MACHINE LEARNING-1

WEEK 1

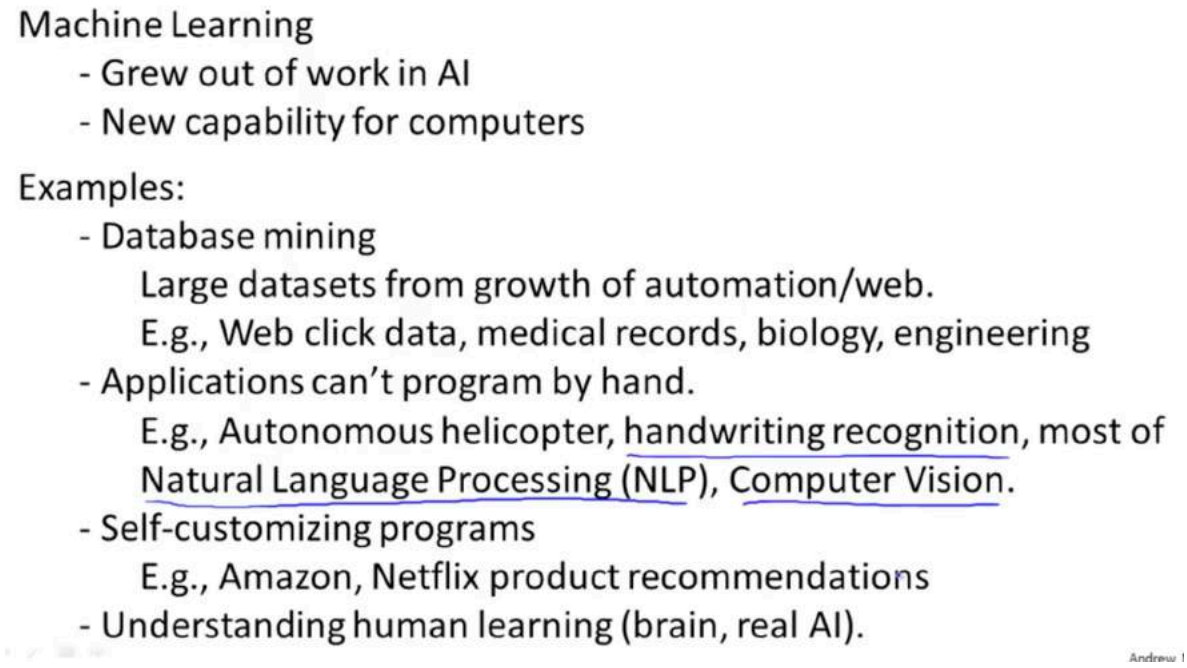
Introduction

Welcome

Here are some other examples of machine learning. There's database mining. One of the reasons machine learning has so pervaded is the growth of the web and the growth of automation All this means that we have much larger data sets than ever before. So, for example tons of Silicon Valley companies are today collecting web click data, also called clickstream data, and are trying to use machine learning algorithms to mine this data to understand the users better and to serve the users better, that's a huge segment of Silicon Valley right now. Medical records. With the advent of automation, we now have electronic medical records, so if we can turn medical records into medical knowledge, then we can start to understand disease better. Computational biology. With automation again, biologists are collecting lots of data about gene sequences, DNA sequences, and so on, and machines running algorithms are giving us a much better understanding of the human genome, and what it means to be human. And in engineering as well, in all fields of engineering, we have larger and larger, and larger and larger data sets, that we're trying to understand using learning algorithms. A second range of machinery applications is ones that we cannot program by hand. So for example, I've worked on autonomous helicopters for many years. We just did not know how to write a computer program to make this helicopter fly by itself. The only thing that worked was having a computer learn by itself how to fly this helicopter.

Every time you go to Amazon or Netflix or iTunes Genius, and it recommends the movies or products and music to you, that's a learning algorithm. If you

think about it they have million users; there is no way to write a million different programs for your million users. The only way to have software give these customized recommendations is to become learn by itself to customize itself to your preferences. Finally learning algorithms are being used today to understand human learning and to understand the brain.



What is Machine Learning?

Two definitions of Machine Learning are offered. Arthur Samuel described it as: "the field of study that gives computers the ability to learn without being explicitly programmed." This is an older, informal definition.

Tom Mitchell provides a more modern definition: "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E."

Example: playing checkers.

E = the experience of playing many games of checkers

T = the task of playing checkers.

P = the probability that the program will win the next game.

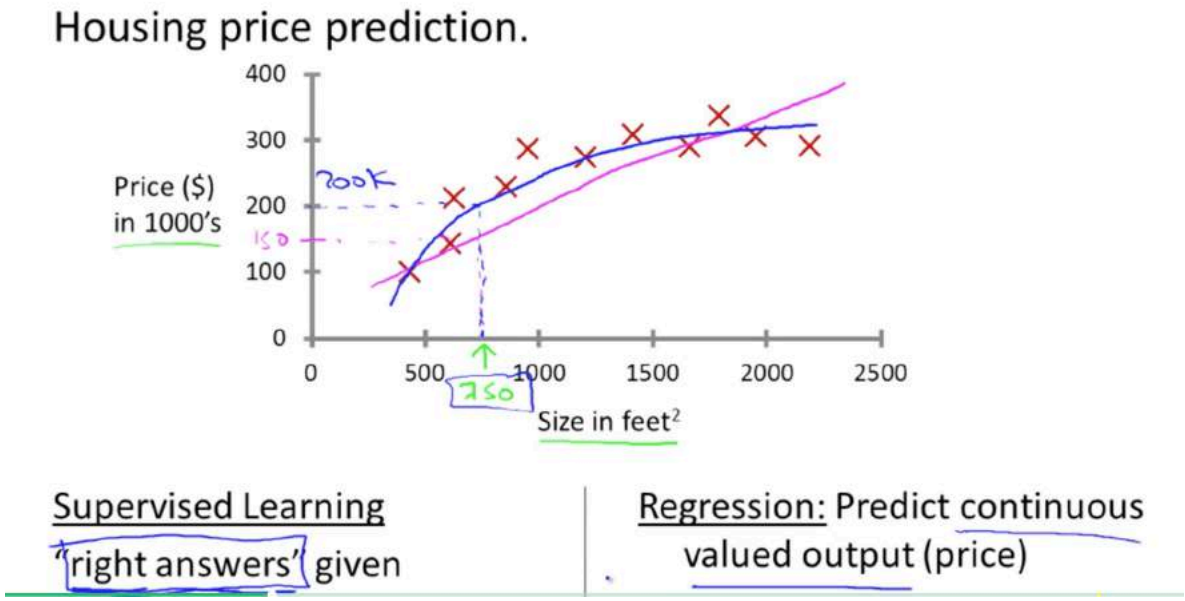
In general, any machine learning problem can be assigned to one of two broad classifications:

Supervised learning and Unsupervised learning.

Supervised learning (Video)

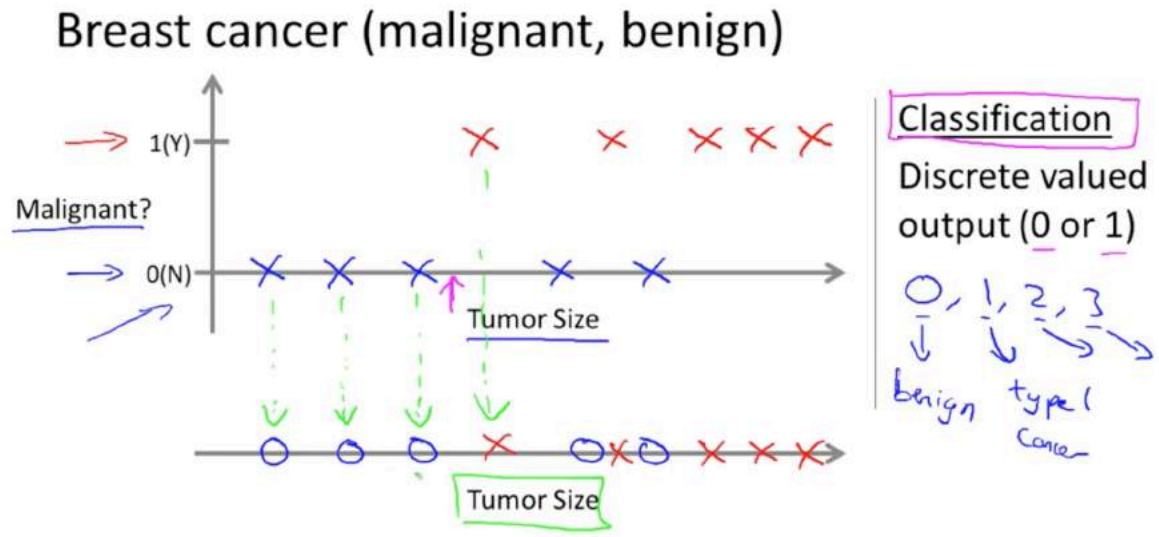
I'll define supervised learning more formally later, but it's probably best to explain or start with an example of what it is and we'll do the formal definition later. Let's say you want to predict housing prices. A while back, a student collected data sets from the Institute of Portland Oregon. And let's say you plot a data set and it looks like this. Here on the horizontal axis, the size of different houses in square feet, and on the vertical axis, the price of different houses in thousands of dollars. So. Given this data, let's say you have a friend who owns a house that is, say 750 square feet and hoping to sell the house and they want to know how much they can get for the house. So how can the learning algorithm help you? One thing a learning algorithm might be able to do is put a straight line through the data or to fit a straight line to the data and, based on that, it looks like maybe the house can be sold for maybe about $150,000. But maybe this isn't the only learning algorithm you can use. There might be a better one. For example, instead of sending a straight line to the data, we might decide that it's better to fit a quadratic function or a second-order polynomial to this data. And if you do that, and make a prediction here, then it looks like, well, maybe we can sell the house for closer to $200,000. One of the things we'll talk about later is how to choose and how to decide do you want to fit a straight line to the data or do you want to fit the quadratic function to the data and there's no fair picking whichever one gives your friend the better house to sell. But each of these would be a fine example of a learning algorithm. So this is an example of a supervised learning algorithm. So this is an example of a supervised learning algorithm. And the term supervised learning refers to the fact that we gave the algorithm a data set in which the "right answers" were given. That is, we gave it a data set of houses in which for every example in this data set, we told it what is the right price so what is

the actual price that, that house sold for and the toss of the algorithm was to just produce more of these right answers such as for this new house, you know, that your friend may be trying to sell. To define with a bit more terminology this is also called a regression problem and by regression problem I mean we're trying to predict a continuous value output. Namely the price. So technically I guess prices can be rounded off to the nearest cent. So maybe prices are actually discrete values, but usually we think of the price of a house as a real number, as a scalar value, as a continuous value number and the term regression refers to the fact that we're trying to predict the sort of continuous values attribute.

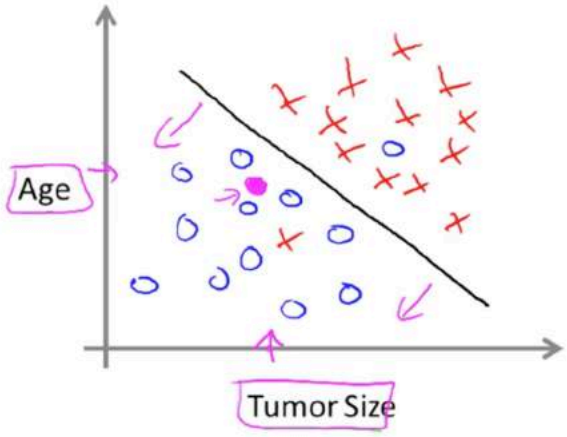


Here's another supervised learning example, some friends and I were actually working on this earlier. Let's see you want to look at medical records and try to predict of a breast cancer as malignant or benign. If someone discovers a breast tumor, a lump in their breast, a malignant tumor is a tumor that is harmful and dangerous and a benign tumor is a tumor that is harmless. So obviously people care a lot about this. Let's see a collected data set and suppose in your data set you have on your horizontal axis the size of the tumor and on the vertical axis I'm going to plot one or zero, yes or no, whether or not these are examples of tumors we've seen before are malignant which is one or zero if not malignant or benign. So let's say our data set looks like this where we saw a tumor of this size that turned out to be benign. One of this size, one of this size. And so on. And sadly we also saw a few malignant tumors, one of that size, one of that size, one of that size... So on. So this example... I have five examples of benign tumors shown down here, and five examples of malignant tumors shown with a vertical axis value of one. And let's say we have a friend who tragically has a breast tumor, and let's say her breast tumor

size is maybe somewhere around this value. The machine learning question is, can you estimate what is the probability, what is the chance that a tumor is malignant versus benign? To introduce a bit more terminology this is an example of a classification problem. The term classification refers to the fact that here we're trying to predict a discrete value output: zero or one, malignant or benign. And it turns out that in classification problems sometimes you can have more than two values for the two possible values for the output. As a concrete example maybe there are three types of breast cancers and so you may try to predict the discrete value of zero, one, two, or three with zero being benign. Benign tumor, so no cancer. And one may mean, type one cancer, like, you have three types of cancer, whatever type one means. And two may mean a second type of cancer, a three may mean a third type of cancer. But this would also be a classification problem, because this other discrete value set of output corresponding to, you know, no cancer, or cancer type one, or cancer type two, or cancer type three. In classification problems there is another way to plot this data. Let me show you what I mean. Let me use a slightly different set of symbols to plot this data. So if tumor size is going to be the attribute that I'm going to use to predict malignancy or benignness, I can also draw my data like this. I'm going to use different symbols to denote my benign and malignant, or my negative and positive examples. So instead of drawing crosses, I'm now going to draw O's for the benign tumors. Like so. And I'm going to keep using X's to denote my malignant tumors. Okay? I hope this is beginning to make sense. All I did was I took, you know, these, my data set on top and I just mapped it down. To this real line like so. And started to use different symbols, circles and crosses, to denote malignant versus benign examples. Now, in this example we use only one feature or one attribute, mainly, the tumor size in order to predict whether the tumor is malignant or benign. In other machine learning problems when we have more than one feature, more than one attribute.



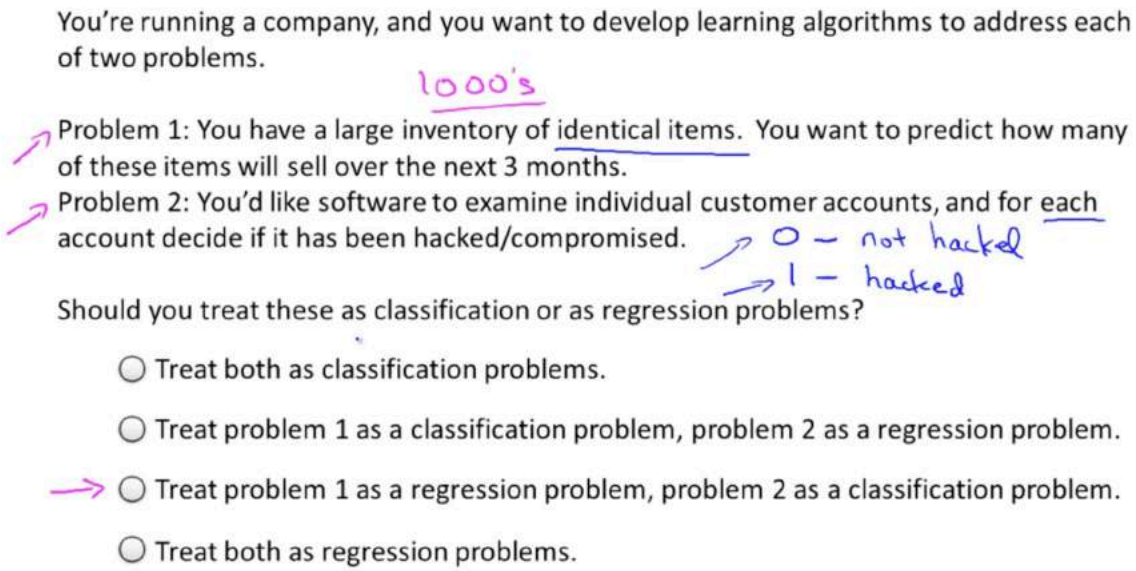
Here's an example. Let's say that instead of just knowing the tumor size, we know both the age of the patients and the tumor size. In that case maybe your data set will look like this where I may have a set of patients with those ages and that tumor size and they look like this. And a different set of patients, they look a little different, whose tumors turn out to be malignant, as denoted by the crosses. So, let's say you have a friend who tragically has a tumor. And maybe, their tumor size and age falls around there. So given a data set like this, what the learning algorithm might do is throw the straight line through the data to try to separate out the malignant tumors from the benign ones and, so the learning algorithm may decide to throw the straight line like that to separate out the two classes of tumors. And. You know, with this, hopefully you can decide that your friend's tumor is more likely to if it's over there, that hopefully your learning algorithm will say that your friend's tumor falls on this benign side and is therefore more likely to be benign than malignant. In this example we had two features, namely, the age of the patient and the size of the tumor. In other machine learning problems we will often have more features, and my friends that work on this problem, they actually use other features like these, which is clump thickness, the clump thickness of the breast tumor. Uniformity of cell size of the tumor. Uniformity of cell shape of the tumor, and so on, and other features as well. And it turns out one of the interest, most interesting learning algorithms that we'll see in this class is a learning algorithm that can deal with, not just two or three or five features, but an infinite number of features. On this slide, I've listed a total of five different features. Right, two on the axes and three more up here. But it turns out that for some learning problems, what you really want is not to use, like, three or five features. But instead, you want to use an infinite number of features, an infinite number of attributes, so that your learning algorithm has lots of attributes or features or cues with which to make those predictions. So how do you deal with an infinite number of features. How do you even store an infinite number of things on the computer when your computer is gonna run out of memory. It turns out that when we talk about an algorithm called the Support Vector Machine, there will be a neat mathematical trick that will allow a computer to deal with an infinite number of features. Imagine that I didn't just write down two features here and three features on the right. But, imagine that I wrote down an infinitely long list, I just kept writing more and more and more features. Like an infinitely long list of features. Turns out, we'll be able to come up with an algorithm that can deal with that. So, just to recap. In this class we'll talk about supervised learning. And the idea is that, in supervised learning, in every example in our data set, we are told what is the "correct answer" that we would have quite liked the algorithms have predicted on that example. Such as the price of the house, or whether a tumor is malignant or benign. We also talked about the regression problem. And by regression, that means that our goal is to predict a continuous valued output. And we talked about the classification problem, where the goal is to predict a discrete value output.



- Clump Thickness
- Uniformity of Cell Size
- Uniformity of Cell Shape

...

