es schwierig sein, vorab festzulegen, welche Merkmale einbezogen und welche ausgeschlossen werden sollen. Erstellen wir also ein Modell, das alle 100 Funktionen nutzt. Sie haben also diese 100 Parameter W1 bis W100 sowie 100 und den ersten Parameter B. Weil wir nicht wissen, welche dieser Parameter wichtig sein werden. Lassen Sie uns alle ein wenig bestrafen und alle verkleinern, indem wir diesen neuen Term Lambda multipliziert mit der Summe von J gleich 1 bis n hinzufügen, wobei n 100 ist. Die Anzahl der Merkmale von wj im Quadrat. Dieser Wert Lambda ist hier das griechische Alphabet Lambda und wird auch als Regularisierungsparameter bezeichnet. Ähnlich wie bei der Auswahl einer Lernrate Alpha

müssen Sie jetzt auch eine Zahl für Lambda auswählen. Auf ein paar Dinge möchte ich konventionell hinweisen, anstatt Lambda mal die Summe von wj im Quadrat zu verwenden. Wir teilen Lambda auch durch 2 m, sodass sowohl der 1. als auch der 2. Term hier über 2 m mit 1 skaliert werden. Es stellt sich heraus, dass es durch die gleiche Skalierung beider Terme etwas einfacher wird, einen guten Wert für Lambda auszuwählen. Und insbesondere stellen Sie fest, dass Sie mehr Trainingsbeispiele finden, auch wenn die Größe Ihres Trainingssatzes zunimmt. Daher ist die Größe des Trainingssatzes jetzt größer. Derselbe Lambda-Wert, den Sie zuvor ausgewählt haben, funktioniert jetzt auch mit größerer Wahrscheinlichkeit weiterhin, wenn Sie diese zusätzliche Skalierung um 2 m haben. Übrigens werden wir den Parameter b vereinbarungsgemäß nicht dafür bestrafen, dass er groß ist. In der Praxis macht es kaum einen Unterschied, ob Sie dies tun oder nicht. Und einige Ingenieure für maschinelles Lernen und tatsächlich einige Implementierungen von Lernalgorithmen werden auch Lambda über das 2-Millionen-fache des b-Quadrat-Terms einbeziehen. In der Praxis macht dies jedoch kaum einen Unterschied, und die in diesem Kurs üblichere Konvention besteht darin, nur die Parameter w und nicht den Parameter b zu regulieren. Zusammenfassend wollen wir in dieser modifizierten Kostenfunktion die ursprünglichen Kosten minimieren, also die mittleren quadratischen Fehlerkosten plus zusätzlich den zweiten Term, der als Regularisierungsterm bezeichnet wird.

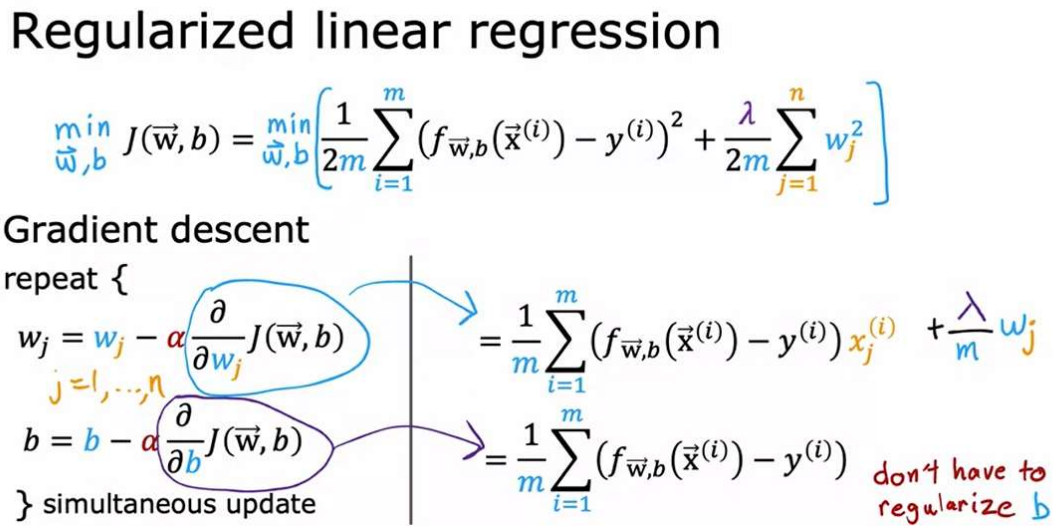


Und so gleicht diese neue Kostenfunktion zwei Ziele aus, die Sie möglicherweise haben. Der Versuch, diesen ersten Term zu minimieren, fördert die gute Anpassung des Algorithmus an die Trainingsdaten, indem die quadrierten Unterschiede zwischen den Vorhersagen und den tatsächlichen Werten minimiert werden. Und versuchen Sie, den zweiten Begriff zu minimieren. Der Algorithmus versucht außerdem, die Parameter wj klein zu halten, wodurch die Überanpassung tendenziell verringert wird. Der von Ihnen gewählte Lambda-Wert gibt die relative Bedeutung oder den relativen Kompromiss an oder wie Sie zwischen diesen beiden Zielen balancieren. Werfen wir einen Blick darauf, welche unterschiedlichen Lambda-Werte Ihren Lernalgorithmus bewirken. Lassen Sie uns das Beispiel der Immobilienpreisvorhersage mithilfe der linearen Regression verwenden. f(x) ist also das lineare Regressionsmodell. Wenn Lambda auf 0 gesetzt wurde, verwenden Sie den

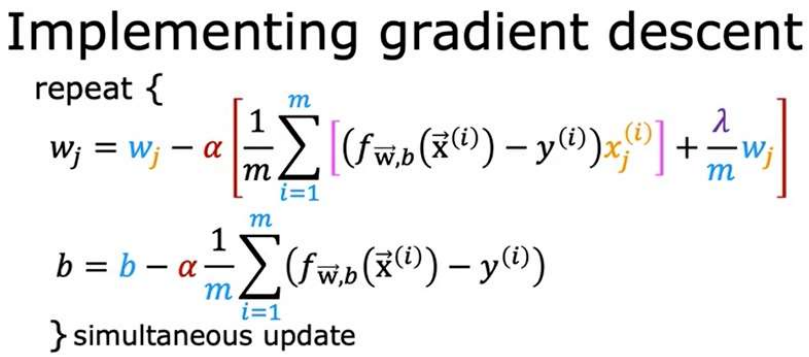
Regularisierungsterm überhaupt nicht, da der Regularisierungsterm mit 0 multipliziert wird. Wenn Lambda also 0 war, passen Sie am Ende diese übermäßig wackelige, übermäßig komplexe Kurve an und sie passt zu gut . Das war also ein Extremfall, wenn Lambda 0 wäre. Schauen wir uns nun das andere Extrem an. Wenn Sie sagen, Lambda sei eine wirklich, wirklich, wirklich große Zahl, sagen wir, Lambda sei gleich 10 hoch 10, dann legen Sie ein sehr großes Gewicht auf diesen Regularisierungsterm auf der rechten Seite. Und die einzige Möglichkeit, dies zu minimieren, besteht darin, sicherzustellen, dass alle Werte von w ziemlich nahe bei 0 liegen. Wenn Lambda also sehr, sehr groß ist, wählt der Lernalgorithmus W1, W2, W3 und W4 so, dass sie extrem nahe bei 0 liegen zu 0 und somit ist F von Um es noch einmal zusammenzufassen: Wenn Lambda 0 ist, passt dieses Modell zu gut. Wenn Lambda enorm ist, etwa 10 hoch 10, passt dieses Modell zu wenig. Was Sie also wollen, ist ein dazwischen liegender Lambda-Wert, der diese ersten und zweiten Bedingungen des Kompromisses besser ausgleicht, den mittleren quadratischen Fehler minimiert und die Parameter klein hält. Und wenn der Wert von Lambda nicht zu klein und nicht zu groß, sondern genau richtig ist, können Sie am Ende hoffentlich ein Polynom 4. Ordnung anpassen und dabei alle diese Merkmale beibehalten, aber mit einer Funktion, die so aussieht. So funktioniert also die Regularisierung. Wenn wir über die Modellauswahl sprechen, werden wir später in der Spezialisierung auch verschiedene Möglichkeiten sehen, gute Werte für Lambda auszuwählen. In den nächsten beiden Videos erfahren Sie, wie Sie die Regularisierung auf die lineare Regression und die logistische Regression anwenden und wie Sie diese Modelle mit großem Widerspruch trainieren. Damit können Sie eine Überanpassung mit beiden Algorithmen vermeiden.