Heres a question
Explanation:
For problem one, I would treat this as a regression problem, because if I have, you know, thousands of items, well, I would probably just treat this as a real value, as a continuous value. And treat, therefore, the number of items I sell, as a continuous value. And for the second problem, I would treat that as a classification problem, because I might say, set the value I want to predict with zero, to denote the account has not been hacked. And set the value one to denote an account that has been hacked into. So just like, you know, breast cancer, is, zero is benign, one is malignant. So I might set this be zero or one depending on whether it's been hacked, and have an algorithm try to predict each one of these two discrete values. And because there's a small number of discrete values, I would therefore treat it as a classification problem. So, that's it for supervised learning and in the next video I'll talk about unsupervised learning, which is the other major category of learning algorithms.



Supervised Learning (Transcript)

In supervised learning, we are given a data set and already know what our correct output should look like, having the idea that there is a relationship between the input and the output.

Supervised learning problems are categorized into "regression" and "classification" problems. In a regression problem, we are trying to predict results within a continuous output, meaning that we are trying to map input variables to some continuous function. In a classification problem, we are instead trying to predict results in a discrete output. In other words, we are trying to map input variables into discrete categories.

Example 1:

Given data about the size of houses on the real estate market, try to predict their price. Price as a function of size is a continuous output, so this is a regression problem.

We could turn this example into a classification problem by instead making our output about whether the house "sells for more or less than the asking price." Here we are classifying the houses based on price into two discrete categories.

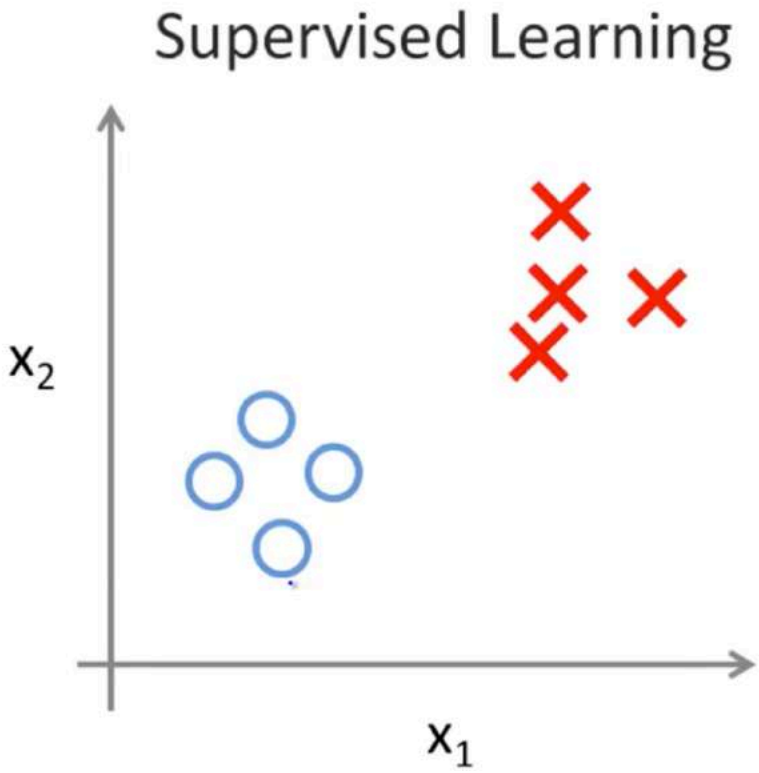
Example 2:

(a) Regression - Given a picture of a person, we have to predict their age on the basis of the given picture

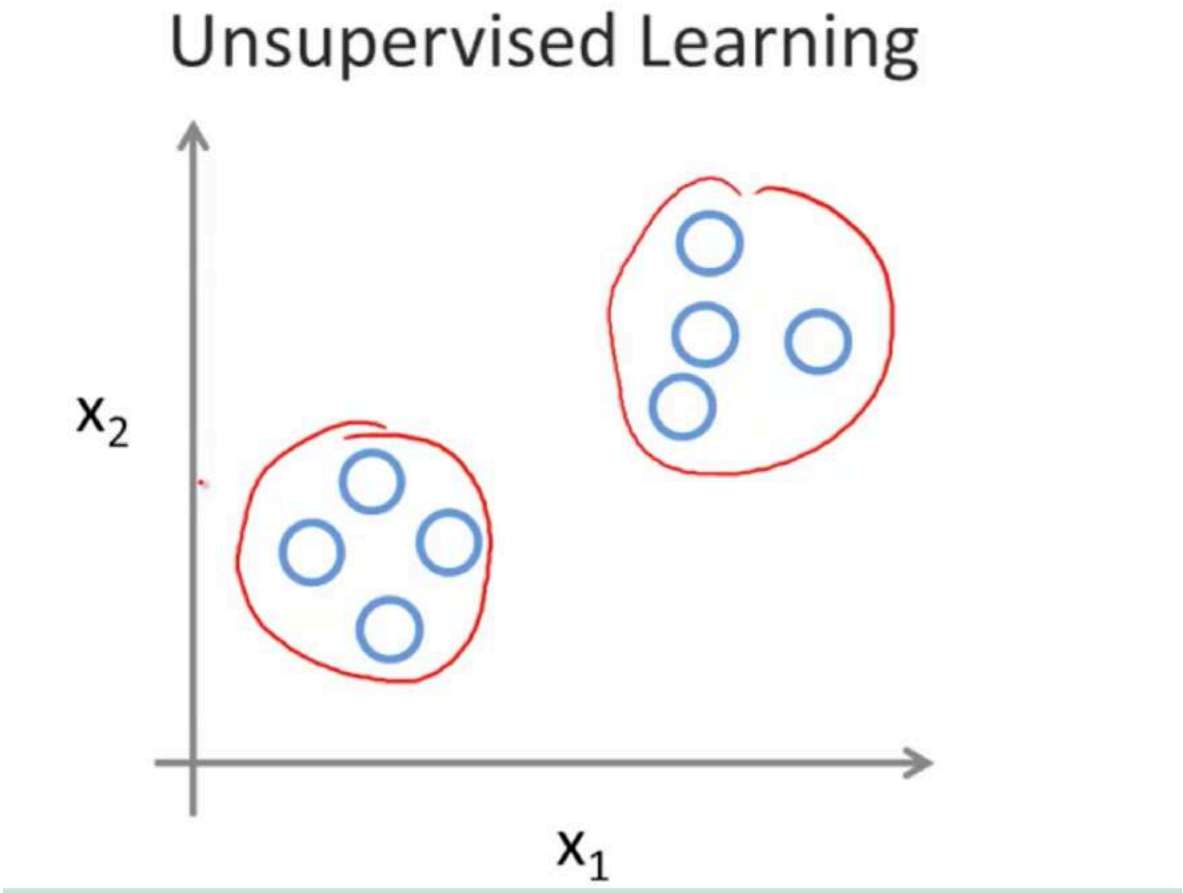
(b) Classification - Given a patient with a tumor, we have to predict whether the tumor is malignant or benign.

Unsupervised Learning (Video)

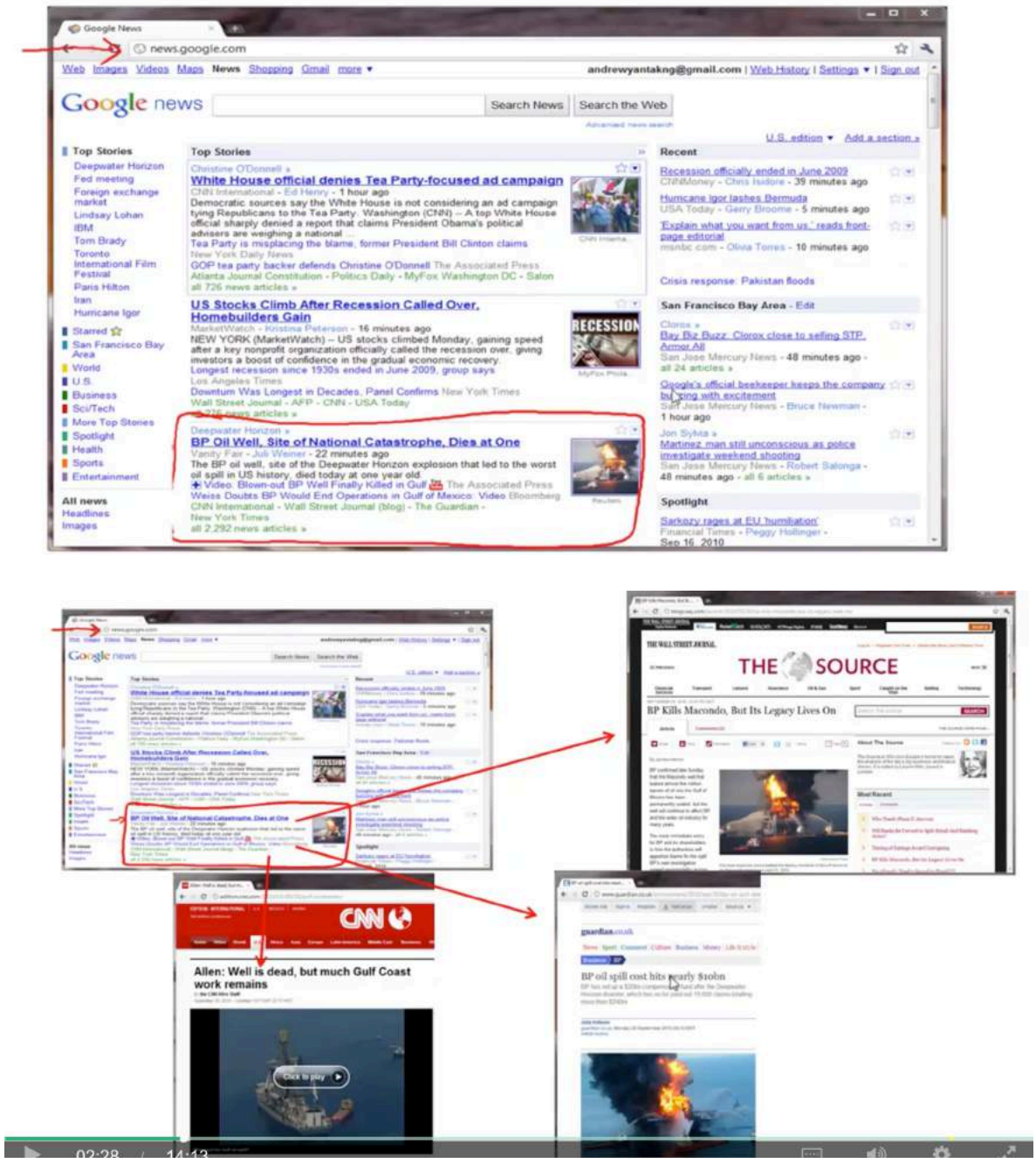
In this video, we'll talk about the second major type of machine learning problem, called Unsupervised Learning. In the last video, we talked about Supervised Learning. Back then, recall data sets that look like this, where each example was labeled either as a positive or negative example, whether it was a benign or a malignant tumor. So for each example in Supervised Learning, we were told explicitly what is the so-called right answer, whether it's benign or malignant.



In Unsupervised Learning, we're given data that looks different than data that looks like this that doesn't have any labels or that all has the same label or really no labels. So we're given the data set and we're not told what to do with it and we're not told what each data point is. Instead we're just told, here is a data set. Can you find some structure in the data? Given this data set, an Unsupervised Learning algorithm might decide that the data lives in two different clusters. And so there's one cluster and there's a different cluster. And yes, Supervised Learning algorithm may break these data into these two separate clusters. So this is called a clustering algorithm. And this turns out to be used in many places.

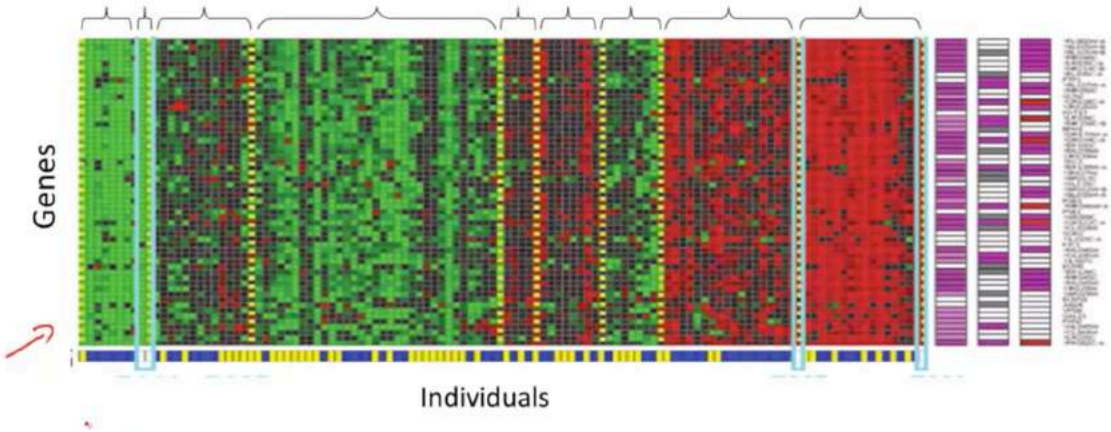


One example where clustering is used is in Google News and if you have not seen this before, you can actually go to this URL news.google.com to take a look. What Google News does is everyday it goes and looks at tens of thousands or hundreds of thousands of new stories on the web and it groups them into cohesive news stories. For example, let's look here. The URLs here link to different news stories about the BP Oil Well story. So, let's click on one of these URL's and we'll click on one of these URL's. What I'll get to is a web page like this. Here's a Wall Street Journal article about, you know, the BP Oil Well Spill stories of "BP Kills Macondo", which is a name of the spill and if you click on a different URL from that group then you might get the different story. Here's the CNN story about a game, the BP Oil Spill, and if you click on yet a third link, then you might get a different story. Here's the UK Guardian story about the BP Oil Spill. So what Google News has done is look for tens of thousands of news stories and automatically cluster them together. So, the news stories that are all about the same topic get displayed together.



It turns out that clustering algorithms and Unsupervised Learning algorithms are used in many other problems as well. Here's one on understanding genomics. Here's an example of DNA microarray data. The idea is put a group of different individuals and for each of them, you measure how much they do or do not have a certain gene. Technically you measure how much certain genes are expressed. So these colors, red, green, gray and so on, they show the degree to which different individuals do or do not have a specific gene. And what you can do is then run a clustering algorithm to group individuals into

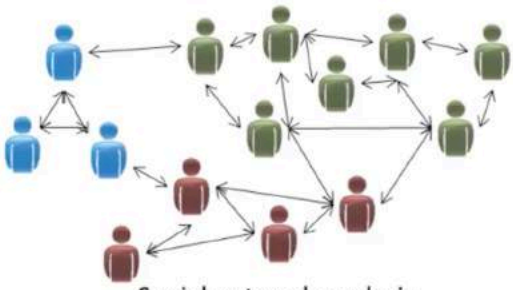
different categories or into different types of people. So this is Unsupervised Learning because we're not telling the algorithm in advance that these are type 1 people, those are type 2 persons, those are type 3 persons and so on and instead what were saying is yeah here's a bunch of data. I don't know what's in this data. I don't know who's and what type. I don't even know what the different types of people are, but can you automatically find structure in the data from the you automatically cluster the individuals into these types that I don't know in advance? Because we're not giving the algorithm the right answer for the examples in my data set, this is Unsupervised Learning.



Unsupervised Learning or clustering is used for a bunch of other applications. It's used to organize large computer clusters. I had some friends looking at large data centers, that is large computer clusters and trying to figure out which machines tend to work together and if you can put those machines together, you can make your data center work more efficiently. This second application is on social network analysis. So given knowledge about which friends you email the most or given your Facebook friends or your Google+ circles, can we automatically identify which are cohesive groups of friends, also which are groups of people that all know each other? Market segmentation. Many companies have huge databases of customer information. So, can you look at this customer data set and automatically discover market segments and automatically group your customers into different market segments so that you can automatically and more efficiently sell or market your different market segments together? Again, this is Unsupervised Learning because we have all this customer data, but we don't know in advance what are the market segments and for the customers in our data set, you know, we don't know in advance who is in market segment one, who is in market segment two, and so on. But we have to let the algorithm discover all this just from the data. Finally, it turns out that Unsupervised Learning is also used for surprisingly astronomical data analysis and these clustering algorithms gives surprisingly interesting useful theories of how galaxies are formed. All of these

are examples of clustering, which is just one type of Unsupervised Learning.

Organize computing clusters



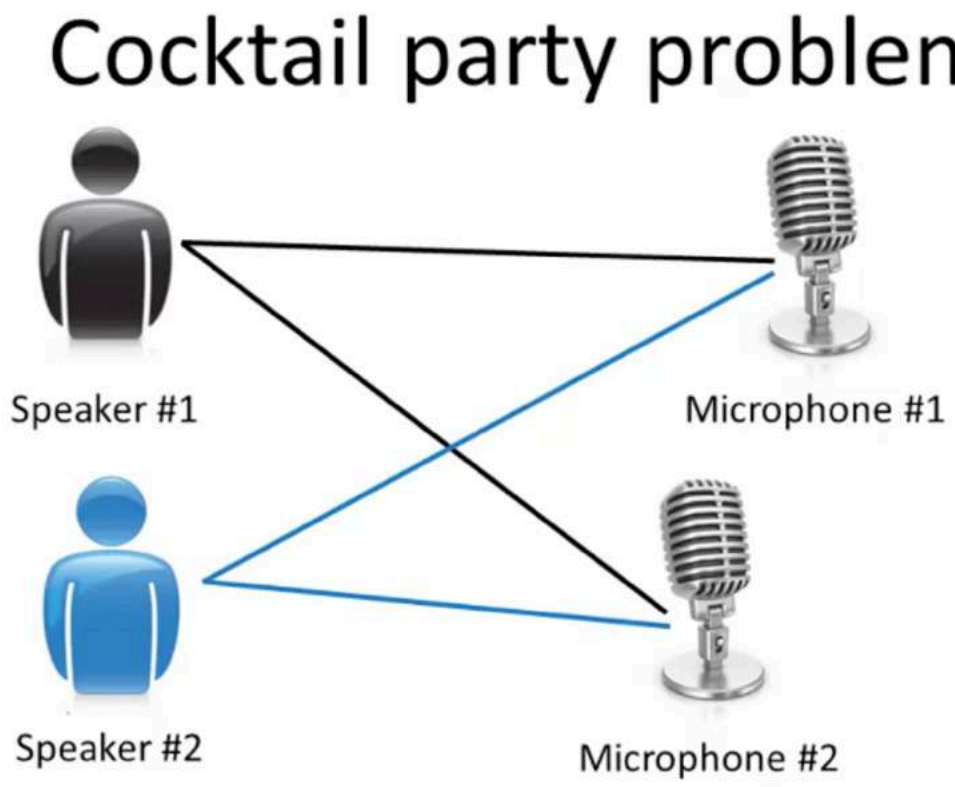
Social network analysis

Market segmentation

Astronomical data analysis

Andrew Ng

Let me tell you about another one. I'm gonna tell you about the cocktail party problem. So, you've been to cocktail parties before, right? Well, you can imagine there's a party, room full of people, all sitting around, all talking at the same time and there are all these overlapping voices because everyone is talking at the same time, and it is almost hard to hear the person in front of you. So maybe at a cocktail party with two people, two people talking at the same time, and it's a somewhat small cocktail party. And we're going to put two microphones in the room so there are microphones, and because these microphones are at two different distances from the speakers, each microphone records a different combination of these two speaker voices. Maybe speaker one is a little louder in microphone one and maybe speaker two is a little bit louder on microphone 2 because the 2 microphones are at different positions relative to the 2 speakers, but each microphone would cause an overlapping combination of both speakers' voices.



So here's an actual recording of two speakers recorded by a researcher. Let me play for you the first, what the first microphone sounds like. One (uno), two (dos), three (tres), four (cuatro), five (cinco), six (seis), seven (siete), eight (ocho), nine (nueve), ten (y diez). All right, maybe not the most interesting cocktail party, there's two people counting from one to ten in two languages but you know. What you just heard was the first microphone recording, here's the second recording. Uno (one), dos (two), tres (three), cuatro (four), cinco (five), seis (six), siete (seven), ocho (eight), nueve (nine) y diez (ten). So we can do, is take these two microphone recorders and give them to an Unsupervised Learning algorithm called the cocktail party algorithm, and tell the algorithm - find structure in this data for you. And what the algorithm will do is listen to these audio recordings and say, you know it sounds like the two audio recordings are being added together or that have being summed together to produce these recordings that we had. Moreover, what the cocktail party algorithm will do is separate out these two audio sources that were being added or being summed together to form other recordings and, in fact, here's the first output of the cocktail party algorithm. One, two, three, four, five, six, seven, eight, nine, ten. So, I separated out the English voice in one of the recordings. And here's the second of it. Uno, dos, tres, quatro, cinco, seis, siete, ocho, nueve y diez. Not too bad, to give you one more example, here's another recording of another similar situation, here's the first microphone : One, two, three, four, five, six, seven, eight, nine, ten. OK so the poor guy's gone home from the cocktail party and he 's now sitting in a room by himself talking to his radio. Here's the second microphone recording. One, two, three, four, five, six, seven, eight, nine, ten. When you give these two microphone

recordings to the same algorithm, what it does, is again say, you know, it sounds like there are two audio sources, and moreover, the album says, here is the first of the audio sources I found. One, two, three, four, five, six, seven, eight, nine, ten. So that wasn't perfect, it got the voice, but it also got a little bit of the music in there. Then here's the second output to the algorithm. Not too bad, in that second output it managed to get rid of the voice entirely. And just, you know, cleaned up the music, got rid of the counting from one to ten.
TOP M1: Hears both TOP M2: Hears both from different position TOP O1: 1 to 10 English numbers TOP O2: Numbers in other language
DOWN M1: Hears both DOWN M2: Hears both from different position DOWN O1: 1 to 10 English numbers with some music DOWN O2: Pure Music

So you might look at an Unsupervised Learning algorithm like this and ask how complicated this is to implement this, right? It seems like in order to, you know, build this application, it seems like to do this audio processing you need to write a ton of code or maybe link into like a bunch of synthesizer Java libraries that process audio, seems like a really complicated program, to do this audio, separating out audio and so on. It turns out the algorithm, to do what you just heard, that can be done with one line of code - shown right here. It take researchers a long time to come up with this line of code. I'm not saying this is an easy problem, But it turns out that when you use the right programming environment, many learning algorithms can be really short programs. So this is also why in this class we're going to use the Octave programming environment. Octave, is free open source software, and using a tool like Octave or Matlab, many learning algorithms become just a few lines of code to implement. Later in this class, I'll just teach you a little bit about how to use Octave and you'll be

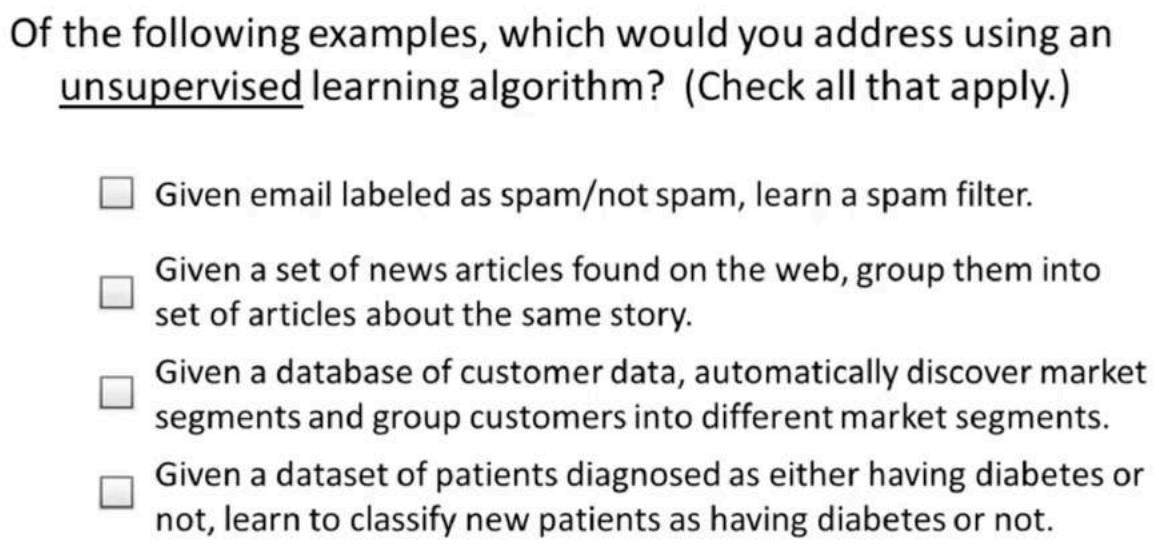
implementing some of these algorithms in Octave. Or if you have Matlab you can use that too. It turns out the Silicon Valley, for a lot of machine learning algorithms, what we do is first prototype our software in Octave because software in Octave makes it incredibly fast to implement these learning algorithms. Here each of these functions like for example the SVD function that stands for singular value decomposition; but that turns out to be a linear algebra routine, that is just built into Octave. If you were trying to do this in C++ or Java, this would be many many lines of code linking complex C++ or Java libraries. So, you can implement this stuff as C++ or Java or Python, it's just much more complicated to do so in those languages. What I've seen after having taught machine learning for almost a decade now, is that, you learn much faster if you use Octave as your programming environment, and if you use Octave as your learning tool and as your prototyping tool, it'll let you learn and prototype learning algorithms much more quickly. And in fact what many people will do to in the large Silicon Valley companies is in fact, use an algorithm like Octave to first prototype the learning algorithm, and only after you've gotten it to work, then you migrate it to C++ or Java or whatever. It turns out that by doing things this way, you can often get your algorithm to work much faster than if you were starting out in C++. So, I know that as an instructor, I get to say "trust me on this one" only a finite number of times, but for those of you who've never used these Octave type programming environments before, I am going to ask you to trust me on this one, and say that you, you will, I think your time, your development time is one of the most valuable resources. And having seen lots of people do this, I think you as a machine learning researcher, or machine learning developer will be much more productive if you learn to start in prototype, to start in Octave, in some other language.

Cocktail party problem algorithm

```
[W,s,v] = svd((repmat(sum(x.*x,1),size(x,1),1).*x)*x');
```

Finally, to wrap up this video, I have one quick review question for you. We talked about Unsupervised Learning, which is a learning setting where you give the algorithm a ton of data and just ask it to find structure in the data for us. Of the following four examples, which ones, which of these four do you think would will be an Unsupervised Learning algorithm as opposed to Supervised Learning problem. For each of the four check boxes on the left, check the ones

for which you think Unsupervised Learning algorithm would be appropriate and then click the button on the lower right to check your answer. B and C are correct and are examples of Unsupervised Learning
Explanation:
So, hopefully, you've remembered the spam folder problem. If you have labeled data, you know, with spam and non-spam e-mail, we'd treat this as a Supervised Learning problem. The news story example, that's exactly the Google News example that we saw in this video, we saw how you can use a clustering algorithm to cluster these articles together so that's Unsupervised Learning. The market segmentation example I talked a little bit earlier, you can do that as an Unsupervised Learning problem because I am just gonna get my algorithm data and ask it to discover market segments automatically. And the final example, diabetes, well, that's actually just like our breast cancer example from the last video. Only instead of, you know, good and bad cancer tumors or benign or malignant tumors we instead have diabetes or not and so we will use that as a supervised, we will solve that as a Supervised Learning problem just like we did for the breast tumor data.



So, that's it for Unsupervised Learning and in the next video, we'll delve more into specific learning algorithms and start to talk about just how these algorithms work and how we can, how you can go about implementing them.

Unsupervised Learning (Transcript)

Unsupervised learning allows us to approach problems with little or no idea what our results should look like. We can derive structure from data where we don't necessarily know the effect of the variables.

We can derive this structure by clustering the data based on relationships among the variables in the data.

With unsupervised learning there is no feedback based on the prediction results.

Example:

Clustering: Take a collection of 1,000,000 different genes, and find a way to automatically group these genes into groups that are somehow similar or related by different variables, such as lifespan, location, roles, and so on.

Non-clustering: The "Cocktail Party Algorithm", allows you to find structure in a chaotic environment. (i.e. identifying individual voices and music from a mesh of sounds at a cocktail party).

Review

Quiz: Introduction

Introduction

1. A computer program is said to learn from experience E with

respect to some task T and some performance measure P if its

performance on T, as measured by P, improves with experience E.

Suppose we feed a learning algorithm a lot of historical weather

data, and have it learn to predict weather. In this setting, what is T?

- [ ] None of these.
- [ ] The process of the algorithm examining a large amount of historical weather data.
- [ ] The probability of it correctly predicting a future date's weather.
- [x] The weather prediction task.

2. Suppose you are working on weather prediction, and your weather station makes one of three predictions for each day's weather: Sunny, Cloudy or Rainy. You'd like to use a learning algorithm to predict tomorrow's weather.

Would you treat this as a classification or a regression problem?

- [ ] Regression
- [x] Classification

3. Suppose you are working on stock market prediction. You would like to predict whether or not a certain company will declare bankruptcy within the next 7 days (by training on data of similar companies that had previously been at risk of bankruptcy). Would you treat this as a classification or a regression problem?

- [x] Classification
- [ ] Regression

4. Some of the problems below are best addressed using a supervised learning algorithm, and the others with an unsupervised learning algorithm. Which of the following would you apply supervised learning to? (Select all that apply.) In each case, assume some

appropriate

dataset is available for your algorithm to learn from.

- ○ Given a large dataset of medical records from patients suffering from heart disease, try to learn whether there might be different clusters of such patients for which we might tailor separate treatments.
- ✓ Have a computer examine an audio clip of a piece of music, and classify whether or not there are vocals (i.e., a human voice singing) in that audio clip, or if it is a clip of only musical instruments (and no vocals).
- ○ Given data on how 1000 medical patients respond to an experimental drug (such as effectiveness of the treatment, side effects, etc.), discover whether there are different categories or "types" of patients in terms of how they respond to the drug, and if so what these categories are.
- ✓ Given genetic (DNA) data from a person, predict the odds of him/her developing diabetes over the next 10 years.

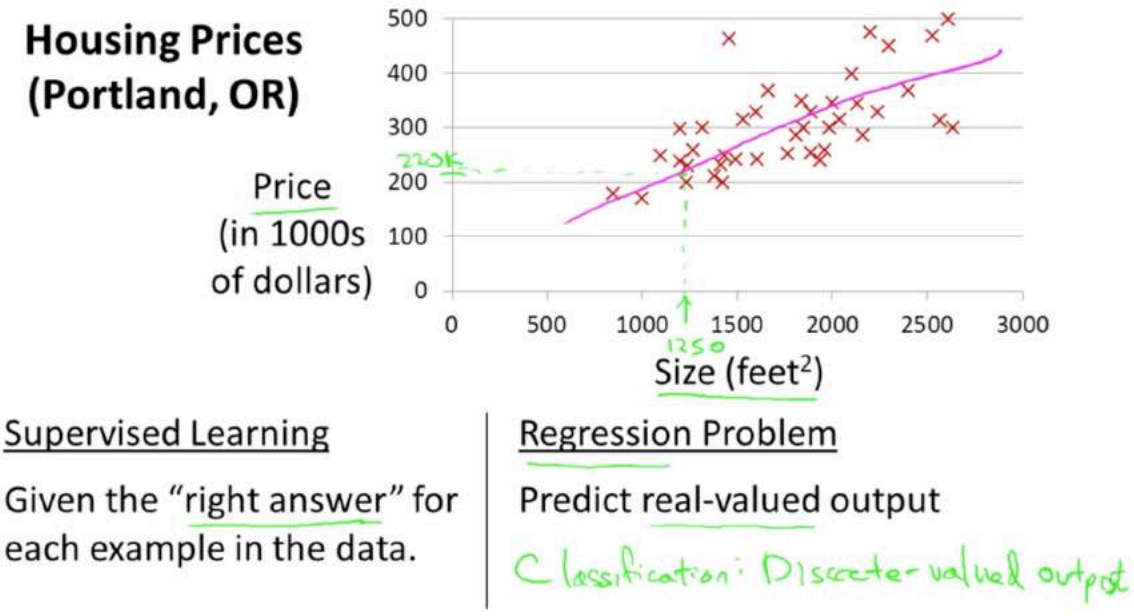
5. Which of these is a reasonable definition of machine learning?

- ○ Machine learning is the science of programming computers.
- ○ Machine learning learns from labeled data.
- ○ Machine learning is the field of allowing robots to act intelligently.
- ✓ Machine learning is the field of study that gives computers the ability to learn without being explicitly programmed.

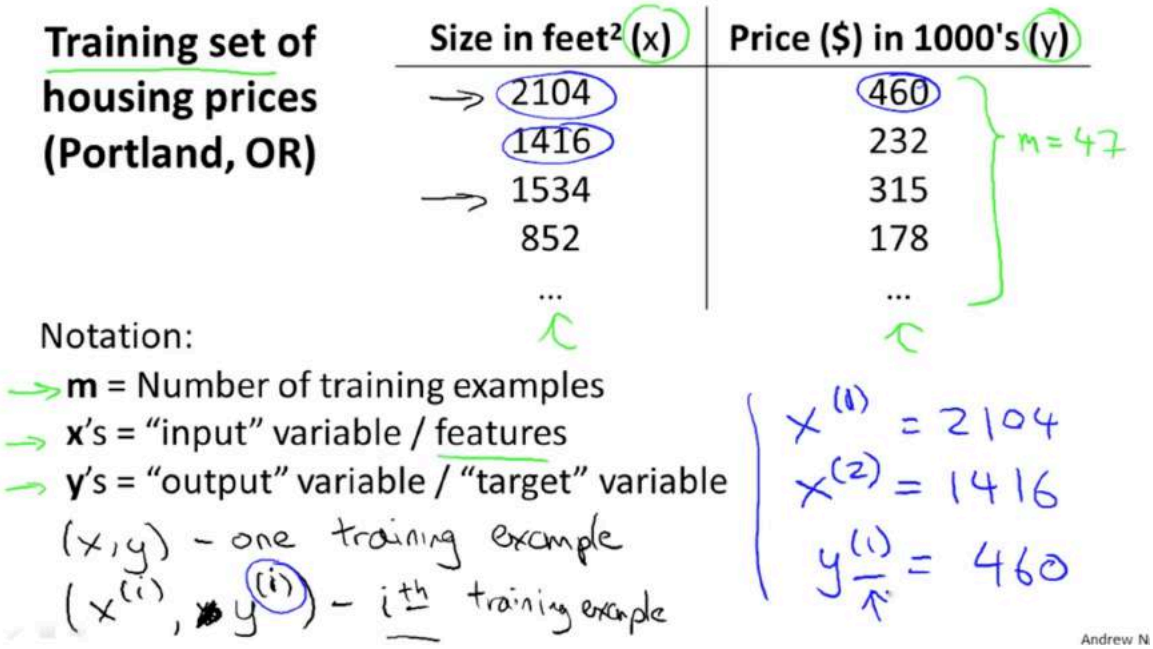
Model and Cost function

Model Representation (Video)

Our first learning algorithm will be linear regression. In this video, you'll see what the model looks like and more importantly you'll see what the overall process of supervised learning looks like. Let's use some motivating example of predicting housing prices. We're going to use a data set of housing prices from the city of Portland, Oregon. And here I'm gonna plot my data set of a number of houses that were different sizes that were sold for a range of different prices. Let's say that given this data set, you have a friend that's trying to sell a house and let's see if friend's house is size of 1250 square feet and you want to tell them how much they might be able to sell the house for. Well one thing you could do is fit a model. Maybe fit a straight line to this data. Looks something like that and based on that, maybe you could tell your friend that let's say maybe he can sell the house for around $220,000. So this is an example of a supervised learning algorithm. And it's supervised learning because we're given the, quotes, "right answer" for each of our examples. Namely we're told what was the actual house, what was the actual price of each of the houses in our data set were sold for and moreover, this is an example of a regression problem where the term regression refers to the fact that we are predicting a real-valued output namely the price. And just to remind you the other most common type of supervised learning problem is called the classification problem where we predict discrete-valued outputs such as if we are looking at cancer tumors and trying to decide if a tumor is malignant or benign. So that's a zero-one valued discrete output.



More formally, in supervised learning, we have a data set and this data set is called a training set. So for housing prices example, we have a training set of different housing prices and our job is to learn from this data how to predict prices of the houses. Let's define some notation that we're using throughout this course. We're going to define quite a lot of symbols. It's okay if you don't remember all the symbols right now but as the course progresses it will be useful [inaudible] convenient notation. So I'm gonna use lower case m throughout this course to denote the number of training examples. So in this data set, if I have, you know, let's say 47 rows in this table. Then I have 47 training examples and m equals 47. Let me use lowercase x to denote the input variables often also called the features. That would be the x is here, it would the input features. And I'm gonna use y to denote my output variables or the target variable which I'm going to predict and so that's the second column here. [inaudible] notation, I'm going to use (x, y) to denote a single training example. So, a single row in this table corresponds to a single training example and to refer to a specific training example, I'm going to use this notation x(i) comma gives me y(i) And, we're going to use this to refer to the ith training example. So this superscript i over here, this is not exponentiation right? This (x(i), y(i)), the superscript i in parentheses that's just an index into my training set and refers to the ith row in this table, okay? So this is not x to the power of i, y to the power of i. Instead (x(i), y(i)) just refers to the ith row of this table. So for example, x(1) refers to the input value for the first training example so that's 2104. That's this x in the first row. x(2) will be equal to 1416 right? That's the second x and y(1) will be equal to 460. The first, the y value for my first training example, that's what that (1) refers to.