Regularisierte lineare Regression

In diesem Video werden wir herausfinden, wie wir den Gradientenabstieg mit der regulierten linearen Regression zum Laufen bringen. Lasst uns einsteigen. Hier ist eine Kostenfunktion, die wir uns im letzten Video für die regulierte lineare Regression ausgedacht haben. Der erste Teil ist die übliche quadratische Fehlerkostenfunktion, und jetzt haben Sie diesen zusätzlichen Regularisierungsterm, wobei Lambda der Regularisierungsparameter ist, und Sie möchten die Parameter w und b finden, die die regulierte Kostenfunktion minimieren. Zuvor verwendeten wir den Gradientenabstieg für die ursprüngliche Kostenfunktion, nur den ersten Term, bevor wir diesen zweiten Regularisierungsterm hinzufügten, und zuvor hatten wir den folgenden Gradientenabstiegsalgorithmus, bei dem wir die Parameter w, j und b für wiederholt aktualisieren j ist gemäß dieser Formel gleich 1 bis n und b wird ebenfalls auf ähnliche Weise aktualisiert. Auch hier ist Alpha eine sehr kleine positive Zahl, die als Lernrate bezeichnet wird. Tatsächlich sehen die Aktualisierungen für eine regulierte lineare Regression genau gleich aus, außer dass die Kosten J jetzt etwas anders definiert sind. Zuvor wurde die Ableitung von J nach w_j durch diesen Ausdruck hier angegeben, und die Ableitung nach b wurde durch diesen Ausdruck hier angegeben. Nachdem wir nun diesen zusätzlichen Regularisierungsterm hinzugefügt haben, ändert sich nur noch, dass der Ausdruck für die Ableitung nach w_j am Ende einen zusätzlichen Term hat, diesen plus Lambda über m mal w_j.

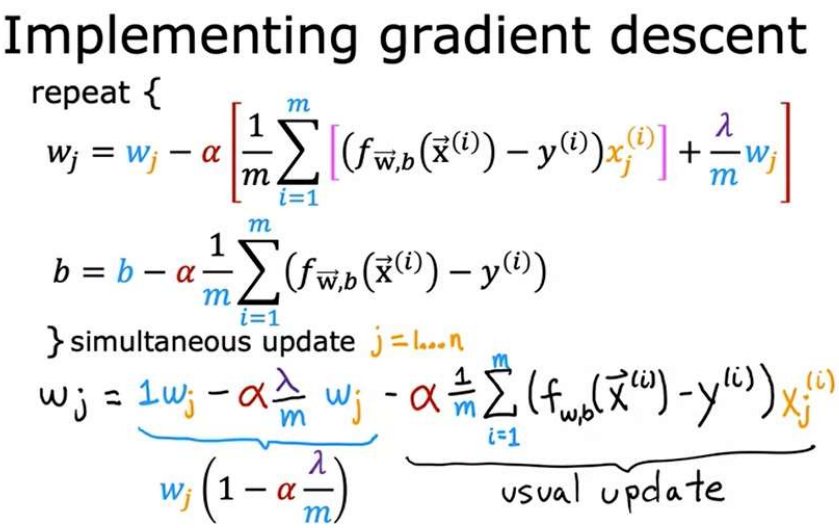


Und insbesondere für die neue Definition der Kostenfunktion j sind diese beiden Ausdrücke hier die neuen Ableitungen von J nach w_j und die Ableitung von J nach b. Denken Sie daran, dass wir b nicht regulieren, also nicht versuchen, B zu verkleinern. Aus diesem Grund bleibt das aktualisierte B dasselbe wie zuvor, während sich das aktualisierte w ändert, weil der Regularisierungsterm dazu führt, dass wir versuchen, w_j zu verkleinern. Nehmen wir diese Definitionen für die Ableitungen und fügen sie wieder in den Ausdruck links ein, um den Gradientenabstiegsalgorithmus für die regulierte lineare Regression zu schreiben. Um einen Gradientenabstieg für die regulierte lineare Regression zu implementieren, müsste Ihr Code Folgendes tun. Hier ist das Update für w_j, für j gleich 1 bis n, und hier ist das Update für b. Bitte denken Sie wie gewohnt daran, alle diese Parameter gleichzeitig zu aktualisieren. Damit dieser Algorithmus funktioniert, müssen Sie nur Folgendes wissen.



Was ich im Rest dieses Videos jedoch gerne tun möchte, ist, optionales Material durchzugehen, um eine etwas tiefere Vorstellung davon zu vermitteln, was diese Formel tatsächlich bewirkt, und außerdem kurz darüber zu plaudern, wie diese Ableitungen

abgeleitet werden. Der Rest dieses Videos ist völlig optional. Es ist völlig in Ordnung, wenn Sie den Rest dieses Videos überspringen, aber wenn Sie ein starkes Interesse an Mathematik haben, dann bleiben Sie bei mir. Es ist immer schön, hier Zeit mit Ihnen zu verbringen, und durch diese Gleichungen können Sie vielleicht auch ein tieferes Verständnis dafür entwickeln, was die Mathematik und die Ableitungen bewirken. Lass uns einen Blick darauf werfen. Schauen wir uns die Aktualisierungsregel für w_j an und schreiben sie auf andere Weise um. Wir aktualisieren w_j als 1 mal w_j minus Alpha mal Lambda über m mal w_j. Ich habe den Begriff hier vom Ende nach vorne verschoben. Dann minus Alpha mal 1 über m und dann der Rest dieses Termes dort drüben. Wir haben die Begriffe nur ein wenig umgestellt. Wenn wir vereinfachen, dann sagen wir, dass w_j aktualisiert wird als w_j mal 1 minus Alpha mal Lambda über m, minus Alpha mal dieser andere Term hier drüben. Möglicherweise erkennen Sie den zweiten Term als die übliche Gradientenabstiegsaktualisierung für die unregulierte lineare Regression. Dies ist das Update für die lineare Regression vor der Regularisierung, und das ist der Begriff, den wir in Woche 2 dieses Kurses gesehen haben. Die einzige Änderung, die wir durch die Regularisierung hinzufügen, besteht darin, dass w_j nicht mehr gleich w_j minus Alpha mal gesetzt wird, sondern dass dieser Term jetzt w mal diese Zahl minus der üblichen Aktualisierung ist. Das haben wir in Woche 1 dieses Kurses erlebt. Was ist dieser erste Begriff hier? Nun, Alpha ist eine sehr kleine positive Zahl, sagen wir 0,01. Lambda ist normalerweise eine kleine Zahl, sagen wir 1 oder vielleicht 10. Nehmen wir an, dass Lambda für dieses Beispiel 1 ist und m die Größe des Trainingssatzes ist, sagen wir 50. Wenn Sie Alpha Lambda mit m multiplizieren, sagen wir 0,01 mal 1 dividiert durch 50, ist dieser Term am Ende ist es eine kleine positive Zahl, sagen wir 0,0002, und daher wird 1 minus Alpha Lambda über m eine Zahl sein, die nur geringfügig kleiner als 1 ist, in diesem Fall 0,9998. Der Effekt dieses Begriffs besteht darin, dass Sie bei jeder einzelnen Iteration des Gradientenabstiegs w_j nehmen und es mit 0,9998 multiplizieren, d. h. mit einigen Zahlen, die etwas kleiner als eins sind, und um die übliche Aktualisierung durchzuführen.