Question

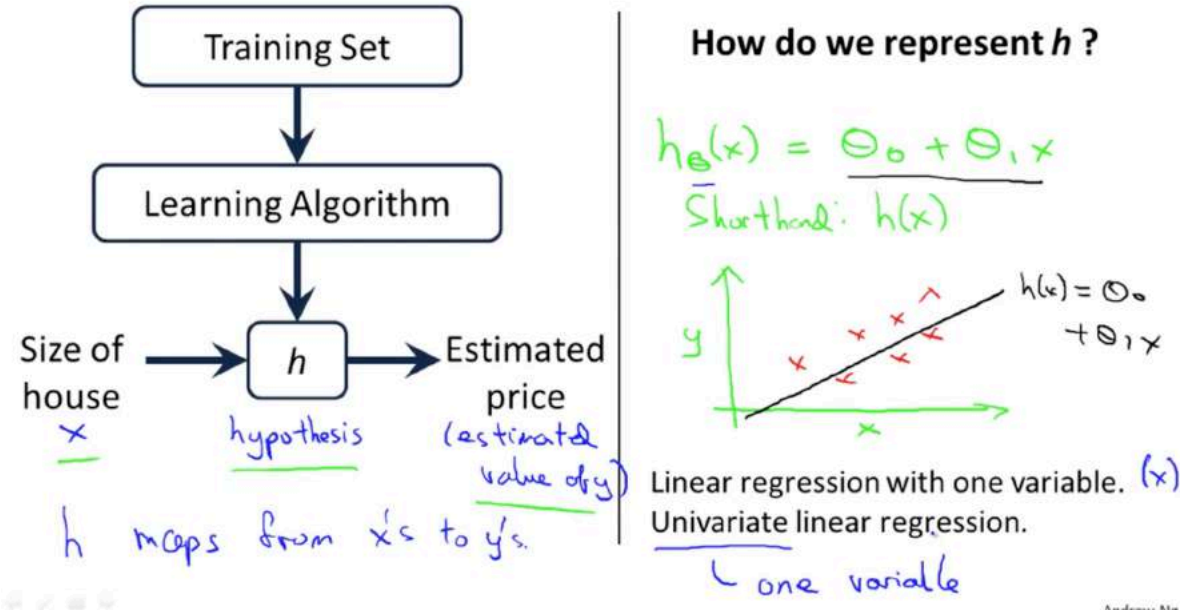
Consider the training set shown below. $(x^{(i)}, y^{(i)})$ is the i^{th} training example. What is $y^{(3)}$?

Size in feet^2 (x)	Price ($) in 1000's (y)
2104	460
1416	232
1534	315
852	178
...	...

- 1416
- 1534
- 315
- 0

So here's how this supervised learning algorithm works. We saw that with the training set like our training set of housing prices and we feed that to our learning algorithm. Is the job of a learning algorithm to then output a function which by convention is usually denoted lowercase h and h stands for hypothesis And what the job of the hypothesis is, is, is a function that takes as input the size of a house like maybe the size of the new house your friend's trying to sell so it takes in the value of x and it tries to output the estimated value of y for the corresponding house. So h is a function that maps from x's to y's. People often ask me, you know, why is this function called hypothesis. Some of you may know the meaning of the term hypothesis, from the dictionary or from science or whatever. It turns out that in machine learning, this is a name that was used in the early days of machine learning and it kinda stuck. 'Cause maybe not a great name for this sort of function, for mapping from sizes of houses to the predictions, that you know.... I think the term hypothesis, maybe isn't the best possible name for this, but this is the standard terminology that people use in machine learning. So don't worry too much about why people call it that. When designing a learning algorithm, the next thing we need to decide is how do we represent this hypothesis h. For this and the next few videos, I'm going to choose our initial choice , for representing the hypothesis, will be the following. We're going to represent h as follows. And we will write this as h theta(x) equals theta0 plus theta1 of x.

And as a shorthand, sometimes instead of writing, you know, h subscript theta of x, sometimes there's a shorthand, I'll just write as a h of x. But more often I'll write it as a subscript theta over there. And plotting this in the pictures, all this means is that, we are going to predict that y is a linear function of x. Right, so that's the data set and what this function is doing, is predicting that y is some straight line function of x. That's h of x equals theta 0 plus theta 1 x, okay? And why a linear function? Well, sometimes we'll want to fit more complicated, perhaps non-linear functions as well. But since this linear case is the simple building block, we will start with this example first of fitting linear functions, and we will build on this to eventually have more complex models, and more complex learning algorithms. Let me also give this particular model a name. This model is called linear regression or this, for example, is actually linear regression with one variable, with the variable being x. Predicting all the prices as functions of one variable X. And another name for this model is univariate linear regression. And univariate is just a fancy way of saying one variable. So, that's linear regression. In the next video we'll start to talk about just how we go about implementing this model.

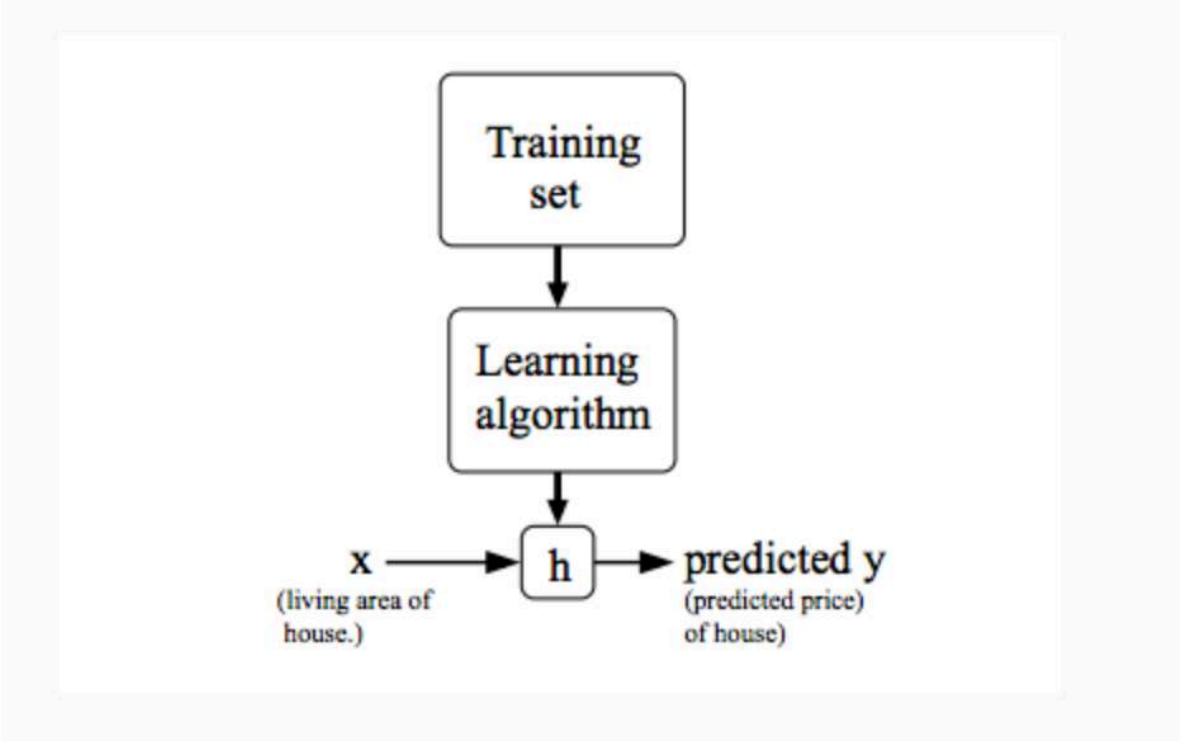


Model Representation (Transcript)

To establish notation for future use, we'll use x(i) to denote the "input" variables (living area in this example), also called input features, and y(i) to

denote the “output” or target variable that we are trying to predict (price). A pair (x(i),y(i)) is called a training example, and the dataset that we’ll be using to learn—a list of m training examples (x(i),y(i));i=1,...,m—is called a training set. Note that the superscript “(i)” in the notation is simply an index into the training set, and has nothing to do with exponentiation. We will also use X to denote the space of input values, and Y to denote the space of output values. In this example, X = Y = $\mathbb{R}$.

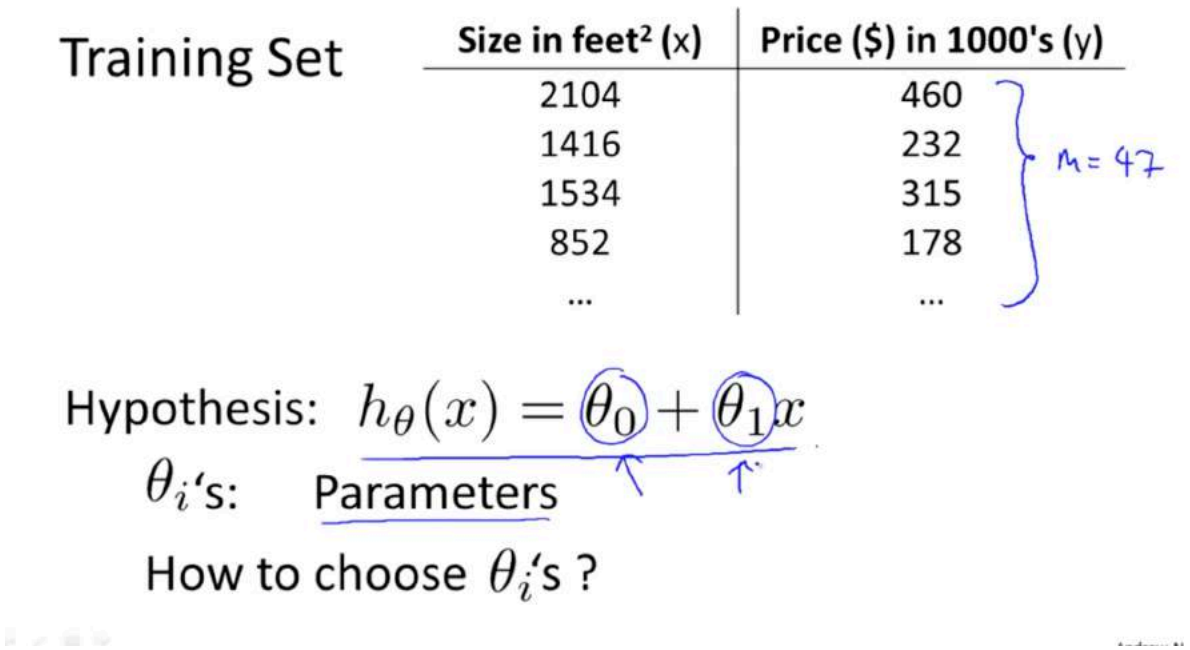
To describe the supervised learning problem slightly more formally, our goal is, given a training set, to learn a function h : X → Y so that h(x) is a “good” predictor for the corresponding value of y. For historical reasons, this function h is called a hypothesis. Seen pictorially, the process is therefore like this:



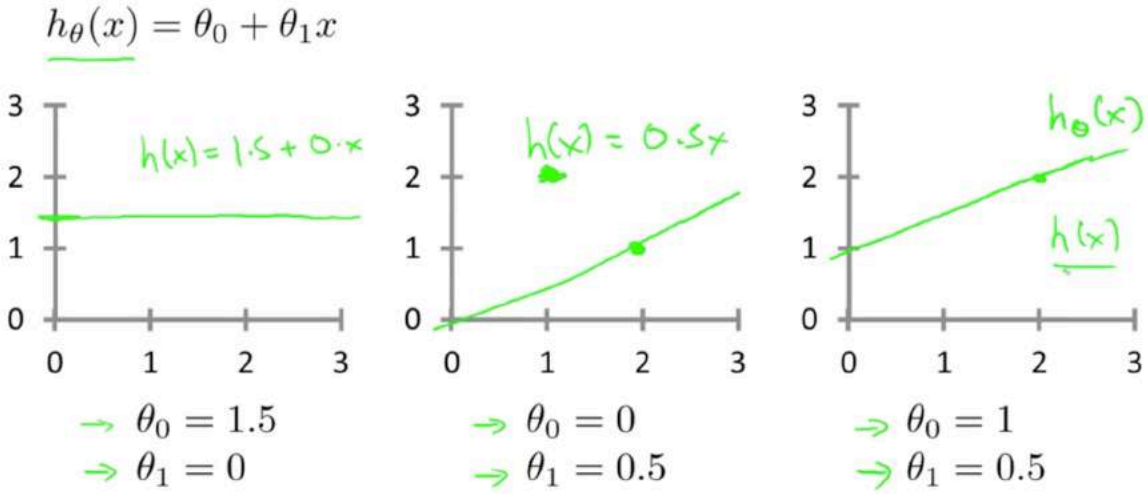
When the target variable that we’re trying to predict is continuous, such as in our housing example, we call the learning problem a regression problem. When y can take on only a small number of discrete values (such as if, given the living area, we wanted to predict if a dwelling is a house or an apartment, say), we call it a classification problem.

Cost Function (Video)

In this video we'll define something called the cost function, this will let us figure out how to fit the best possible straight line to our data. In linear progression, we have a training set that I showed here remember on notation M was the number of training examples, so maybe m equals 47. And the form of our hypothesis, which we use to make predictions is this linear function. To introduce a little bit more terminology, these theta zero and theta one, they stabilize what I call the parameters of the model.

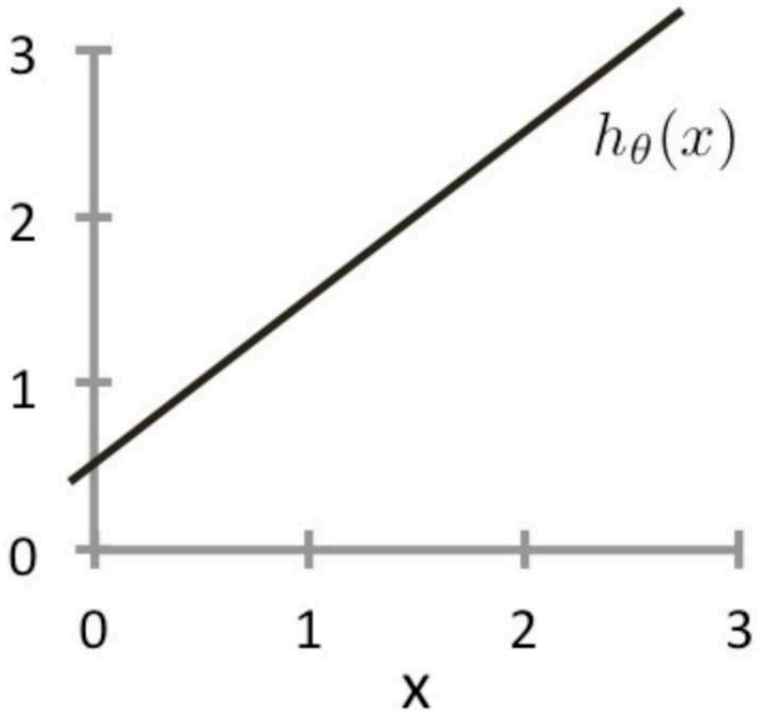


And what we're going to do in this video is talk about how to go about choosing these two parameter values, theta 0 and theta 1. With different choices of the parameter's theta 0 and theta 1, we get different hypothesis, different hypothesis functions. I know some of you will probably be already familiar with what I am going to do on the slide, but just for review, here are a few examples. If theta 0 is 1.5 and theta 1 is 0, then the hypothesis function will look like this. Because your hypothesis function will be h of x equals 1.5 plus 0 times x which is this constant value function which is flat at 1.5. If theta0 = 0, theta1 = 0.5, then the hypothesis will look like this, and it should pass through this point 2,1 so that you now have h(x). Or really h of theta(x), but sometimes I'll just omit theta for brevity. So h(x) will be equal to just 0.5 times x, which looks like that. And finally, if theta zero equals one, and theta one equals 0.5, then we end up with a hypothesis that looks like this. Let's see, it should pass through the two-two point. Like so, and this is my new vector of x, or my new h subscript theta of x. Whatever way you remember, I said that this is h subscript theta of x, but that's a shorthand, sometimes I'll just write this as h of x.



Question

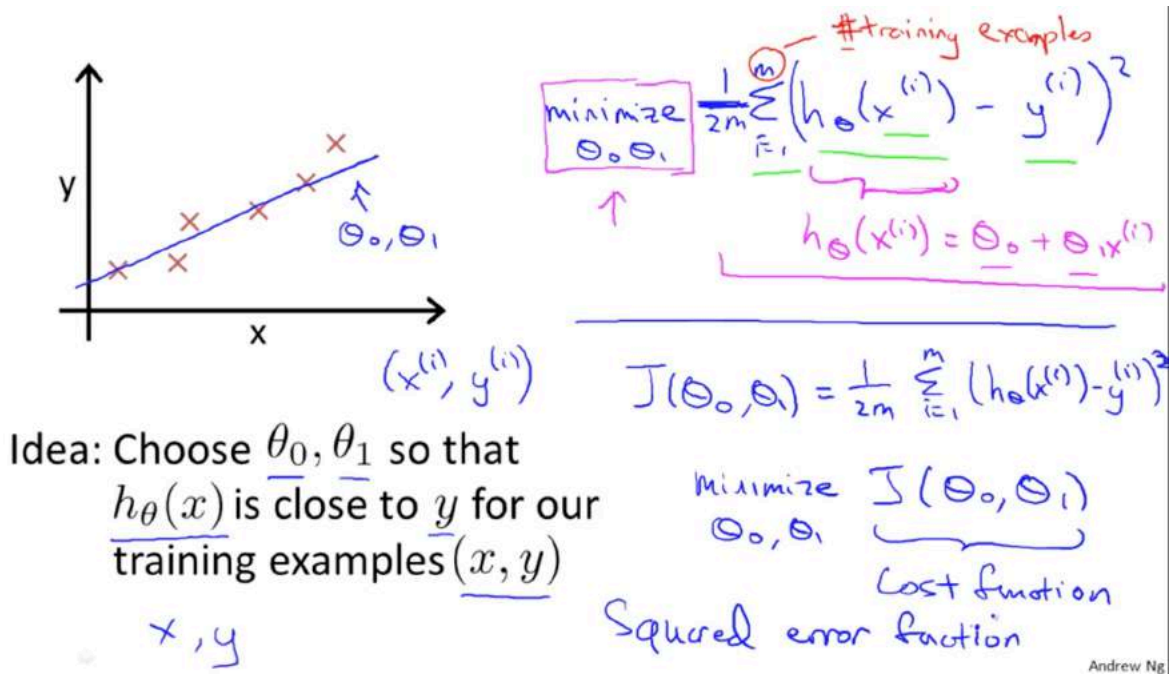
Consider the plot below of $h_\theta(x) = \theta_0 + \theta_1 x$. What are θ_0 and θ_1?



- ○ $\theta_0 = 0, \theta_1 = 1$
- ◉ $\theta_0 = 0.5, \theta_1 = 1$
- ○ $\theta_0 = 1, \theta_1 = 0.5$
- ○ $\theta_0 = 1, \theta_1 = 1$

In linear regression, we have a training set, like maybe the one I've plotted here. What we want to do, is come up with values for the parameters theta zero and theta one so that the straight line we get out of this, corresponds to a straight line that somehow fits the data well, like maybe that line over there. So, how do we come up with values, theta zero, theta one, that corresponds to a good fit to the data? The idea is we get to choose our parameters theta 0, theta 1 so that h of x, meaning the value we predict on input x, that this is at least close to the values y for the examples in our training set, for our training examples. So in our training set, we've given a number of examples where we know X decides the wholes and we know the actual price is was sold for. So, let's try to choose values for the parameters so that, at least in the training set, given the X in the training set we make reason of the active predictions for the Y values. Let's formalize this. So linear regression, what we're going to do is, I'm going to want to solve a minimization problem. So I'll write minimize over theta0 theta1. And I want this to be small, right? I want the difference between h(x) and y to be small. And one thing I might do is try to minimize the square difference between the output of the hypothesis and the actual price of a house. Okay. So lets find some details. You remember that I was using the notation (x(i),y(i)) to represent the ith training example. So what I want really is to sum over my training set, something i = 1 to m, of the square difference between, this is the prediction of my hypothesis when it is input to size of house number i. Right? Minus the actual price that house number I was sold for, and I want to minimize the sum of my training set, sum from I equals one through M, of the difference of this squared error, the square difference between the predicted price of a house, and the price that it was actually sold for. And just remind you of notation, m here was the size of my training set right? So my m there is my number of training examples. Right that hash sign is the abbreviation for number of training examples, okay? And to make some of our, make the math a little bit easier, I'm going to actually look at we are 1 over m times that so let's try to minimize my average minimize one over 2m. Putting the 2 at the constant one half in front, it may just sound the math probably

easier so minimizing one-half of something, right, should give you the same values of the process, theta 0 theta 1, as minimizing that function. And just to be sure, this equation is clear, right? This expression in here, h subscript theta(x), this is our usual, right? That is equal to this plus theta one xi. And this notation, minimize over theta 0 theta 1, this means you'll find me the values of theta 0 and theta 1 that causes this expression to be minimized and this expression depends on theta 0 and theta 1, okay? So just a recap. We're closing this problem as, find me the values of theta zero and theta one so that the average, the 1 over the 2m, times the sum of square errors between my predictions on the training set minus the actual values of the houses on the training set is minimized. So this is going to be my overall objective function for linear regression. And just to rewrite this out a little bit more cleanly, what I'm going to do is, by convention we usually define a cost function, which is going to be exactly this, that formula I have up here. And what I want to do is minimize over theta0 and theta1. My function j(theta0, theta1). Just write this out. This is my cost function. So, this cost function is also called the squared error function. When sometimes called the squared error cost function and it turns out that why do we take the squares of the errors. It turns out that these squared error cost function is a reasonable choice and works well for problems for most regression programs. There are other cost functions that will work pretty well. But the square cost function is probably the most commonly used one for regression problems. Later in this class we'll talk about alternative cost functions as well, but this choice that we just had should be a pretty reasonable thing to try for most linear regression problems. Okay. So that's the cost function. So far we've just seen a mathematical definition of this cost function. In case this function j of theta zero, theta one. In case this function seems a little bit abstract, and you still don't have a good sense of what it's doing, in the next video, in the next couple videos, I'm actually going to go a little bit deeper into what the cause function "J" is doing and try to give you better intuition about what is computing and why we want to use it.



Cost Function (Transcript)

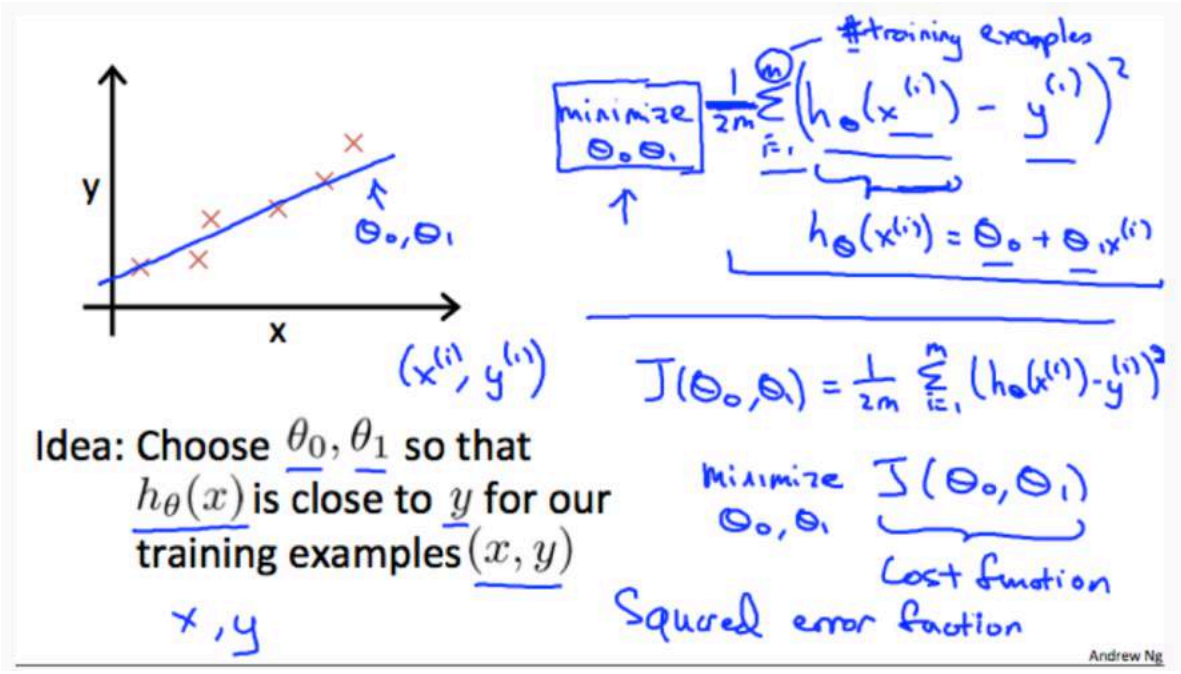
We can measure the accuracy of our hypothesis function by using a cost function. This takes an average difference (actually a fancier version of an average) of all the results of the hypothesis with inputs from x's and the actual output y's.

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(\hat{y}_i - y_i \right)^2 = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x_i) - y_i \right)^2$$

J(θ0,θ1)=12m∑i=1m(y^i−yi)2=12m∑i=1m(hθ(xi)−yi)2
To break it apart, it is 12 x¯ where x¯ is the mean of the squares of hθ(xi)−yi , or the difference between the predicted value and the actual value.

This function is otherwise called the "Squared error function", or "Mean squared error". The mean is halved (12) as a convenience for the computation of the gradient descent, as the derivative term of the square function will

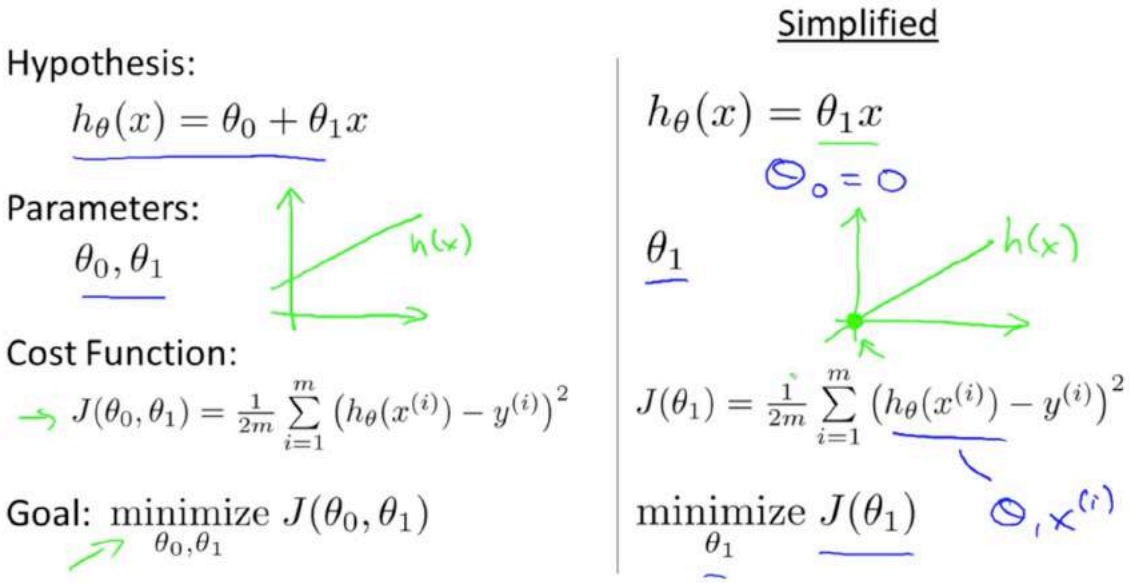
cancel out the 12 term. The following image summarizes what the cost function does:



Cost Function - Intuition 1 (Video)

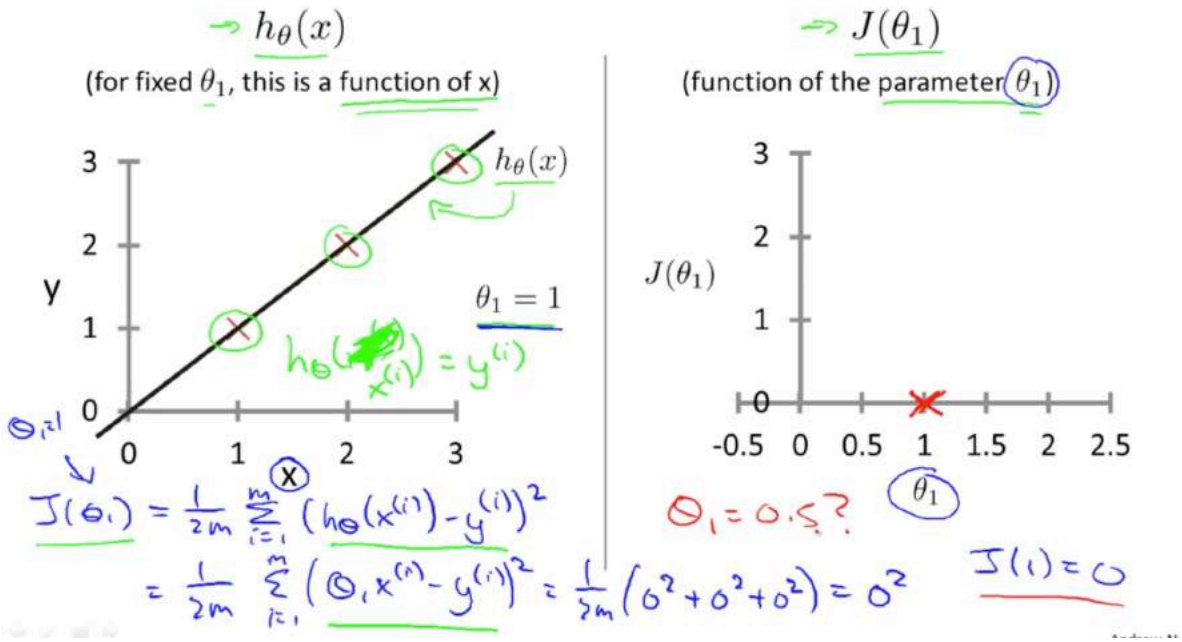
In the previous video, we gave the mathematical definition of the cost function. In this video, let's look at some examples, to get back to intuition about what the cost function is doing, and why we want to use it. To recap, here's what we had last time. We want to fit a straight line to our data, so we had this formed as a hypothesis with these parameters theta zero and theta one, and with different choices of the parameters we end up with different straight line fits. So the data which are fit like so, and there's a cost function, and that was our optimization objective. [sound] So this video, in order to better visualize the cost function J, I'm going to work with a simplified hypothesis function, like that shown on the right. So I'm gonna use my simplified hypothesis, which is just theta one times X. We can, if you want, think of this as setting the parameter theta zero equal to 0. So I have only one parameter theta one and my cost function is similar to before except that now H of X that is now equal to just theta one times X. And I have only one parameter theta one and so my

optimization objective is to minimize j of theta one. In pictures what this means is that if theta zero equals zero that corresponds to choosing only hypothesis functions that pass through the origin, that pass through the point (0, 0).



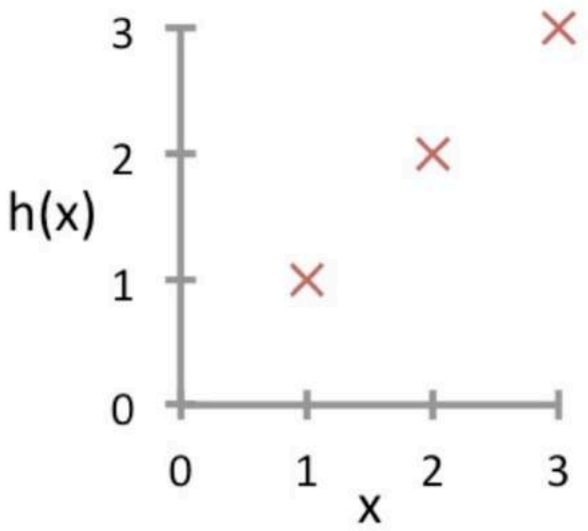
Using this simplified definition of a hypothesizing cost function let's try to understand the cost function concept better. It turns out that two key functions we want to understand. The first is the hypothesis function, and the second is a cost function. So, notice that the hypothesis, right, H of X. For a face value of theta one, this is a function of X. So the hypothesis is a function of, what is the size of the house X. In contrast, the cost function, J, that's a function of the parameter, theta one, which controls the slope of the straight line. Let's plot these functions and try to understand them both better. Let's start with the hypothesis. On the left, let's say here's my training set with three points at (1, 1), (2, 2), and (3, 3). Let's pick a value theta one, so when theta one equals one, and if that's my choice for theta one, then my hypothesis is going to look like this straight line over here. And I'm gonna point out, when I'm plotting my hypothesis function. X-axis, my horizontal axis is labeled X, is labeled you know, size of the house over here. Now, of temporary, set theta one equals one, what I want to do is figure out what is j of theta one, when theta one equals one. So let's go ahead and compute what the cost function has for. You'll devalue one. Well, as usual, my cost function is defined as follows, right? Some from, some of 'em are training sets of this usual squared error term. And, this is therefore equal to. And this. Of theta one x I minus y I and if you simplify this turns out to be. That. Zero Squared to zero squared to zero squared which is of course, just equal to zero. Now, inside the cost function. It turns out each of these terms here is equal to zero. Because for the specific training set I have or my 3 training examples are (1, 1), (2, 2), (3,3). If theta one is equal to one. Then h of x. H of x i. Is equal to y I exactly, let me write this better. Right? And so, h

of x minus y, each of these terms is equal to zero, which is why I find that j of one is equal to zero. So, we now know that j of one Is equal to zero. Let's plot that. What I'm gonna do on the right is plot my cost function j. And notice, because my cost function is a function of my parameter theta one, when I plot my cost function, the horizontal axis is now labeled with theta one. So I have j of one zero zero so let's go ahead and plot that. End up with. An X over there. Now lets look at some other examples. Theta-1 can take on a range of different values. Right? So theta-1 can take on the negative values, zero, positive values.



Question

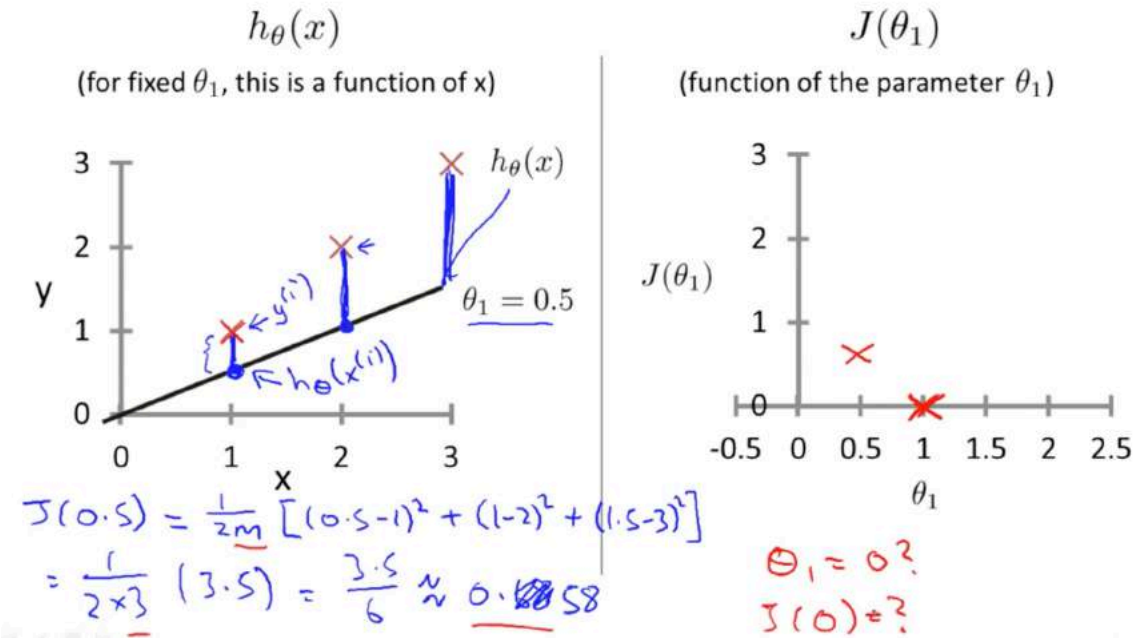
Suppose we have a training set with m=3 examples, plotted below. Our hypothesis representation is $h_\theta(x) = \theta_1 x$, with parameter θ_1. The cost function $J(\theta_1)$ is $J(\theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$. What is $J(0)$?