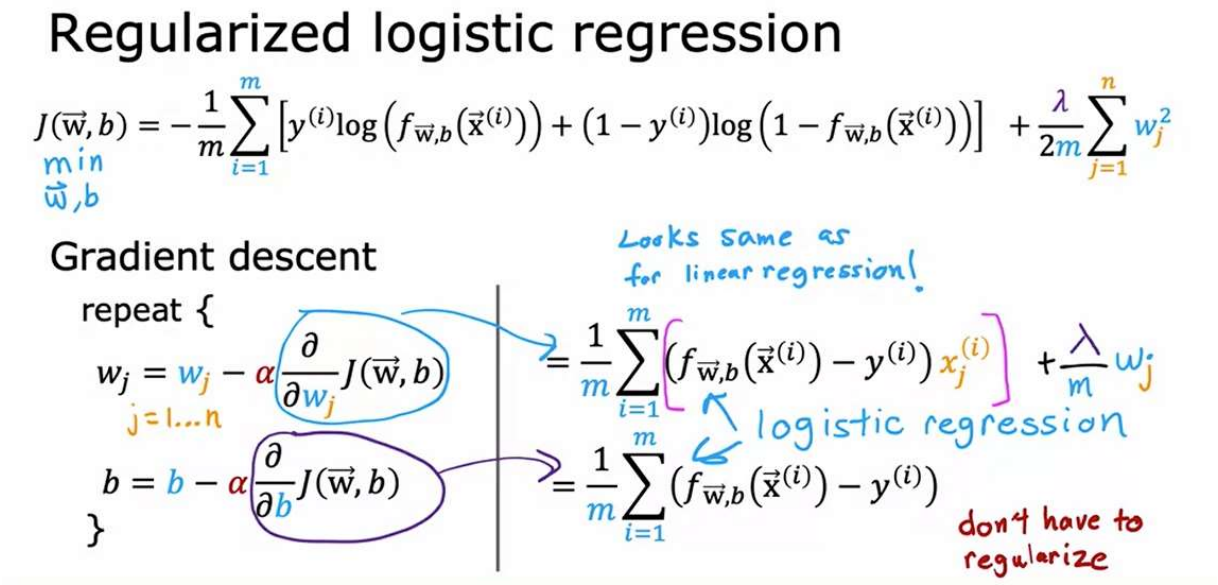


Was die Regularisierung bei jeder einzelnen Iteration bewirkt, ist, dass Sie w mit einer Zahl etwas kleiner als 1 multiplizieren, was zur Folge hat, dass der Wert von w_j nur ein wenig

schrumpft. Dies gibt uns eine andere Sicht darauf, warum die Regularisierung dazu führt, dass die Parameter w_j bei jeder Iteration ein wenig schrumpfen, und so funktioniert die Regularisierung. Wenn Sie neugierig sind, wie diese Ableitungsterme berechnet wurden, habe ich nur noch eine letzte optionale Folie, die eine kleine Berechnung des Ableitungsterms durchführt. Auch hier sind diese Folie und der Rest dieses Videos völlig optional, was bedeutet, dass Sie nichts davon benötigen, um an den Übungsaufgaben und Tests teilzunehmen. Lassen Sie uns schnell zur Ableitungsrechnung übergehen. Die Ableitung von J nach w_j sieht folgendermaßen aus. Denken Sie daran, dass f(x) für die lineare Regression als w Punkt x plus b oder w Punktprodukt x plus b definiert ist. Es stellt sich heraus, dass die Ableitungen nach den Regeln der Analysis so aussehen: 1 über 2m mal die Summe i gleich 1 bis m von w Punkt x plus b minus y mal 2x_j plus der Ableitung des Regularisierungsterms, die Lambda über ist 2m mal 2 w_j. Beachten Sie, dass der zweite Term nicht mehr den Summationsterm von j gleich 1 bis n hat. Die beiden heben sich hier und hier und auch hier und hier auf. Es vereinfacht sich hier auf diesen Ausdruck. Denken Sie schließlich daran, dass wx + b f(x) ist, und Sie können es daher hier unten als diesen Ausdruck umschreiben. Aus diesem Grund wird dieser Ausdruck zur Berechnung des Gradienten in der regulierten linearen Regression verwendet. Sie wissen jetzt, wie Sie eine regulierte lineare Regression implementieren. Auf diese Weise reduzieren Sie die Überanpassung wirklich, wenn Sie über viele Funktionen und einen relativ kleinen Trainingssatz verfügen. Dadurch sollte die lineare Regression bei vielen Problemen viel besser funktionieren. Im nächsten Video wenden wir diese Regularisierungsidee auf die logistische Regression an, um eine Überanpassung auch für die logistische Regression zu vermeiden. Schauen wir uns das im nächsten Video an.

Regularisierte logistische Regression

In diesem Video sehen Sie, wie Sie eine regulierte logistische Regression implementieren. So wie die Gradientenaktualisierung für die logistische Regression der Gradientenaktualisierung für die lineare Regression überraschend ähnlich zu sein schien, stellen Sie fest, dass die Gradientenabstiegsaktualisierung für die regulierte logistische Regression auch der Aktualisierung für die regulierte lineare Regression ähnlich sein wird. Lass uns einen Blick darauf werfen. Hier ist die Idee. Wir haben bereits gesehen, dass die logistische Regression anfällig für eine Überanpassung sein kann, wenn man sie mit Polynommerkmalen sehr hoher Ordnung wie diesem anpasst. Hier ist z ein Polynom höherer Ordnung, das auf diese Weise an die Sigmoidfunktion übergeben wird, um f zu berechnen. Insbesondere kann es zu einer Entscheidungsgrenze kommen, die zu komplex ist und nicht als Trainingssatz geeignet ist. Generell gilt: Wenn Sie die logistische Regression mit vielen Merkmalen trainieren, seien es Polynommerkmale oder andere Merkmale, besteht möglicherweise ein höheres Risiko einer Überanpassung.



Dies war die Kostenfunktion für die logistische Regression. Wenn Sie es ändern möchten, um die Regularisierung zu verwenden, müssen Sie ihm lediglich den folgenden Begriff hinzufügen. Addieren wir Lambda zum Regularisierungsparameter über das 2m-fache der Summe von j gleich 1 bis n, wobei n wie üblich die Anzahl der Features von wj im Quadrat ist. Wenn Sie diese Kostenfunktion als Funktion von w und b minimieren, werden die Parameter w_1, w_2 bis w_n bestraft und verhindert, dass sie zu groß werden. Wenn Sie dies tun, erhalten Sie, auch wenn Sie ein Polynom höherer Ordnung mit vielen Parametern anpassen, immer noch eine Entscheidungsgrenze, die so aussieht. Etwas, das sinnvoller erscheint, um positive und negative Beispiele zu trennen und gleichzeitig hoffentlich auf neue Beispiele zu verallgemeinern, die nicht im Trainingssatz enthalten sind. Bei Verwendung der Regularisierung, auch wenn Sie über viele Funktionen verfügen. Wie kann man das konkret umsetzen? Wie kann man diese Kostenfunktion j von wb, die den Regularisierungsterm enthält, tatsächlich minimieren? Nun, verwenden wir den Gradientenabstieg wie zuvor. Hier ist eine Kostenfunktion, die Sie minimieren möchten. Um den Gradientenabstieg wie zuvor zu implementieren, führen wir die folgenden gleichzeitigen Aktualisierungen für wj und b durch. Dies sind die üblichen Aktualisierungsregeln für den Gradientenabstieg. Wenn Sie genau wie bei der regulierten linearen Regression berechnen, wo sich diese Ableitungsterme befinden, ändert sich jetzt nur noch, dass die Ableitung in Bezug auf wj diesen zusätzlichen Term erhält, Lambda über m mal wj, der hier am Ende hinzugefügt wird. Auch hier ähnelt es stark dem Update für die regulierte lineare Regression. Tatsächlich handelt es sich um genau dieselbe Gleichung, mit der Ausnahme, dass die Definition von f nun nicht mehr die lineare Funktion ist, sondern die auf z angewendete logistische Funktion. Ähnlich wie bei der linearen Regression werden wir nur die Parameter w, j, nicht jedoch den Parameter b regulieren, weshalb sich an der Aktualisierung, die Sie für b vornehmen, keine Änderung ergibt. Im letzten optionalen Labor dieser Woche befassen Sie sich noch einmal mit der Überanpassung. Im interaktiven Diagramm im optionalen Labor können Sie nun die Regularisierung Ihrer Modelle, sowohl der Regression als auch der Klassifizierung, auswählen, indem Sie die Regularisierung während des Gradientenabstiegs aktivieren, indem Sie einen Wert für Lambda auswählen. Bitte schauen Sie sich insbesondere den Code zur

Implementierung der regulierten logistischen Regression an, da Sie diesen Ende dieser Woche selbst in der Praxis umsetzen werden. Jetzt wissen Sie, wie Sie eine regulierte logistische Regression implementieren. Wenn ich durch das Silicon Valley spaziere, sehe ich viele Ingenieure, die maschinelles Lernen nutzen, um eine Menge Mehrwert zu schaffen und manchmal viel Geld für die Unternehmen zu verdienen. Ich weiß, dass Sie sich erst seit ein paar Wochen mit diesem Thema beschäftigen, aber wenn Sie lineare Regression und logistische Regression verstehen und anwenden können, ist das eigentlich alles, was Sie brauchen, um einige sehr wertvolle Anwendungen zu erstellen. Während die konkreten Lernergebnisse, die Sie nutzen, wichtig sind, erweist sich das Wissen darüber, wann und wie Überanpassung reduziert werden kann, auch in der realen Welt als eine der sehr wertvollen Fähigkeiten. Ich möchte dir gratulieren, wie weit du gekommen bist, und ich möchte dir sagen, dass du es großartig gemacht hast, bis zum Ende dieses Videos durchzukommen. Ich hoffe, dass Sie auch die Übungsaufgaben und Quizze durcharbeiten. Dennoch gibt es noch viele weitere spannende Dinge zu lernen. Im zweiten Kurs dieser Spezialisierung lernen Sie neuronale Netze, auch Deep-Learning-Algorithmen genannt, kennen. Neuronale Netze sind heute für viele der neuesten Durchbrüche im Auge verantwortlich, von der praktischen Spracherkennung über Computer, die Objekte und Bilder genau erkennen, bis hin zu selbstfahrenden Autos. Die Art und Weise, wie ein neuronales Netzwerk aufgebaut wird, nutzt tatsächlich viel von dem, was Sie bereits gelernt haben, wie Kostenfunktionen, Gradientenabstieg und Sigmoidfunktionen. Nochmals herzlichen Glückwunsch zum Ende dieser dritten und letzten Woche von Kurs 1. Ich hoffe, Sie haben [unverständlich] und wir sehen uns nächste Woche im Material über neuronale Netze.