- 0
- 1/6
- 1
- 14/6

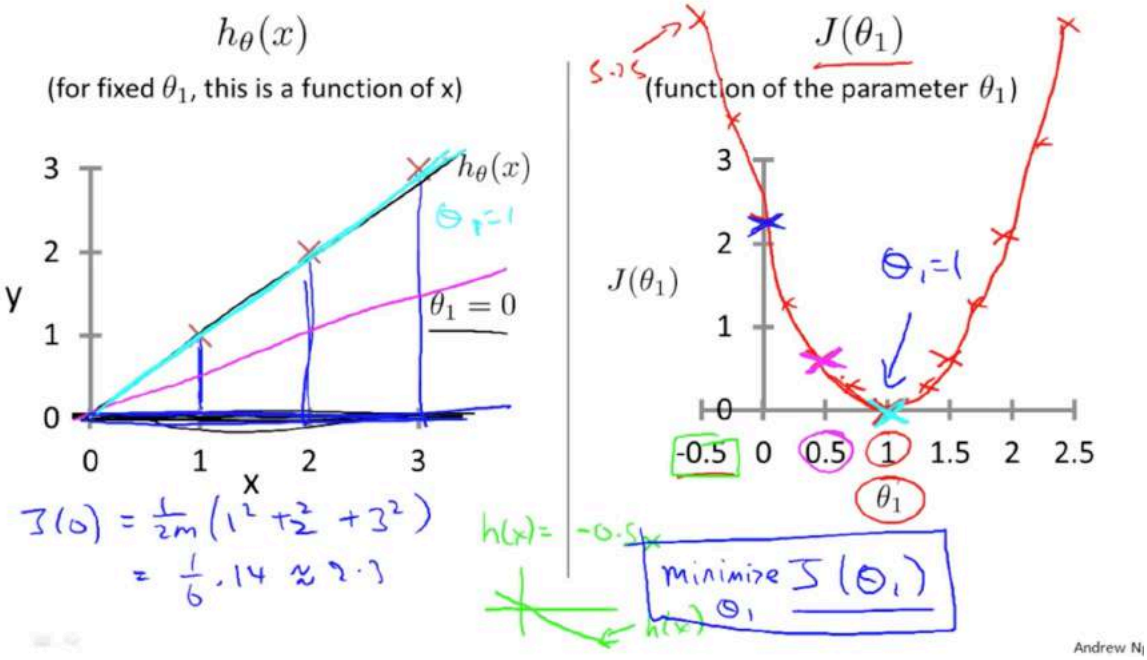
So what if theta-1 is equal to 0.5. What happens then? Let's go ahead and plot that. I'm now going to set theta-1 equals 0.5, and in that case my hypothesis now looks like this. As a line with slope equals to 0.5, and, lets compute J, of 0.5. So that is going to be one over 2M of, my usual cost function. It turns out that the cost function is going to be the sum of square values of the height of this line. Plus the sum of square of the height of that line, plus the sum of square of the height of that line, right? ?Cause just this vertical distance, that's the difference between, you know, Y. I. and the predicted value, H of XI, right? So the first example is going to be 0.5 minus one squared. Because my hypothesis predicted 0.5. Whereas, the actual value was one. For my second example, I get, one minus two squared, because my hypothesis predicted one,

but the actual housing price was two. And then finally, plus. 1.5 minus three squared. And so that's equal to one over two times three. Because, M when trading set size, right, have three training examples. In that, that's times simplifying for the parentheses it's 3.5. So that's 3.5 over six which is about 0.68. So now we know that j of 0.5 is about 0.68.[Should be 0.58] Lets go and plot that. Oh excuse me, math error, it's actually 0.58. So we plot that which is maybe about over there.



Okay? Now, let's do one more. How about if theta one is equal to zero, what is J of zero equal to? It turns out that if theta one is equal to zero, then H of X is just equal to, you know, this flat line, right, that just goes horizontally like this. And so, measuring the errors. We have that J of zero is equal to one over two M, times one squared plus two squared plus three squared, which is, One six times fourteen which is about 2.3. So let's go ahead and plot as well. So it ends up with a value around 2.3 and of course we can keep on doing this for other values of theta one. It turns out that you can have you know negative values of theta one as well so if theta one is negative then h of x would be equal to say minus 0.5 times x then theta one is minus 0.5 and so that corresponds to a hypothesis with a slope of negative 0.5. And you can actually keep on computing these errors. This turns out to be, you know, for 0.5, it turns out to have really high error. It works out to be something, like, 5.25. And so on, and the different values of theta one, you can compute these things, right? And it turns out that you, your computed range of values, you get something like that. And by computing the range of values, you can actually slowly create out. What does function J of Theta say and that's what J of Theta is. To recap, for each value of theta one, right? Each value of theta one corresponds to a different hypothesis, or to a different straight line fit on the left. And for each value of theta one, we could then derive a different value of j of theta one. And

for example, you know, theta one=1, corresponded to this straight line straight through the data. Whereas theta one=0.5. And this point shown in magenta corresponded to maybe that line, and theta one=zero which is shown in blue that corresponds to this horizontal line. Right, so for each value of theta one we wound up with a different value of J of theta one and we could then use this to trace out this plot on the right. Now you remember, the optimization objective for our learning algorithm is we want to choose the value of theta one. That minimizes J of theta one. Right? This was our objective function for the linear regression. Well, looking at this curve, the value that minimizes j of theta one is, you know, theta one equals to one. And low and behold, that is indeed the best possible straight line fit through our data, by setting theta one equals one. And just, for this particular training set, we actually end up fitting it perfectly. And that's why minimizing j of theta one corresponds to finding a straight line that fits the data well. So, to wrap up. In this video, we looked up some plots. To understand the cost function. To do so, we simplify the algorithm. So that it only had one parameter theta one. And we set the parameter theta zero to be only zero. In the next video. We'll go back to the original problem formulation and look at some visualizations involving both theta zero and theta one. That is without setting theta zero to zero. And hopefully that will give you, an even better sense of what the cost function j is doing in the original linear regression formulation.

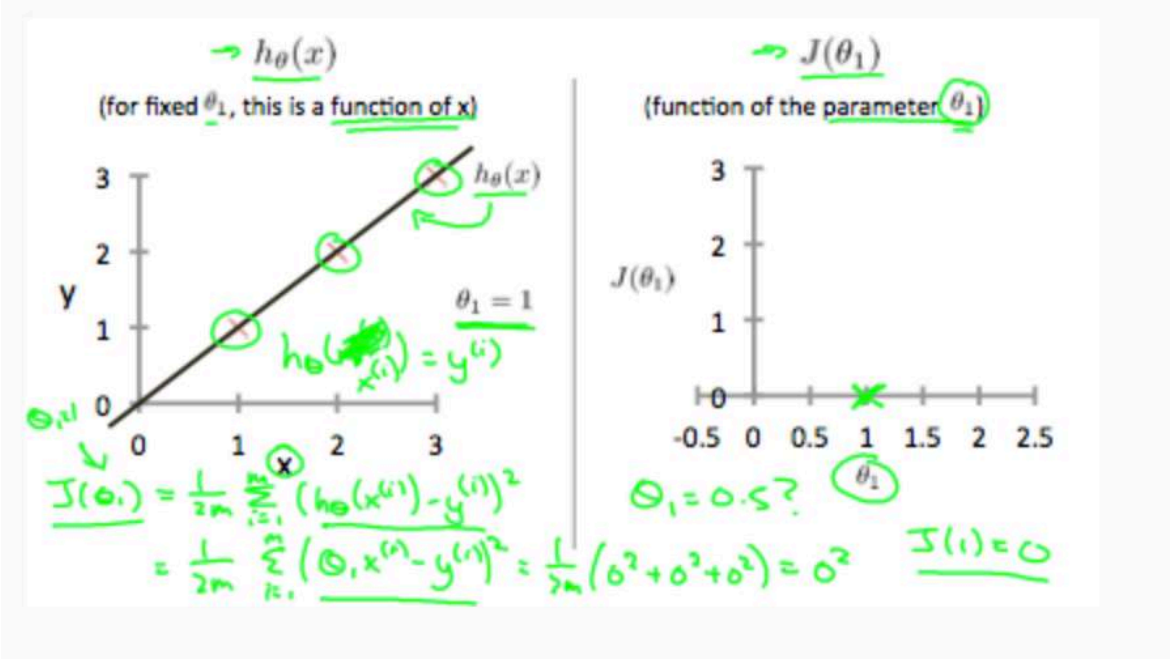


Cost Function - Intuition 1

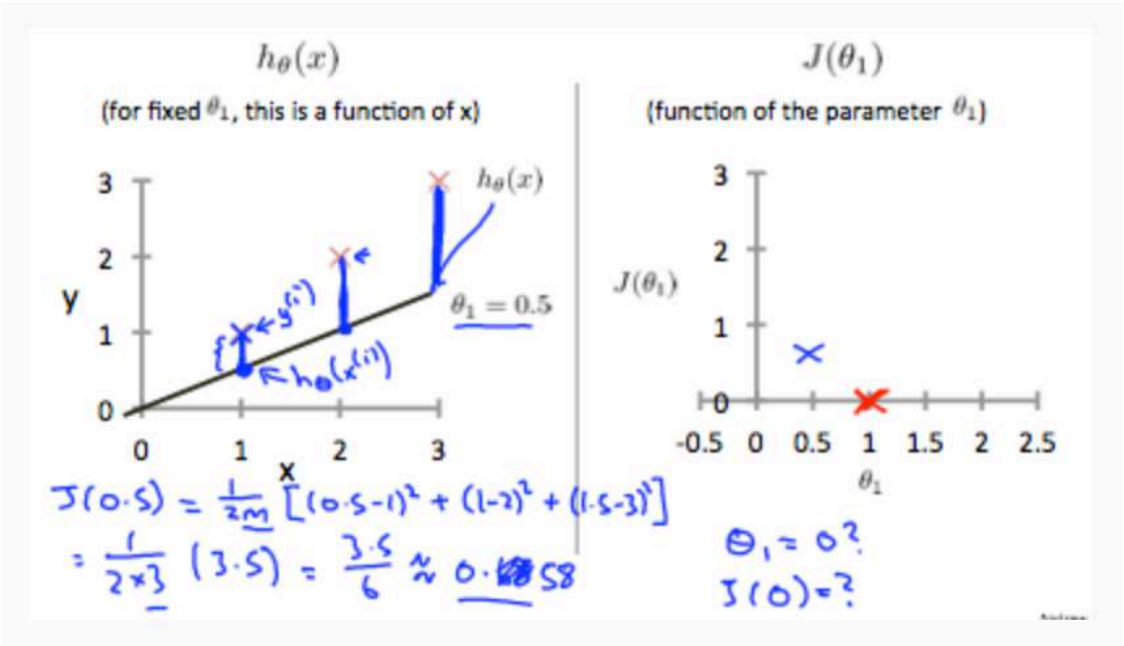
(Transcript)

If we try to think of it in visual terms, our training data set is scattered on the x-y plane. We are trying to make a straight line (defined by hθ(x)) which passes through these scattered data points.

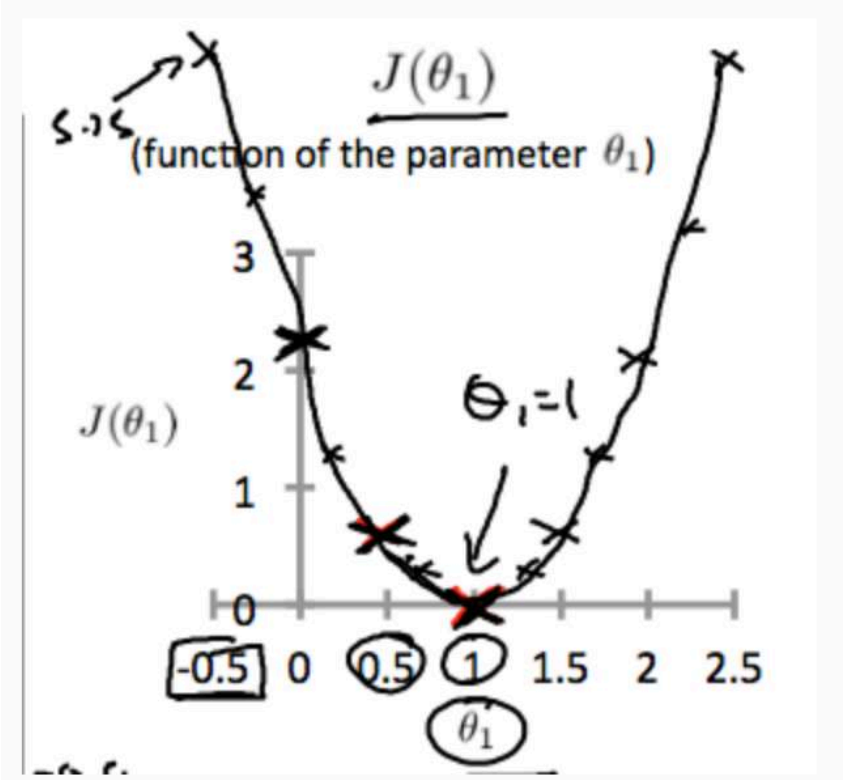
Our objective is to get the best possible line. The best possible line will be such so that the average squared vertical distances of the scattered points from the line will be the least. Ideally, the line should pass through all the points of our training data set. In such a case, the value of J(θ0,θ1) will be 0. The following example shows the ideal situation where we have a cost function of 0.



When θ1=1, we get a slope of 1 which goes through every single data point in our model. Conversely, when θ1=0.5, we see the vertical distance from our fit to the data points increase.



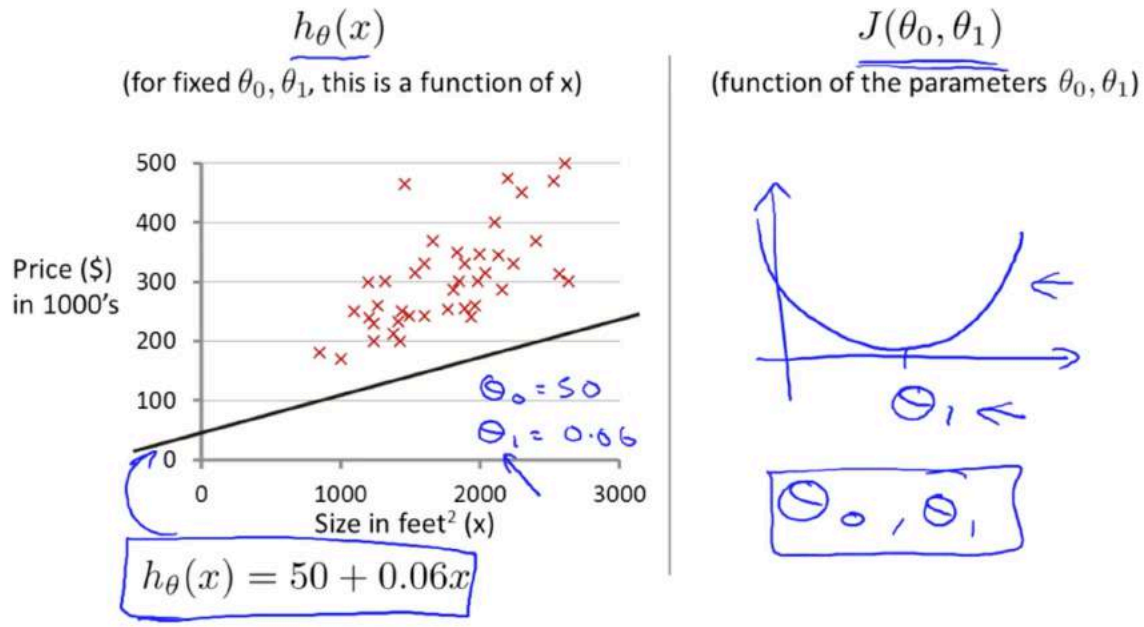
This increases our cost function to 0.58. Plotting several other points yields to the following graph:



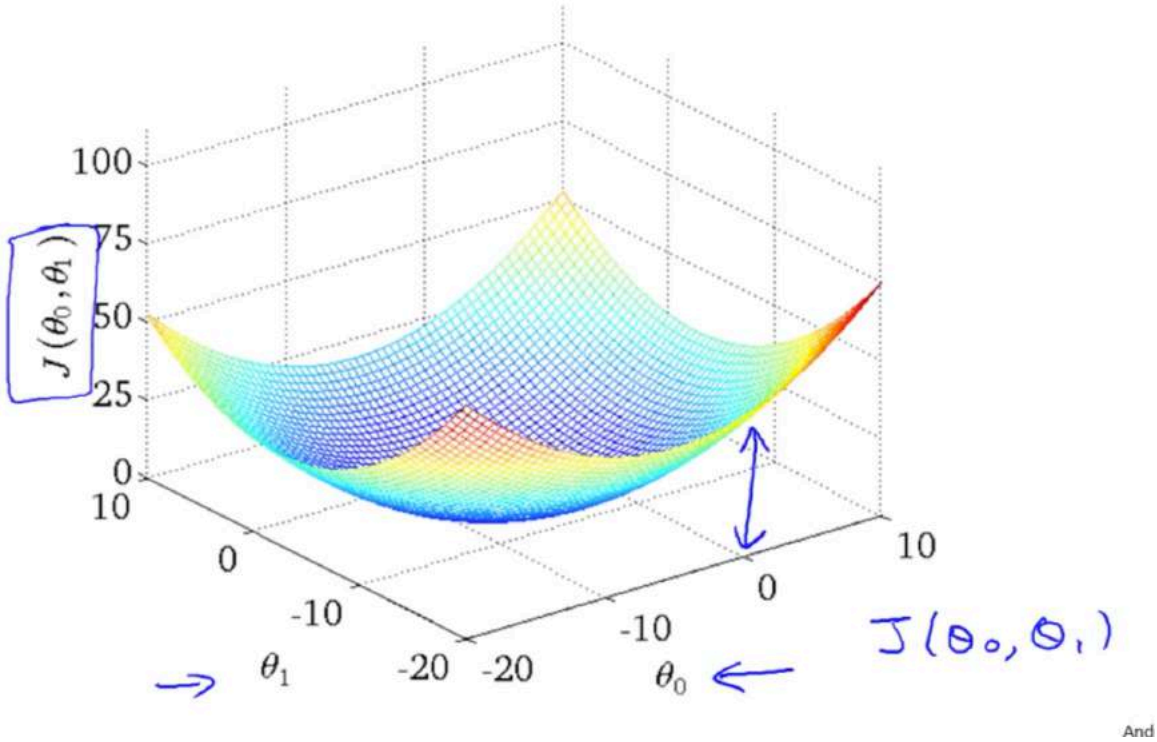
Thus as a goal, we should try to minimize the cost function. In this case, θ1=1 is our global minimum.

Cost Function - Intuition 2 (Video)

In this video, lets delve deeper and get even better intuition about what the cost function is doing. This video assumes that you're familiar with contour plots. If you are not familiar with contour plots or contour figures some of the illustrations in this video may or may not make sense to you but is okay and if you end up skipping this video or some of it does not quite make sense because you haven't seen contour plots before. That's okay and you will still understand the rest of this course without those parts of this. Here's our problem formulation as usual, with the hypothesis parameters, cost function, and our optimization objective. Unlike before, unlike the last video, I'm going to keep both of my parameters, theta zero, and theta one, as we generate our visualizations for the cost function. So, same as last time, we want to understand the hypothesis H and the cost function J. So, here's my training set of housing prices and let's make some hypothesis. You know, like that one, this is not a particularly good hypothesis. But, if I set theta zero=50 and theta one=0.06, then I end up with this hypothesis down here and that corresponds to that straight line. Now given these value of theta zero and theta one, we want to plot the corresponding, you know, cost function on the right. What we did last time was, right, when we only had theta one. In other words, drawing plots that look like this as a function of theta one. But now we have two parameters, theta zero, and theta one, and so the plot gets a little more complicated.



It turns out that when we have only one parameter, that the parts we drew had this sort of bow shaped function. Now, when we have two parameters, it turns out the cost function also has a similar sort of bow shape. And, in fact, depending on your training set, you might get a cost function that maybe looks something like this. So, this is a 3-D surface plot, where the axes are labeled theta zero and theta one. So as you vary theta zero and theta one, the two parameters, you get different values of the cost function J (theta zero, theta one) and the height of this surface above a particular point of theta zero, theta one. Right, that's, that's the vertical axis. The height of the surface of the points indicates the value of J of theta zero, J of theta one. And you can see it sort of has this bow like shape.

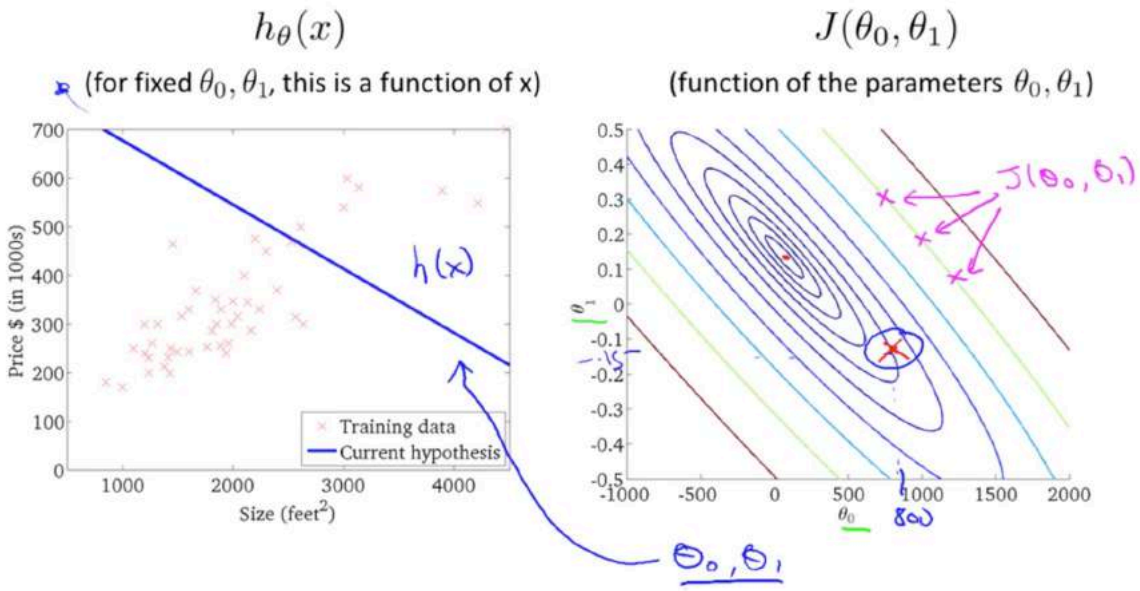


And

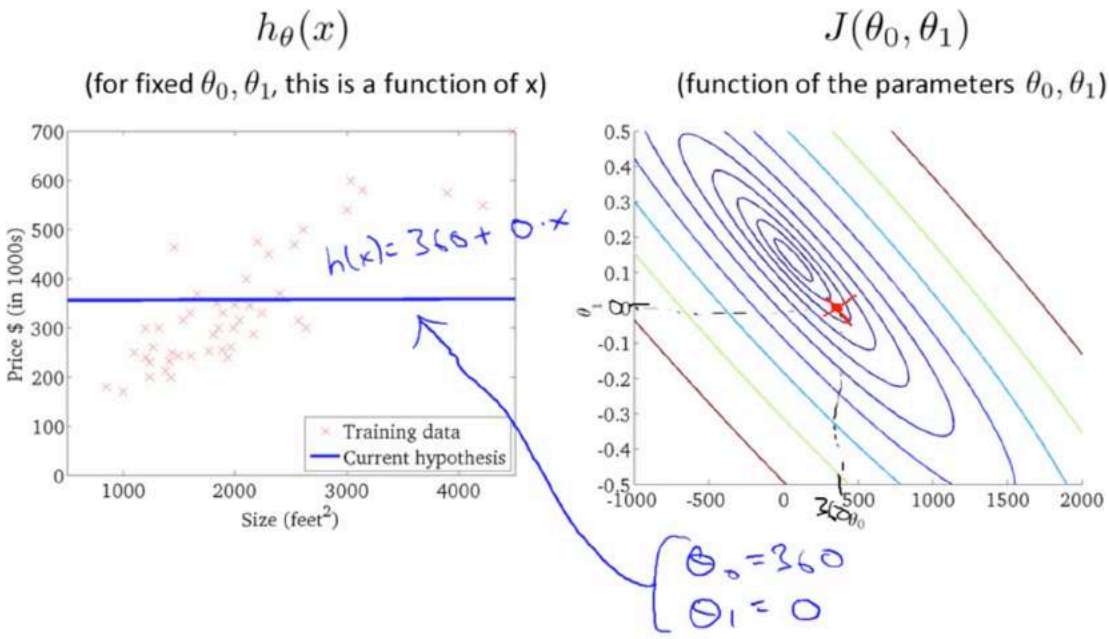
Let me show you the same plot in 3D. So here's the same figure in 3D, horizontal axis theta one and vertical axis J(theta zero, theta one), and if I rotate this plot around. You kinda of a get a sense, I hope, of this bowl shaped surface as that's what the cost function J looks like.

Now for the purpose of illustration in the rest of this video I'm not actually going to use these sort of 3D surfaces to show you the cost function J, instead I'm going to use contour plots. Or what I also call contour figures. I guess they mean the same thing. To show you these surfaces. So here's an example of a contour figure, shown on the right, where the axis are theta zero and theta one. And what each of these ovals, what each of these ellipsis shows is a set of points that takes on the same value for J(theta zero, theta one). So concretely, for example this, you'll take that point and that point and that point. All three of these points that I just drew in magenta, they have the same value for J (theta zero, theta one). Okay. Where, right, these, this is the theta zero, theta one axis but those three have the same Value for J (theta zero, theta one) and if you haven't seen contour plots much before think of, imagine if you will. A bow shaped function that's coming out of my screen. So that the minimum, so the bottom of the bow is this point right there, right? This middle, the middle of these concentric ellipses. And imagine a bow shape that sort of grows out of my screen like this, so that each of these ellipses, you know, has

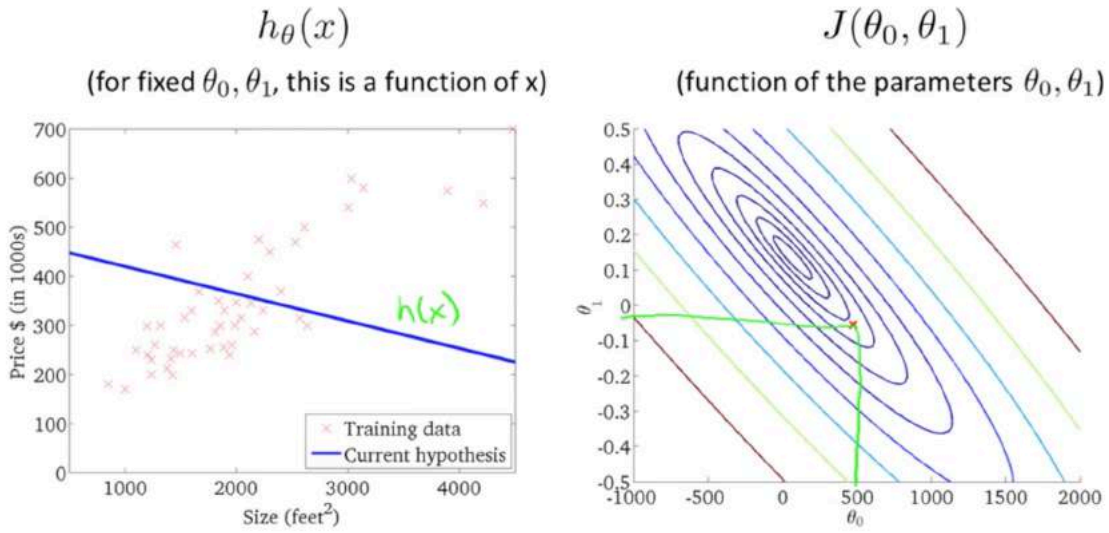
the same height above my screen. And the minimum with the bow, right, is right down there. And so the contour figures is a, is way to, is maybe a more convenient way to visualize my function J. [sound] So, let's look at some examples. Over here, I have a particular point, right? And so this is, with, you know, theta zero equals maybe about 800, and theta one equals maybe a -0.15 . And so this point, right, this point in red corresponds to one set of pair values of theta zero, theta one and the corresponding, in fact, to that hypothesis, right, theta zero is about 800, that is, where it intersects the vertical axis is around 800, and this is slope of about -0.15. Now this line is really not such a good fit to the data, right. This hypothesis, h(x), with these values of theta zero, theta one, it's really not such a good fit to the data. And so you find that, it's cost. Is a value that's out here that's you know pretty far from the minimum right it's pretty far this is a pretty high cost because this is just not that good a fit to the data.



Let's look at some more examples. Now here's a different hypothesis that's you know still not a great fit for the data but may be slightly better so here right that's my point that those are my parameters theta zero theta one and so my theta zero value. Right? That's bout 360 and my value for theta one. Is equal to zero. So, you know, let's break it out. Let's take theta zero equals 360 theta one equals zero. And this pair of parameters corresponds to that hypothesis, corresponds to flat line, that is, h(x) equals 360 plus zero times x. So that's the hypothesis. And this hypothesis again has some cost, and that cost is, you know, plotted as the height of the J function at that point.

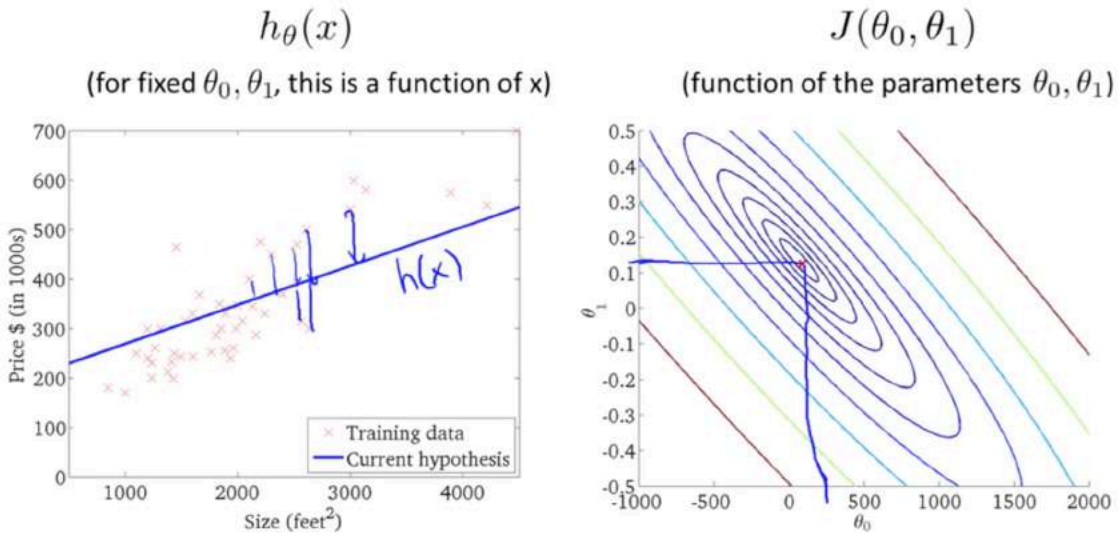


Let's look at just a couple of examples. Here's one more, you know, at this value of theta zero, and at that value of theta one, we end up with this hypothesis, h(x) and again, not a great fit to the data, and is actually further away from the minimum.



Last example, this is actually not quite at the minimum, but it's pretty close to the minimum. So this is not such a bad fit to the, to the data, where, for a particular value, of, theta zero. Which, one of them has value, as in for a particular value for theta one. We get a particular h(x). And this is, this is not

quite at the minimum, but it's pretty close. And so the sum of squares errors is sum of squares distances between my, training samples and my hypothesis. Really, that's a sum of square distances, right? Of all of these errors. This is pretty close to the minimum even though it's not quite the minimum.

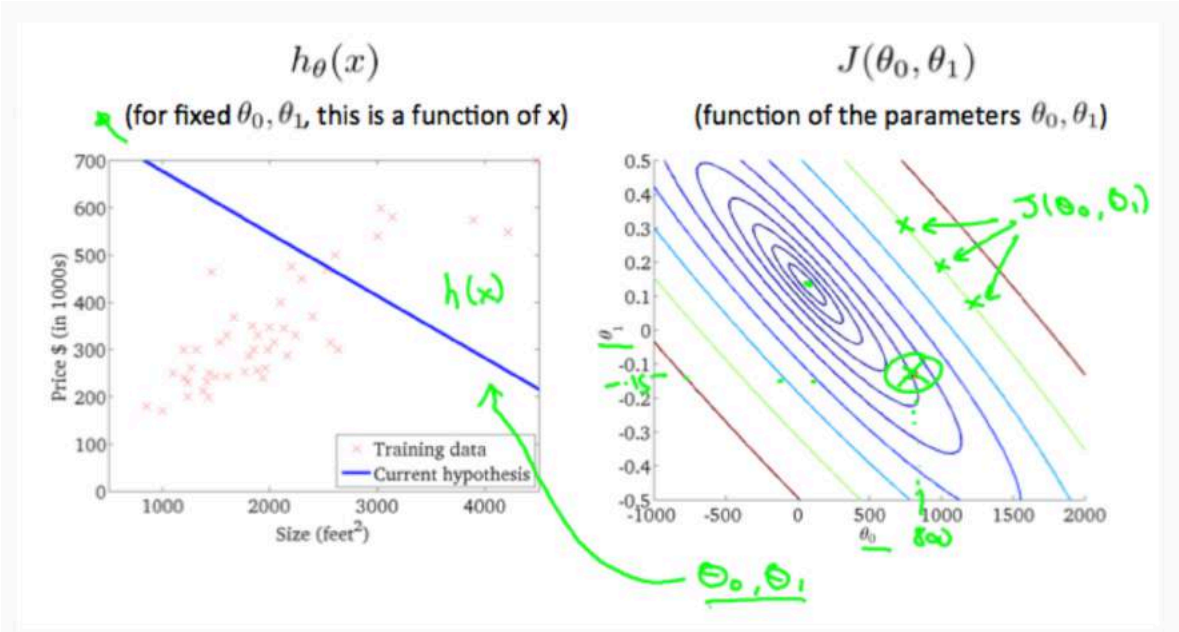


So with these figures I hope that gives you a better understanding of what values of the cost function J, how they are and how that corresponds to different hypothesis and so as how better hypotheses may corresponds to points that are closer to the minimum of this cost function J. Now of course what we really want is an efficient algorithm, right, a efficient piece of software for automatically finding The value of theta zero and theta one, that minimizes the cost function J, right? And what we, what we don't wanna do is to, you know, how to write software, to plot out this point, and then try to manually read off the numbers, that this is not a good way to do it. And, in fact, we'll see it later, that when we look at more complicated examples, we'll have high dimensional figures with more parameters, that, it turns out, we'll see in a few, we'll see later in this course, examples where this figure, you know, cannot really be plotted, and this becomes much harder to visualize. And so, what we want is to have software to find the value of theta zero, theta one that minimizes this function and in the next video we start to talk about an algorithm for automatically finding that value of theta zero and theta one that minimizes the cost function J.

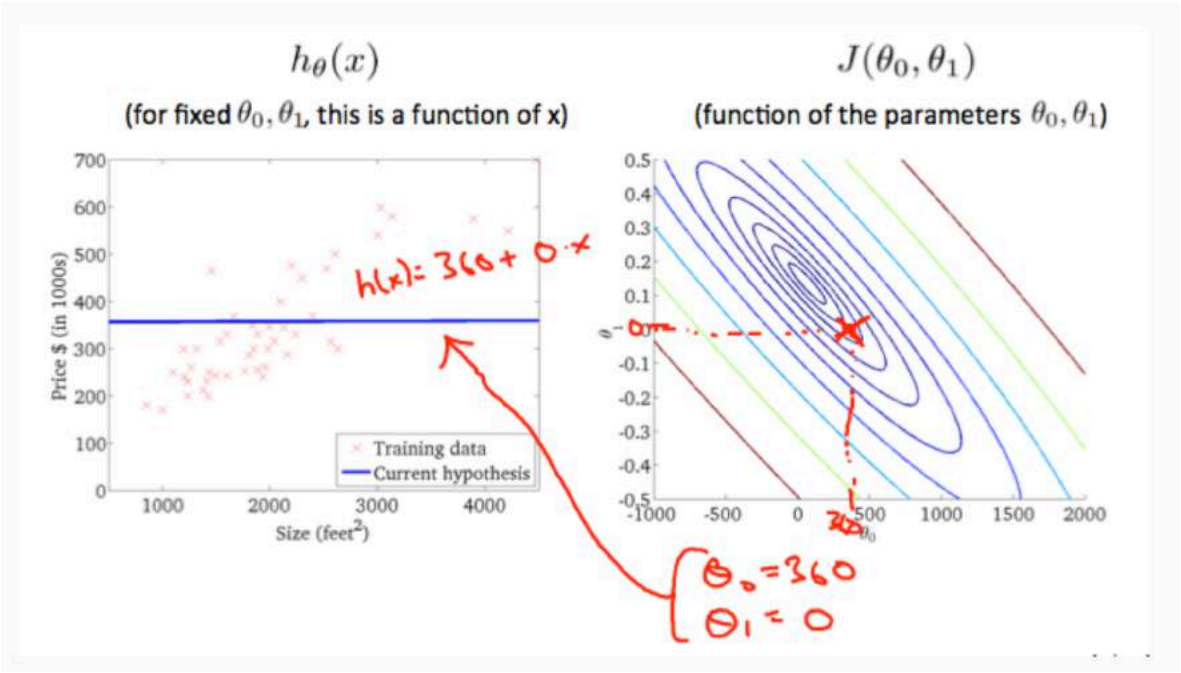
Cost Function - Intuition 2

(Transcript)

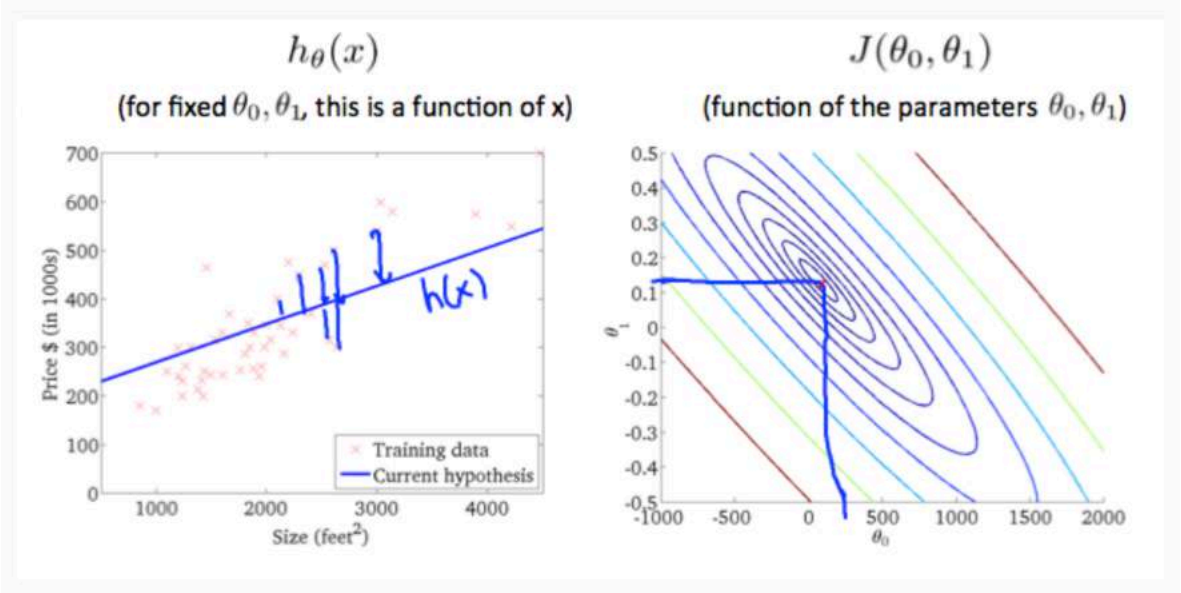
A contour plot is a graph that contains many contour lines. A contour line of a two variable function has a constant value at all points of the same line. An example of such a graph is the one to the right below.