Andrew Ng und Fei-Fei Li über menschenzentrierte KI

Hallo, ich freue mich, heute meinen alten Freund, Professor Fei-Fei Li, hier bei uns zu haben. Fei-Fei ist Professorin für Informatik an der Stanford University und außerdem Co-Direktorin von HAI; das Human-Centered AI Institute. Zuvor war sie auch als Chefwissenschaftlerin der Abteilung für KI bei Google Cloud verantwortlich. Schön, dass du da bist, Fei-Fei. Danke, Andrew. Ich freue mich sehr, hier zu sein. Wie lange kennen wir uns schon? Ich habe den Überblick verloren. Auf jeden Fall mehr als ein Jahrzehnt. Ich kannte Ihre Arbeit, bevor wir uns überhaupt trafen, und ich kam 2009 nach Stanford, aber wir begannen 2007, also vor 15 Jahren, miteinander zu reden. Ich habe tatsächlich noch sehr klare Erinnerungen daran, wie stressig es war, als wir alle zusammen waren; Ich, Chris Manning, ein paar von uns. Wir haben versucht herauszufinden, wie wir Sie rekrutieren können, um nach Stanford zu kommen. Es war nicht schwer. Ich musste nur das Leben meines Studenten regeln, aber es ist schwer, Stanford zu widerstehen. Wirklich toll, Sie als Freund und Kollegen hier zu haben. Ich auch. Es ist lange her und wir haben großes Glück, diese Generation zu sein. KI zu sehen ist ein großer Fortschritt. Es gab etwas an Ihrem Hintergrund, das mich immer inspiriert hat, nämlich dass heute Menschen aus allen Gesellschaftsschichten in die KI einsteigen und sich manchmal immer noch fragen: Ist KI der richtige Weg für mich? Ich dachte, einer der interessantesten Aspekte Ihres Hintergrunds bestand darin, dass Sie eigentlich nicht Informatik oder KI studiert haben, sondern dass Sie zunächst Physik studiert haben und dann diesen Weg zu einem der weltweit bekanntesten KI-Wissenschaftler eingeschlagen haben. Wie haben Sie den Wechsel von der Physik zur KI geschafft? Nun, das ist eine großartige

Frage, Andrew, vor allem liegt uns beiden die Zukunft junger Menschen am Herzen, und sie kommen in die Welt der KI. Die Wahrheit ist: Wenn ich damals vor mehr als 20 Jahren in die KI einsteigen konnte, dann kann heute jeder in die KI einsteigen, denn KI ist zu einer so vorherrschenden und weltweit einflussreichen Technologie geworden. Aber ich selbst war vielleicht ein Unfall. Ich war schon immer ein Physikkind oder ein MINT-Kind. Ich bin mir sicher, dass Sie das auch waren, aber Physik war meine Leidenschaft während der Mittelschule, der Oberschule und dem College, und ich ging nach Princeton und studierte Physik als Hauptfach. Eine Sache, die mich die Physik bis heute gelehrt hat, ist die Leidenschaft, große Fragen zu stellen, die Leidenschaft, keine Sterne zu suchen. Als Physikstudent in Princeton hatte ich wirklich viel Spaß. Eine Sache, die ich gemacht habe, war, Geschichten und Schriften großer Physiker des 20. Jahrhunderts zu lesen und einfach zu hören, was sie über die Welt denken, insbesondere Menschen wie Albert Einstein, Roger Penrose und Erwin Schrödinger, und es war wirklich lustig, so viele zu bemerken In den Schriften in der späteren Hälfte der Karriere dieser großen Physiker ging es nicht nur um die Atomwelt oder die physische Welt, sondern um Überlegungen zu ebenso kühnen Fragen wie dem Leben, der Intelligenz und den menschlichen Bedingungen. Schrödinger hat dieses Buch geschrieben: Was ist Leben? Roger Penrose hat dieses Buch geschrieben, Emperor's New Mind. Das hat mich wirklich sehr neugierig auf das Thema Intelligenz gemacht. Während meiner Studienzeit führte eins zum anderen. Ich absolvierte ein Praktikum an der Spitze neurowissenschaftlicher Labore, insbesondere im Bereich der Sehkraft, und dachte mir: „Wow, das ist eine genauso gewagte Frage wie der Anfang des Universums oder die Frage, woraus Materie besteht.“ von? Das brachte mich dazu, vom Grundstudium der Physik zum weiterführenden Studium der KI zu wechseln. Auch wenn ich zu unserer Zeit nichts über Sie weiß, war KI ein Schimpfwort. Es war KI-Winter, also mehr maschinelles Lernen, Computer Vision und Computational Neuroscience. Ja ich weiß. Ehrlich gesagt glaube ich, dass ich als Student zu sehr damit beschäftigt war, Code zu schreiben. Ich habe es einfach geschafft, den KI-Winter unbekümmert zu ignorieren und einfach weiter zu programmieren. Ja, nun ja, ich war zu sehr damit beschäftigt, PDE-Gleichungen zu lösen. Haben Sie jetzt tatsächlich eine dreiste Frage? Ja, meine kühne Frage ist immer noch Intelligenz. Ich denke, seit Alan Turing hat die Menschheit die grundlegenden Computerprinzipien hinter der Intelligenz nicht vollständig verstanden. Heute verwenden wir die Worte KI, wir verwenden das Wort AGI, aber letzten Endes träume ich immer noch von einer Reihe einfacher Gleichungen oder einfacher Prinzipien, die den Prozess der Intelligenz definieren können, egal ob es sich um tierische oder maschinelle Intelligenz handelt. Das ähnelt der Physik. Viele Menschen haben zum Beispiel den Vergleich mit dem Fliegen gezogen. Reproduzieren wir fliegende Vögel oder bauen wir Flugzeuge? Viele Menschen stellen die Frage nach der Beziehung zwischen KI und Gehirn. Egal, ob wir einen Vogel bauen, einen Vogel nachbilden oder ein Flugzeug bauen, für mich bestimmen letztendlich Aerodynamik und Physik den Prozess des Fliegens, und ich glaube, dass wir das eines Tages entdecken werden. Ich denke manchmal über diese eine Lernalgorithmus-Hypothese nach. Könnte eine Menge Intelligenz, vielleicht nicht alle, aber ein Großteil davon, durch ein oder ein sehr einfaches Prinzip des maschinellen Lernens erklärt werden? Es fühlt sich an, als wären wir noch so weit davon entfernt, diese Nuss zu knacken. Aber an den Wochenenden, an denen ich Freizeit habe und über das Erlernen von Algorithmen nachdenke und darüber nachdenke, wo sie hingehen könnten, ist das eines der Dinge, worüber ich mich freue. Ich denke gerade darüber nach. Ich bin vollkommen

einverstanden. Ich habe immer noch das Gefühl, dass wir Vor-Newtonianer sind, wenn wir vor Newton physikalische Analogien anstellen. Es gab großartige Physik, großartige Physiker, viel Phänomenologie, viele Studien darüber, wie sich die Astralkörper bewegen und so weiter. Aber es war Newton, der begann, sehr einfache Gesetze zu schreiben. Ich denke, wir erleben immer noch das sehr aufregende Erwachsenwerden der KI als Grundlagenwissenschaft. Meiner Meinung nach sind wir vor Newton. Es ist wirklich schön, Sie darüber sprechen zu hören, wie es trotz maschinellem Lernen und KI so weit gekommen ist. Es fühlt sich immer noch so an, als gäbe es noch viel mehr unbeantwortete Fragen, viel mehr Arbeit, die vielleicht von einigen der Leute, die sich heute in diesem Bereich engagieren, erledigt werden muss, als die Arbeit, die bereits erledigt wurde. Absolut. Rechnen wir mal, es sind nur etwa 160 Jahre. Es ist ein sehr junges Feld. Die moderne Physik, Chemie und Biologie sind alle Hunderte Jahre alt. Ich finde es sehr spannend, heute in den Bereich der Intelligenzwissenschaft einzusteigen und KI zu studieren. Ja, eigentlich ist das gut. Ich glaube, ich erinnere mich an ein Gespräch mit dem verstorbenen Professor John McCarthy, der den Begriff künstliche Intelligenz geprägt hatte, und Junge, das Fachgebiet hat sich verändert, seit er in einem Workshop darüber nachgedacht und den Begriff KI erfunden hat. Aber vielleicht kommt in zehn Jahren jemand, der sich das anschaut, mit neuen Ideen und dann werden wir sagen: „Junge, die KI zeigt uns etwas anderes, als du und ich es uns vorgestellt haben." Das ist die spannende Zukunft, auf die wir aufbauen können. Ja, ich bin sicher, Newton hätte nicht von Einstein geträumt. Unsere Entwicklung der Wissenschaft schreitet manchmal voran, manchmal dauert es eine Weile, und ich denke, wir befinden uns gerade in einer absolut aufregenden Phase der KI. Es ist interessant zu hören, wie Sie diese großartige Vision für KI malen. Wenn ich etwas zurückblicke, gab es einen Teil Ihres Hintergrunds, den ich inspirierend fand: Als Sie gerade erst angefangen haben, habe ich Sie darüber sprechen hören, wie Ihr Physikstudent ist, aber nicht nur das, Sie sind Außerdem betreibt er einen Waschsalon, um die Schulkosten zu finanzieren. Erzählen Sie uns einfach mehr darüber. Ich kam in dieses Land, nach New Jersey, als ich 15 war. Das Tolle daran, in New Jersey zu sein, ist die Nähe zu Princeton, sodass ich oft einfach einen Wochenendausflug mit meinen Eltern mache und den Ort bewundere, an dem Einstein die meiste Zeit verbracht hat seine Karriere in der zweiten Hälfte seines Lebens. Aber mit dem typischen Einwandererleben war es hart. Als ich Princeton betrat, sprachen meine Eltern kein Englisch und eins führte zum anderen. Es stellte sich heraus, dass der Betrieb einer chemischen Reinigung die beste Option für meine Familie sein könnte, insbesondere für mich, um dieses Unternehmen zu leiten, da es sich um ein Wochenendgeschäft handelt. Wenn es sich um ein Werktagsgeschäft handelt, wäre es für mich schwierig, Student zu sein. Tatsächlich ist es, ob Sie es glauben oder nicht, so, dass der Betrieb einer Textilreinigung sehr maschinenintensiv ist, was für einen MINT-Studenten wie mich gut ist. Wir beschlossen, eine chemische Reinigung in einer kleinen Stadt in New Jersey namens Parsippany in New Jersey zu eröffnen. Es stellte sich heraus, dass wir physisch nicht allzu weit von den Bell Labs entfernt waren und wo viele frühe Forschungen zu Faltungs-Neuronalen Netzwerken stattfanden, aber ich hatte keine Ahnung. Es ist nur ein Sommerpraktikant bei der Antike. Das ist richtig, mit Rob Shapiro. Mit Michael Kearns war mein Mentor und Rob Shapiro, dem Erfinder dieser Dinge-Creator-Algorithmen. Sie kodieren KI. Ich habe es versucht [unverständlich] Erst viel später begann ich mit dem Praktikum. Ja. Dann waren es sieben Jahre. Ich habe das während des gesamten Grundstudiums und des größten Teils meiner

Graduiertenschule gemacht und stelle meine Eltern ein. Ja. Ja. Das ist wirklich inspirierend. Ich weiß, dass du dein ganzes Leben lang dazu erzogen wurdest, genau das zu tun. Ich denke, die Geschichte vom Betrieb eines Waschsalons gehört zu den weltweit führenden Informatikern. Ich hoffe, das inspiriert einige Leute, die sich das ansehen, dass es, egal wo man ist, für jeden etwas dabei hat. Ich weiß das nicht einmal. In meiner High-School-Zeit war ich als Bürokauffrau tätig, und so erinnere ich mich bis heute daran, dass ich viel fotokopiert habe. Der aufregende Teil war die Verwendung dieses Aktenvernichters. Das war etwas Glamouröses. Ich habe in der High School so viel Kaffee getrunken, dass ich dachte, Junge, Schwerverbrecher, ich könnte einen Roboter bauen, der das für den Kaffee macht, vielleicht könnte ich etwas tun. Warst du erfolgreich? Ich arbeite immer noch daran. Wenn die Leute an Sie und die Arbeit denken, die Sie geleistet haben, ist einer der großen Erfolge, die jeder an dieses ImageNet denkt, das dabei hilft, einen frühen Maßstab für Computer Vision zu etablieren. Es war wirklich entscheidend für den modernen Aufstieg von Deep Learning und Computer Vision. Ich wette, nicht viele Leute wissen, wie Sie eigentlich mit ImageNet angefangen haben. Erzählen Sie uns die Entstehungsgeschichte von ImageNet. Ja. Nun, Andrew, das ist eine gute Frage, denn viele Leute sehen in ImageNet nur die Kennzeichnung einer Menge Bilder. Aber eigentlich haben wir damit begonnen, dass ein Northstar meinen Physik-Hintergrund zurückbringt. Als ich in die Graduiertenschule kam, wann sind Sie in das Graduiertenjahr eingetreten? '97. Ich war drei Jahre später im Jahr 2000. Das war eine sehr aufregende Zeit, denn ich war im Labor für Computer Vision und Computational Neural Science von Pietro Purana und Christoph Car am Caltech. Im Vorfeld gab es zunächst zwei Dinge, die sehr aufregend waren. Einer davon ist, dass die Welt der KI zu diesem Zeitpunkt noch nicht KI hieß. Computer Vision oder die Verarbeitung natürlicher Sprache hat Lingua Franca gegründet. Es ist maschinelles Lernen und statistische Modellierung als neues Werkzeug aufgetaucht. Ich meine, es gibt es schon. Ich erinnere mich, dass die Idee, maschinelles Lernen auf Computer Vision anzuwenden, umstritten war. Ich gehörte zur ersten Generation von Doktoranden, die sich mit dem gesamten Basisnetz, allen Inferenzalgorithmen und all dem beschäftigten. Das war ein aufregendes Ereignis. Ein sicherlich aufregendes Ereignis, das die meisten Menschen nicht kennen und nicht wertschätzen, sind die paar Jahrzehnte, wahrscheinlich eines von zwei oder drei Jahrzehnten unglaublicher Arbeit in der Kognitionswissenschaft und kognitiven Neurowissenschaft im Bereich des Sehens, in der Welt des Sehens, des Menschen Vision. Das hat wirklich ein paar wirklich kritische Northstar-Probleme aufgedeckt. Nur das Verständnis der menschlichen visuellen Verarbeitung und der menschlichen Intelligenz. Eine davon ist die Anerkennung des Verständnisses natürlicher Objekte und natürlicher Dinge. Denn viele Arbeiten aus der Psychologie und der Kognitionswissenschaft weisen auf uns hin. Das ist von Natur aus optimiert, was auch immer das Wort ist. Die Funktionalität und Leistungsfähigkeit der menschlichen Intelligenz ist robuster, schneller und differenzierter als wir dachten. Wir finden sogar neuronale Korrelate, Gehirnbereiche, die Gesichtern, Orten oder Körperteilen zugeordnet sind. Diese beiden Dinge führten zu meiner Doktorarbeit über den Einsatz von Methoden des maschinellen Lernens zur Arbeit an der Erkennung realer Objekte. Aber es wird sehr schnell sehr schmerzhaft, wenn wir mit einer der nach wie vor größten Herausforderungen beim KI-Maschinenlernen konfrontiert werden, nämlich der mangelnden Generalisierbarkeit. Sie können ein schönes Modell entwerfen oder möchten, wenn Sie das Modell übermäßig anpassen. Ich erinnere mich an die Zeit, als es möglich war, eine

Computervision und einen Aufsatz zu veröffentlichen, der zeigt, dass es auf einem Bild funktioniert. Exakt. Es ist einfach die Überanpassung. Die Modelle sind nicht sehr aussagekräftig und uns fehlen die Daten. Als Fachgebiet haben wir auch darauf gesetzt, die Variablen durch von Hand entwickelte Funktionen sehr umfangreich zu gestalten. Denken Sie daran, dass jede Variable eine Menge semantischer Bedeutung hat, aber über handgefertigte Funktionen verfügt. Dann, gegen Ende meiner Doktorarbeit, sehen mein Betreuer Pietro und ich uns gegenseitig an und sagen: „Junge, wir brauchen mehr Daten." Wenn wir an dieses North-Star-Problem der Objekterkennung glauben und auf die Werkzeuge zurückblicken, die uns zur Verfügung stehen, überpassen wir mathematisch gesehen jedes Modell, auf das wir stoßen, über. Wir müssen das aus einem neuen Blickwinkel betrachten. Eins führte zum anderen. Er und ich haben beschlossen, dass wir es an diesem Punkt einfach tun werden. Wir glauben, dass es sich um ein großes Datenprojekt namens Caltech 101 handelte. Ich erinnere mich an den Datensatz. Ich habe vor langer Zeit Artikel mit Ihrem Caltech 101-Datensatz geschrieben. Das haben Sie getan, Sie und Ihr früher Doktorand. Dieser Caltech 101-Datensatz hat vielen Forschern geholfen. Das waren meine Mutter und ich, die Bilder beschrifteten, und ein paar Studenten, aber das waren die Anfänge des Internets. Plötzlich war die Verfügbarkeit von Daten etwas Neues. Ich erinnere mich, dass Pietro immer noch diese super teure Digitalkamera hatte. Ich glaube, es war Canon oder etwas in der Größenordnung von 6.000 US-Dollar, das am Caltech herumlief und Fotos machte. Aber wir sind die Generation Internet. Ich gehe zur Google-Bildersuche, sehe Tausende und Abertausende Bilder und sage Pietro: „Lass uns einfach herunterladen." Natürlich ist das Herunterladen nicht so einfach. Eins führte zum anderen. Wir erstellen diesen Caltech 101-Datensatz mit 101 Objektkategorien und etwa 30.000 Bildern. Ich denke, wir sagen wirklich, dass, obwohl heute jeder von ImageNet gehört hat, sogar Sie ein paar Iterationen durchgeführt haben, bei denen Sie Caltech 101 durchgeführt haben, und das war ein Erfolg. Viele Leute nutzten es sogar für die ersten Erkenntnisse aus dem Bau von Caltech 101. Sie gaben Ihnen die Grundlage für den Aufbau dessen, was sich als noch größerer Erfolg herausstellte. Als ich jedoch Assistenzprofessor wurde, begannen wir, uns mit dem Problem zu befassen. Mir wurde klar, dass es viel größer ist, als wir denken. Rein mathematisch gesehen reichte Caltech 101 nicht aus, um die Algorithmen zu betreiben. Wir beschlossen, dort Bilder zu machen. Das war der Zeitpunkt, an dem die Leute anfingen zu glauben, wir würden zu viel tun. Es ist einfach zu verrückt, die Idee, das gesamte Internet voller Bilder herunterzuladen und alle englischen Substantive zu kartieren, war ein bisschen zu groß. Ich fange an, viel Gegenwind zu bekommen. Ich erinnere mich, dass auf einer der CVPR-Konferenzen, als ich die frühe Idee von ImageNet vorstellte, ein paar Forscher öffentlich befragt und gesagt haben: „Wenn Sie eine Kategorie von Objekten nicht erkennen können, sagen wir den Stuhl, auf dem Sie sitzen, wie stellen Sie sich das vor?" oder was nützt ein Datensatz aus 22.000 Klassen mit 15 Millionen Bildern?" Letztendlich hat dieser riesige Datensatz für eine unverständliche Anzahl von Forschern auf der ganzen Welt einen großen Mehrwert geschaffen. Ich denke, es war die Kombination aus Wetten auf das richtige Nordsternproblem und den Daten, die es antreiben. Es war ein lustiger Prozess. Wenn ich über diese Geschichte nachdenke, kommt es mir wie eines dieser Beispiele vor, bei denen Menschen manchmal das Gefühl haben, sie sollten nur an Projekten arbeiten, bei denen das große Ding am Anfang fehlt. Aber ich habe das Gefühl, dass es für Leute, die im Bereich maschinelles Lernen arbeiten, völlig in Ordnung ist, wenn ihr erstes Projekt etwas kleiner ist.

Ich wünsche dir einen guten Sieg. Nutzen Sie das Lernen, um noch größere Dinge voranzutreiben, und manchmal gewinnt die ImageNet-Größe alles. Aber mittlerweile denke ich, dass es auch wichtig ist, von einem kühnen Ziel getrieben zu sein. Sie können Ihr Problem oder Ihr Projekt anhand lokaler Meilensteine usw. auf dieser Reise einschätzen, aber ich schaue mir auch einige unserer aktuellen Studenten an. Sie stehen unter dem enormen Druck des aktuellen Klimas des ununterbrochenen Publizierens. Es wird immer mehr zu inkrementellen Arbeiten kommen, einfach nur um ihrer selbst willen in eine Veröffentlichung einzusteigen. Ich persönlich dränge meine Schüler immer dazu, die Frage zu stellen: Was ist der Polarstern, der Sie antreibt? Ja, das ist wahr. Wenn ich selbst recherchiere, habe ich im Laufe der Jahre immer das getan, was mich begeistert, wobei ich versuchen möchte, die Sichtweise voranzutreiben. Man muss nicht auf die Leute hören. Man muss den Leuten zuhören und sie seine Meinung formen lassen. Aber letzten Endes denke ich, dass die besten Forscher die Welt ihre Meinung formen lassen, aber am Ende die Dinge vorantreiben, indem sie ihre eigene Meinung verwenden. Stimmt voll und ganz zu, ja. Es ist dein eigenes Feuer. Als ein Forschungsprogramm entwickelt wurde, haben Sie Ihre, sagen wir mal, Grundlagen in Computer Vision und Neurowissenschaften übernommen und sie auf alle möglichen Themen angewendet, einschließlich Ihrer ganz offensichtlichen Gesundheitsfürsorge. Blick auf neurowissenschaftliche Anwendungen. Wir würden gerne etwas mehr darüber hören. Ja, gerne. Ich denke, dass die Entwicklung meiner Forschung im Bereich Computer Vision auch der Entwicklung der visuellen Intelligenz bei Tieren folgt. Es gibt zwei Themen, die mich wirklich begeistern. Erstens: Was ist ein wirklich wirkungsvoller Anwendungsbereich, der Menschenleben helfen würde? Das ist meine Arbeit im Gesundheitswesen. Die andere Frage ist, worum es bei der Vision am Ende des Tages geht? Das bringt mich dazu, den Kreis zwischen Wahrnehmung und robotergestütztem Lernen zu schließen. Was das Gesundheitswesen betrifft, Andrew, gab es vor etwa zehn Jahren eine Zahl, die mich schockierte, als ich meinen langjährigen Mitarbeiter Dr. Arnie Milstein an der Stanford Medical School traf, und diese Zahl beläuft sich auf etwa eine Viertelmillion Amerikaner, die sterben medizinischer Fehler jedes Jahr. Ich hätte nie gedacht, dass die Zahl aufgrund medizinischer Fehler so hoch sein könnte. Es gibt viele Gründe, aber wir können sicher sein, dass die meisten Gründe nicht beabsichtigt sind. Dies sind zum Beispiel Fehler, unbeabsichtigte Fehler und Sonneneinstrahlung. Das ist eine umwerfende Zahl. Es ist. Jährlich sterben etwa 40.000 Menschen bei Autounfällen. Es ist einfach völlig tragisch und das ist sogar [unhörbar], das wollte ich sagen. Ich bin froh, dass du es angesprochen hast. Nur ein Beispiel. Eine Zahl innerhalb dieser verblüffenden Zahl ist, dass die Zahl der Todesfälle durch im Krankenhaus erworbene Infektionen bei über 95.000 liegt. Das ist das 2,5-fache der Zahl der Todesopfer bei Autounfällen. In diesem speziellen Fall hat sich das Krankenhaus aus vielen Gründen eine Infektion zugezogen. Aber im Großen und Ganzen mangelt es an guter Handhygienepraxis. Wenn man sich die WHO anschaut, gibt es viele Protokolle über die Handhygienepraxis von Ärzten. Aber in der echten Gesundheitsversorgung macht man immer noch viele Fehler, wenn es hektisch zugeht, der Prozess langwierig ist und es an einem Feedback-System mangelt. Eine weitere tragische medizinische Tatsache ist, dass jedes Jahr mehr als 70 Milliarden US-Dollar für Verletzungen und Todesfälle ausgegeben werden. Die meisten davon passieren älteren Menschen zu Hause, aber auch in den Krankenzimmern. Das sind riesige Probleme. Als Arnie und ich uns 2012 trafen, war das selbstfahrende Auto auf dem Höhepunkt, sagen wir mal, kein Hype.

Aber welches ist das richtige Wort? Aufregung im Silicon Valley. Dann werfen wir einen Blick auf die Technologie intelligenter Sensorkameras, Lidars usw., intelligenter Sensoren, Algorithmen für maschinelles Lernen und ein ganzheitliches Verständnis einer komplexen Umgebung, in der Menschenleben sehr auf dem Spiel stehen. Ich habe mir das alles für selbstfahrende Autos angesehen und festgestellt, dass wir bei der Gesundheitsversorgung die gleiche Situation haben. Ein Großteil des Prozesses, des menschlichen Verhaltensprozesses im Gesundheitswesen, liegt im Dunkeln. Wenn wir intelligente Sensoren haben könnten, sei es in Patientenzimmern oder Seniorenheimen, die unseren Ärzten und Patienten helfen, sicherer zu bleiben, wäre das großartig. Arnie und ich haben uns an die Arbeit gemacht, die wir als Forschungsagenda für Umgebungsintelligenz bezeichnen. Aber eine Sache, die ich gelernt habe und die wahrscheinlich zu unseren anderen Themen führen wird, ist, dass es, sobald man KI auf reale menschliche Bedingungen anwendet, neben den Problemen des maschinellen Lernens auch viele menschliche Probleme gibt , zum Beispiel den Datenschutz. Ich erinnere mich an die Lektüre einiger Ihrer Aufsätze mit Arnie und fand es wirklich interessant, wie Sie Systeme aufbauen und einsetzen können, die relativ datenschutzkonform sind. Ja, nun ja, danke. Nun, die erste Version dieser Technologie besteht darin, dass wir Kameras verwenden, die keine RGB-Informationen erfassen. Sie haben viel davon in selbstfahrenden Autos verwendet, zum Beispiel diese Tiefenkameras. Dort wahren Sie viele Datenschutzinformationen, indem Sie einfach die Gesichter und die Identität der Personen nicht sehen. Aber was im letzten Jahrzehnt wirklich interessant ist, ist, dass die technologischen Veränderungen uns tatsächlich ein größeres Toolset für den Schutz der Privatsphäre und der Datenverarbeitung in diesem Zustand zur Verfügung gestellt haben, zum Beispiel durch Inferenz auf dem Gerät. Da die Chips immer leistungsfähiger werden, können Sie den Menschen besser helfen, wenn Sie keine Daten mehr über das Netzwerk und an den zentralen Server übertragen müssen. Wir wissen, dass sich das föderierte Lernen noch in einem frühen Stadium befindet, aber es ist ein weiteres potenzielles Werkzeug für Datenschutz und geschütztes Computing. Dann differenzierter Datenschutz und auch Verschlüsselungstechnologien. Wir beginnen zu erkennen, dass menschliche Anforderungen, Privatsphäre und andere Probleme tatsächlich eine neue Welle maschineller Lerntechnologie im Bereich der Umgebungsintelligenz im Gesundheitswesen antreiben. Ja, es hat mich ermutigt, Ihre realen praktischen Anwendungen der differenzierten Privatsphäre zu sehen, die tatsächlich real sind. Federated Learning, wie Sie sagten: Beten Sie, dass die PRRs der Realität ein wenig voraus sind, aber ich denke, wir werden es schaffen. Aber es ist interessant, wie die Verbraucher in den letzten Jahren glücklicherweise viel besser und zunehmend besser über den Datenschutz informiert sind. Ich denke, die Öffentlichkeit macht uns auch zu besseren Wissenschaftlern. Ja, ich denke, letztendlich verstehen die Leute, dass die KI jeden beherbergt, uns eingeschlossen, aber jeden dafür verantwortlich macht, wirklich das Richtige zu tun. Ja. In diesem Sinne bestand eine der wirklich interessanten Arbeiten, die Sie geleistet haben, darin, mehrere Bemühungen zur Aufklärung der Gesetzgeber voranzutreiben. Ich hoffe, dass die Regierungen, insbesondere die US-Regierung, bessere Gesetze und eine bessere Regulierung durchsetzen, insbesondere im Zusammenhang mit KI. Das hört sich sehr wichtig an und ich vermute, dass es an manchen Tagen eine etwas frustrierende Arbeit sein wird, aber wir würden gerne mehr darüber hören. Ja, zunächst einmal muss ich vielen, vielen Leuten Anerkennung zollen. Vor ungefähr vier Jahren beendete ich tatsächlich mein Sabbatical von

Google. Es war für mich ein großes Privileg, mit so vielen Unternehmen zusammenzuarbeiten. Unternehmensentwickler, aber auch eine große Anzahl und Vielfalt vertikaler Branchen erkennen die Auswirkungen von KI auf den Menschen. Das war eines, das bedeutet, dass die Fakultätsleiter von Stanford und auch nur unser Präsident, der Rektor, der ehemalige Präsident und der ehemalige Rektor zusammenkommen und erkennen, dass Stanford eine historische Rolle bei der Weiterentwicklung der KI spielen muss. Wir waren Teil des Geburtsortes der KI. Eine Menge Arbeit, die unsere vorherige Generation geleistet hat, und die gesamte Arbeit, die Sie geleistet haben, und einige der Arbeiten, die ich geleistet habe, haben zur heutigen KI geführt. Was ist unsere historische Chance und Verantwortung? Daher glauben wir, dass die nächste Generation der KI-Ausbildung, -Forschung und -Politik den Menschen in den Mittelpunkt stellen muss. Nach der Gründung des Human Center AI Institute, das wir HAI nennen, war eine der Arbeiten, die mich wirklich aus meiner Komfortzone herausführte, deren Ziel darin bestand, Fachwissen wirklich auf eine tiefere Auseinandersetzung mit politischen Denkern und Entscheidungsträgern zu richten. Denn wir sind hier im Silicon Valley und es gibt eine Kultur im Silicon Valley, die darin besteht, einfach weiter Dinge zu machen und das Gesetz wird von selbst aufholen. Aber KI wirkt sich manchmal so schnell negativ auf das Leben der Menschen aus, dass sie für keinen von uns gut ist. Wenn wir als Experten nicht mit den politischen Denkern und Entscheidungsträgern am Tisch sitzen und wirklich versuchen, diese Technologie für die Menschen besser zu machen. Wir reden über Fairness, wir reden über Privatsphäre, wir reden auch über die Abwanderung von KI in die Industrie und die Konzentration von Daten und Rechenleistung in einer kleinen Anzahl von Technologieunternehmen. All dies ist wirklich Teil der Veränderungen unserer Zeit. Einige sind wirklich aufregende Veränderungen, andere haben tiefgreifende Auswirkungen, die wir noch nicht unbedingt vorhersagen können. Eine der politischen Arbeiten, an denen sich Stanford AI sehr stolz beteiligt, besteht darin, dass wir eine der führenden Universitäten waren, die sich für einen Gesetzentwurf namens „National AI Research Cloud Task Force Bill“ eingesetzt haben. Der Name wurde von „Research Cloud“ in „Research Resource“ geändert. Das Akronym des Gesetzentwurfs lautet nun NAIR, National AI Research Resource. Dieser Gesetzentwurf fordert eine Task Force, die einen Fahrplan für den öffentlichen Sektor Amerikas, insbesondere den Hochschul- und Forschungssektor, erstellen soll, um seinen Zugang zu Ressourcen für KI-Rechen- und KI-Daten zu verbessern, und zielt tatsächlich darauf ab, Amerikas Ökosystem für KI-Innovationen zu verjüngen und Forschung. Ich bin Teil der 12-köpfigen Task Force unter der Biden-Regierung für diesen Gesetzentwurf, und wir hoffen, dass es sich hierbei um einen Teil der Politik handelt, der keine Regulierungspolitik ist, sondern eher eine Anreizpolitik zum Aufbau und zur Verjüngung von Ökosystemen . Ich freue mich, dass Sie dies tun, The Help Shape US Policy, und dafür sorgen, dass genügend Ressourcen bereitgestellt werden, um eine gesunde Entwicklung der KI zu gewährleisten. Ich habe das Gefühl, dass dies zum jetzigen Zeitpunkt jedes Land braucht. Ja. Allein aufgrund der Dinge, die Sie selbst tun, ganz zu schweigen von den Dingen, die die globale KI-Community tut, gibt es in der KI derzeit einfach so viel zu tun, so viele Möglichkeiten, so viel Aufregung. Ich habe festgestellt, dass für jemanden, der zum ersten Mal mit maschinellem Lernen anfängt, manchmal so viel passiert, dass er sich fast ein wenig überwältigend fühlt. Total. Welchen Rat haben Sie für jemanden, der mit maschinellem Lernen beginnt? Gute Frage, Andrew, ich bin sicher, du hast einen tollen Rat. Sie sind einer der weltweit bekannten

Befürworter der Ausbildung im Bereich KI-Maschinenlernen. Diese Frage wird mir auch oft gestellt, und Sie haben völlig Recht: KI fühlt sich heute wirklich anders an als unsere Zeit. Nur zur Klarstellung: Ihr seid alle noch pünktlich. Das stimmte, als wir mit der KI angefangen haben. Ich finde es toll, dass wir immer noch Teil davon sind. Als wir anfingen, war der Zugang zu KI und maschinellem Lernen relativ eng. Man muss fast mit der Informatik anfangen und loslegen. Als Physikstudent musste ich mich immer noch in die Informatik- oder Elektrotechnik-Studiengänge zwängen, um zur KI zu gelangen. Aber heute denke ich tatsächlich, dass es viele Aspekte der KI gibt, die Einstiegspunkte für Menschen aus allen Lebensbereichen schaffen. Was die technische Seite angeht, denke ich, dass es offensichtlich ist, dass es im Internet einfach eine unglaubliche Fülle an Ressourcen gibt, von Coursera über YouTube bis hin zu TikTok. Im Vergleich zu der Zeit, als wir mit maschinellem Lernen begannen, gibt es für Studenten auf der ganzen Welt einfach so viel, was sie über KI und maschinelles Lernen lernen können Lernen. Auch alle Campusse, wir reden hier nicht nur über College-Campusse, sondern über High-School-Campusse, oder sogar manchmal früher, wir sehen mehr verfügbare Kurse und Ressourcen. Ich ermutige die jungen Leute mit technischem Interesse, Ressourcen und Möglichkeiten, diese Ressourcen zu nutzen, weil es viel Spaß macht. Aber dennoch gilt für diejenigen unter Ihnen, die nicht aus technischer Sicht kommen, die dennoch eine Leidenschaft für KI haben, sei es die nachgelagerte Anwendung oder die Kreativität, die sie erzeugt, oder der politische und soziale Aspekt oder wichtige soziale Probleme Digitale Ökonomie oder Governance oder Geschichte, Ethik, Politikwissenschaften, ich lade Sie ein, sich uns anzuschließen, denn es gibt noch viel zu tun. Es gibt viele unbekannte Fragen, zum Beispiel versucht mein Kollege bei HAI Antworten darauf zu finden: Wie definieren Sie unsere Wirtschaft im digitalen Zeitalter? Was bedeutet es, wenn Roboter bzw. Software immer stärker in den Arbeitsablauf eingreifen? Wie messen Sie unsere Wirtschaft? Das ist keine Frage zur KI-Codierung, sondern eine Frage zur KI-Auswirkung. Wir schauen uns die unglaublichen Fortschritte der generativen KI an und es wird noch mehr geben. Was bedeutet das für die Kreativität und für die Schöpfer von Musik über Kunst bis hin zum Schreiben? Ich denke, es gibt viele Bedenken, und ich denke, das ist auch berechtigt, aber in der Zwischenzeit müssen die Leute zusammenarbeiten, um das herauszufinden und dieses neue Tool auch zu nutzen. Kurz gesagt, ich denke, es ist einfach eine sehr aufregende Zeit, und jeder, egal in welchem Lebensbereich, hat eine Rolle zu spielen, solange er eine Leidenschaft dafür hat. Das ist wirklich spannend, wenn man über Wirtschaftswissenschaften spricht, denken Sie an meine Gespräche mit Professor Erik Brynjolfsson. Auswirkungen von KI auf die Wirtschaft, aber nach dem, was Sie sagen, und ich stimme zu, scheint es, dass KI unabhängig von Ihren aktuellen Interessen eine so universelle Technologie ist, dass die Kombination Ihres aktuellen Interesses an KI oft vielversprechend ist. Ich finde, dass selbst bei Lernenden, die vielleicht noch kein spezifisches Interesse haben, oft das Interesse wächst, wenn man den Weg in die KI findet und anfängt, Dinge zu lernen, und dann kann man beginnen, seinen eigenen Weg zu finden. Angesichts der Art und Weise, wie die KI heute ist, gibt es immer noch so viel Raum und so viel Bedarf an viel mehr Menschen, die ihren eigenen Weg gehen, um diese aufregende Arbeit zu leisten, von der die Welt meiner Meinung nach noch viel mehr braucht. Stimme voll und ganz zu. Eine Arbeit, die Sie gemacht haben, fand ich sehr cool, war die Einführung eines Programms namens SAILORS und später AI4ALL, das sich wirklich an Oberstufenschüler und sogar jüngere Schüler richtete, um zu versuchen, ihnen mehr Möglichkeiten in der KI zu bieten,

einschließlich Menschen aus alle Lebensbereiche. Ich würde gerne mehr darüber hören. Dies entspricht dem Geist dieser Diskussion im Jahr 2015. Die Begeisterung für KI begann groß zu werden, aber es begann auch die Rede davon, dass nebenan Killerroboter und Terminatoren auftauchen würden. Andrew, ich war damals Direktor des Stanford AI Lab, und ich dachte: Wir wissen, wie weit wir von der Ankunft von Terminatoren entfernt sind, und das schien eine etwas weit hergeholte Sorge zu sein. Aber ich lebte mein Arbeitsleben mit einer echten Sorge, von der ich das Gefühl hatte, dass niemand darüber sprach, nämlich der mangelnden Repräsentation in der KI. Ich schätze, ich war damals , nachdem Daphne gegangen war, die einzige weibliche Dozentin am Stanford AI Lab. Wir haben nur einen sehr kleinen Anteil von etwa 15 % an weiblichen Doktoranden, und wir sehen wirklich niemanden aus den unterrepräsentierten Minderheitengruppen im Stanford AI-Programm. Dies ist ein nationales oder sogar weltweites Problem, also war es nicht nur Stanford. Ehrlich gesagt ist heute noch viel Arbeit nötig. Exakt. Wie machen wir das? Nun, ich traf mich mit meiner ehemaligen Schülerin Ugawa Sakofski und auch einem langjährigen Dozenten für MINT-Themen, Dr. Rick Summer vom Stanford Pre-Collegiate Study Program, und überlegte, ob ich Highschool-Frauen, junge Highschool-Frauen, dazu einladen sollte Nehmen Sie an einem Sommerprogramm teil, um sie zum Erlernen von KI zu inspirieren, und so begann es 2015. 2017 erhielten wir viel Ermutigung und Unterstützung von Menschen wie Jensen, Lori Hung und Melinda Gates und gründeten eine nationale gemeinnützige Organisation AI4ALL, das sich wirklich dafür einsetzt, die Führungskräfte von morgen für KI auszubilden oder auszubilden. Von Studenten aller Gesellschaftsschichten, insbesondere aus traditionell unterversorgten und unterrepräsentierten Gemeinschaften. Bis heute haben wir im ganzen Land viele Sommercamps und Sommerprogramme durchgeführt, an denen mehr als 15 Universitäten beteiligt sind, und wir verfügen über Online-Lehrpläne, um Studenten sowie College-Pfadprogramme zu ermutigen, die Karriere dieser Studenten weiterhin zu unterstützen, indem sie ihnen Praktika und Praktika anbieten Mentoren. Es handelt sich um ein kontinuierliches Bemühen, Schüler aller Gesellschaftsschichten zu ermutigen. Ich erinnere mich, dass Ihre Gruppe damals diese wirklich coolen T-Shirts gedruckt hat, auf denen die Frage gestellt wurde. KI wird die Welt verändern, wer wird KI verändern? Ich fand die Antwort, sicherzustellen, dass jeder hereinkommen und teilnehmen kann, eine großartige Antwort. Auch heute noch eine wichtige Frage. Das ist ein toller Gedanke und ich denke, das bringt uns zum Ende des Interviews. Irgendwelche abschließenden Gedanken für die Leute, die das sehen? Dennoch ist dies ein sehr junges Feld. Wie du schon sagtest, Andrew, wir sind immer noch mittendrin. Ich habe immer noch das Gefühl, dass es einfach so viele Fragen gibt, an denen ich nach dem Aufwachen voller Vorfreude mit meinen Studenten im Labor arbeiten möchte, und ich denke, dass es für die jungen Leute da draußen, die lernen, einen Beitrag leisten und die KI von morgen gestalten wollen, noch viel mehr Möglichkeiten gibt.