Taking any color and going along the 'circle', one would expect to get the same value of the cost function. For example, the three green points found on the green line above have the same value for J(θ0,θ1) and as a result, they are found along the same line. The circled x displays the value of the cost function for the graph on the left when θ0 = 800 and θ1= -0.15. Taking another h(x) and plotting its contour plot, one gets the following graphs:



When θ0 = 360 and θ1 = 0, the value of J(θ0,θ1) in the contour plot gets closer to the centre thus reducing the cost function error. Now giving our hypothesis function a slightly positive slope results in a better fit of the data.



The graph above minimizes the cost function as much as possible and consequently, the result of θ1 and θ0 tend to be around 0.12 and 250 respectively. Plotting those values on our graph to the right seems to put our point in the centre of the inner most 'circle'.

Parameter Learning

Gradient Descent (Video)

We previously defined the cost function J. In this video, I want to tell you about an algorithm called gradient descent for minimizing the cost function J. It turns out gradient descent is a more general algorithm, and is used not only in linear regression. It's actually used all over the place in machine learning. And later in the class, we'll use gradient descent to minimize other functions as well, not just the cost function J for the linear regression. So in this video, we'll talk about gradient descent for minimizing some arbitrary function J and then in later videos, we'll take this algorithm and apply it specifically to the cost function J that we have defined for linear regression. So here's the problem setup. Going to assume that we have some function J(theta 0, theta 1) maybe it's the cost function from linear regression, maybe it's some other function we wanna minimize. And we want to come up with an algorithm for minimizing that as a function of J(theta 0, theta 1). Just as an aside it turns out that gradient descent actually applies to more general functions. So imagine, if you have a function that's a function of J, as theta 0, theta 1, theta 2, up to say some theta n, and you want to minimize theta 0. You minimize over theta 0 up to theta n of this J of theta 0 up to theta n. And it turns our gradient descent is an algorithm for solving this more general problem. But for the sake of brevity, for the sake of succinctness of notation, I'm just going to pretend I have only two parameters throughout the rest of this video. Here's the idea for gradient descent. What we're going to do is we're going to start off with some initial guesses for theta 0 and theta 1. Doesn't really matter what they are, but a common choice would be we set theta 0 to 0, and set theta 1 to 0, just initialize them to 0. What we're going to do in gradient descent is we'll keep changing theta 0 and theta 1 a little bit to try to reduce J(theta 0, theta 1), until hopefully, we wind at a minimum, or maybe at a local minimum.

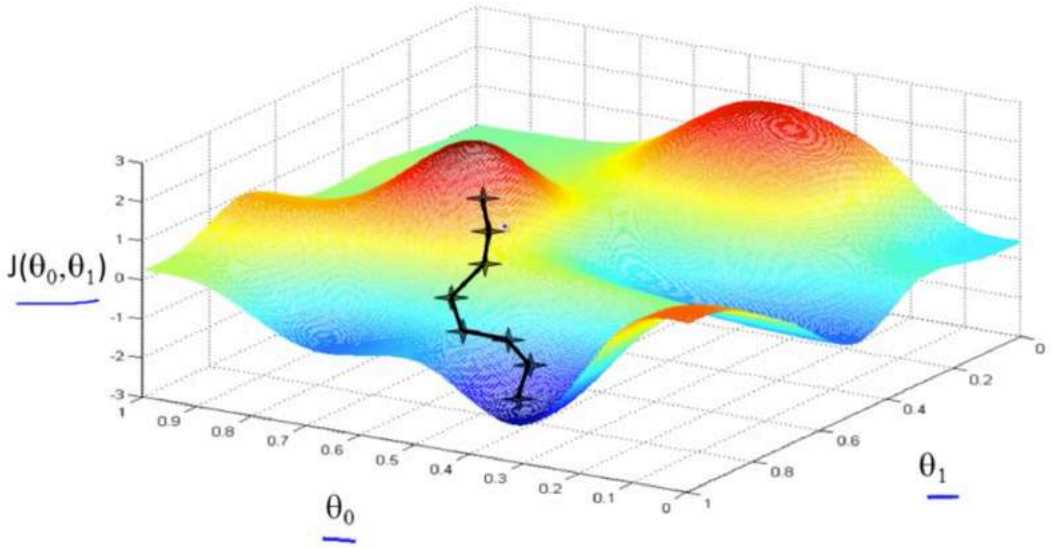
Have some function $J(\theta_0, \theta_1)$ $J(\theta_0, \theta_1, \theta_2, \ldots, \theta_n)$

Want $\min_{\theta_0, \theta_1} J(\theta_0, \theta_1)$ $\min_{\theta_0 \cdots \theta_n} J(\theta_0, \ldots, \theta_n)$

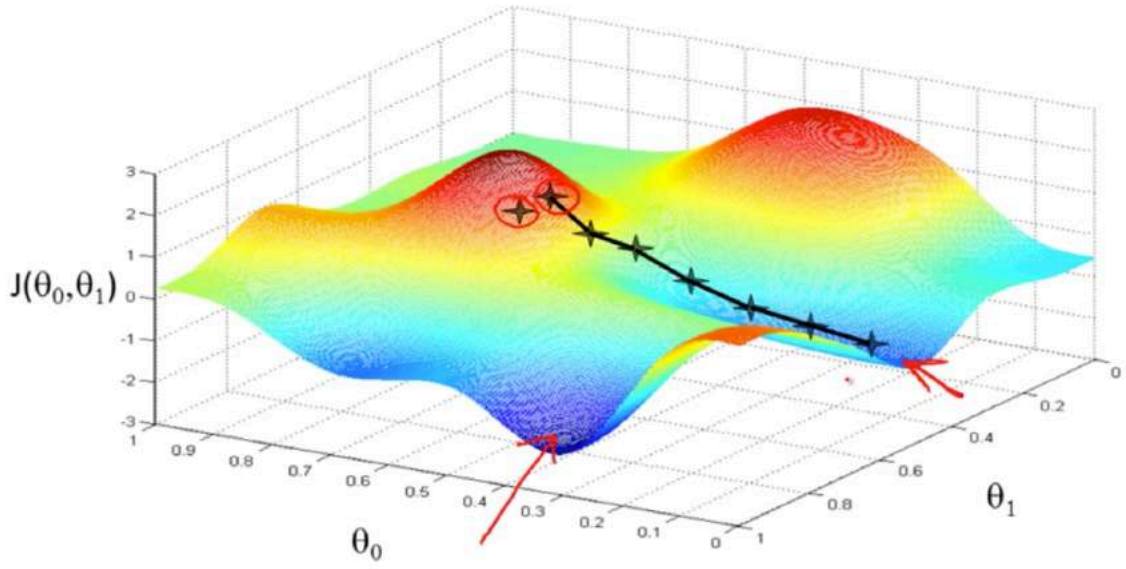
Outline:

- Start with some θ_0, θ_1 (Say $\theta_0 = 0, \theta_1 = 0$)
- Keep changing θ_0, θ_1 to reduce $J(\theta_0, \theta_1)$ until we hopefully end up at a minimum

So let's see in pictures what gradient descent does. Let's say you're trying to minimize this function. So notice the axes, this is theta 0, theta 1 on the horizontal axes and J is the vertical axis and so the height of the surface shows J and we want to minimize this function. So we're going to start off with theta 0, theta 1 at some point. So imagine picking some value for theta 0, theta 1, and that corresponds to starting at some point on the surface of this function. So whatever value of theta 0, theta 1 gives you some point here. I did initialize them to 0, 0 but sometimes you initialize it to other values as well. Now, I want you to imagine that this figure shows a hole. Imagine this is like the landscape of some grassy park, with two hills like so, and I want us to imagine that you are physically standing at that point on the hill, on this little red hill in your park. In gradient descent, what we're going to do is we're going to spin 360 degrees around, just look all around us, and ask, if I were to take a little baby step in some direction, and I want to go downhill as quickly as possible, what direction do I take that little baby step in? If I wanna go down, so I wanna physically walk down this hill as rapidly as possible. Turns out, that if you're standing at that point on the hill, you look all around and you find that the best direction is to take a little step downhill is roughly that direction. Okay, and now you're at this new point on your hill. You're gonna, again, look all around and say what direction should I step in order to take a little baby step downhill? And if you do that and take another step, you take a step in that direction. And then you keep going. From this new point you look around, decide what direction would take you downhill most quickly. Take another step, another step, and so on until you converge to this local minimum down here.

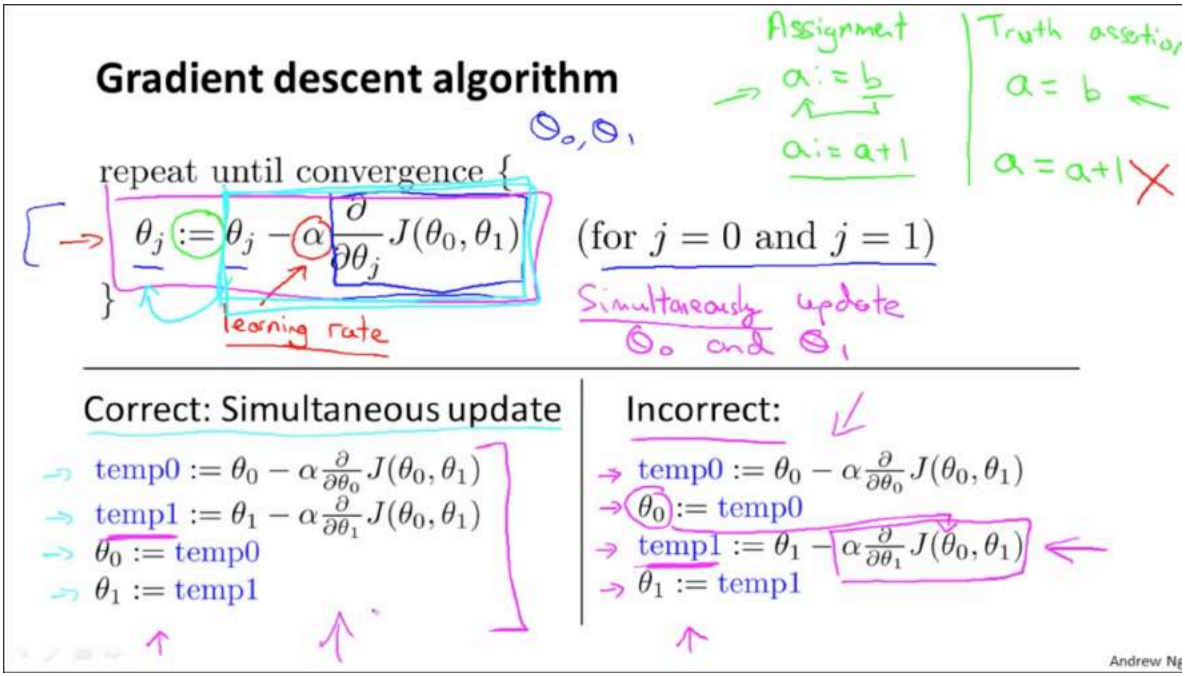


Gradient descent has an interesting property. This first time we ran gradient descent we were starting at this point over here, right? Started at that point over here. Now imagine we had initialized gradient descent just a couple steps to the right. Imagine we'd initialized gradient descent with that point on the upper right. If you were to repeat this process, so start from that point, look all around, take a little step in the direction of steepest descent, you would do that. Then look around, take another step, and so on. And if you started just a couple of steps to the right, gradient descent would've taken you to this second local optimum over on the right. So if you had started this first point, you would've wound up at this local optimum, but if you started just at a slightly different location, you would've wound up at a very different local optimum. And this is a property of gradient descent that we'll say a little bit more about later.



So that's the intuition in pictures. Let's look at the math. This is the definition of the gradient descent algorithm. We're going to just repeatedly do this until convergence, we're going to update my parameter theta j by taking theta j and subtracting from it alpha times this term over here, okay? So let's see, there's lot of details in this equation so let me unpack some of it. First, this notation here, :=, gonna use := to denote assignment, so it's the assignment operator. So briefly, if I write a := b, what this means is, it means in a computer, this means take the value in b and use it overwrite whatever value is a. So this means set a to be equal to the value of b, which is assignment. And I can also do a := a + 1. This means take a and increase its value by one. Whereas in contrast, if I use the equal sign and I write a equals b, then this is a truth assertion. Okay? So if I write a equals b, then I'll asserting that the value of a equals to the value of b, right? So the left hand side, that's the computer operation, where we set the value of a to a new value. The right hand side, this is asserting, I'm just making a claim that the values of a and b are the same, and so whereas you can write a := a + 1, that means increment a by 1, hopefully I won't ever write a = a + 1 because that's just wrong. a and a + 1 can never be equal to the same values. Okay? So this is first part of the definition. This alpha here is a number that is called the learning rate. And what alpha does is it basically controls how big a step we take downhill with creating descent. So if alpha is very large, then that corresponds to a very aggressive gradient descent procedure where we're trying take huge steps downhill and if alpha is very small, then we're taking little, little baby steps downhill. And I'll come back and say more about this later, about how to set alpha and so on. And finally, this term here, that's a derivative term. I don't wanna talk about it right now, but I will derive this derivative term and tell you exactly what this is

later, okay? And some of you will be more familiar with calculus than others, but even if you aren't familiar with calculus, don't worry about it. I'll tell you what you need to know about this term here. Now, there's one more subtlety about gradient descent which is in gradient descent we're going to update, you know, theta 0 and theta 1, right? So this update takes place for j = 0 and j = 1, so you're gonna update theta 0 and update theta 1. And the subtlety of how you implement gradient descent is for this expression, for this update equation, you want to simultaneously update theta 0 and theta 1. What I mean by that is that in this equation, we're gonna update theta 0 := theta 0 minus something, and update theta 1 := theta 1 minus something. And the way to implement is you should compute the right hand side, right? Compute that thing for theta 0 and theta 1 and then simultaneously, at the same time, update theta 0 and theta 1, okay? So let me say what I mean by that. This is a correct implementation of gradient descent meaning simultaneous update. So I'm gonna set temp0 equals that, set temp1 equals that so basic compute the right-hand sides, and then having computed the right-hand sides and stored them into variables temp0 and temp1, I'm gonna update theta 0 and theta 1 simultaneously because that's the correct implementation. In contrast, here's an incorrect implementation that does not do a simultaneous update. So in this incorrect implementation, we compute temp0, and then we update theta 0, and then we compute temp1, and then we update temp1. And the difference between the right hand side and the left hand side implementations is that If you look down here, you look at this step, if by this time you've already updated theta 0, then you would be using the new value of theta 0 to compute this derivative term. And so this gives you a different value of temp1, than the left-hand side, right? Because you've now plugged in the new value of theta 0 into this equation. And so, this on the right-hand side is not a correct implementation of gradient descent, okay? So I don't wanna say why you need to do the simultaneous updates. It turns out that the way gradient descent is usually implemented, which I'll say more about later, it actually turns out to be more natural to implement the simultaneous updates. And when people talk about gradient descent, they always mean simultaneous update. If you implement the non simultaneous update, it turns out it will probably work anyway. But this algorithm wasn't right. It's not what people refer to as gradient descent, and this is some other algorithm with different properties. And for various reasons this can behave in slightly stranger ways, and so what you should do is really implement the simultaneous update of gradient descent.



Question

Suppose $\theta_0 = 1, \theta_1 = 2$, and we simultaneously update θ_0 and θ_1 using the rule: $\theta_j := \theta_j + \sqrt{\theta_0 \theta_1}$ (for j = 0 and j=1) What are the resulting values of θ_0 and θ_1?

- ○ $\theta_0 = 1, \theta_1 = 2$
- ◉ $\theta_0 = 1 + \sqrt{2}, \theta_1 = 2 + \sqrt{2}$
- ○ $\theta_0 = 2 + \sqrt{2}, \theta_1 = 1 + \sqrt{2}$
- ○ $\theta_0 = 1 + \sqrt{2}, \theta_1 = 2 + \sqrt{(1 + \sqrt{2}) \cdot 2}$

So, that's the outline of the gradient descent algorithm. In the next video, we're going to go into the details of the derivative term, which I wrote up but didn't really define. And if you've taken a calculus class before and if you're familiar with partial derivatives and derivatives, it turns out that's exactly what that derivative term is, but in case you aren't familiar with calculus, don't worry about it. The next video will give you all the intuitions and will tell you everything you need to know to compute that derivative term, even if you haven't seen calculus, or even if you haven't seen partial derivatives before. And with that, with the next video, hopefully we'll be able to give you all the

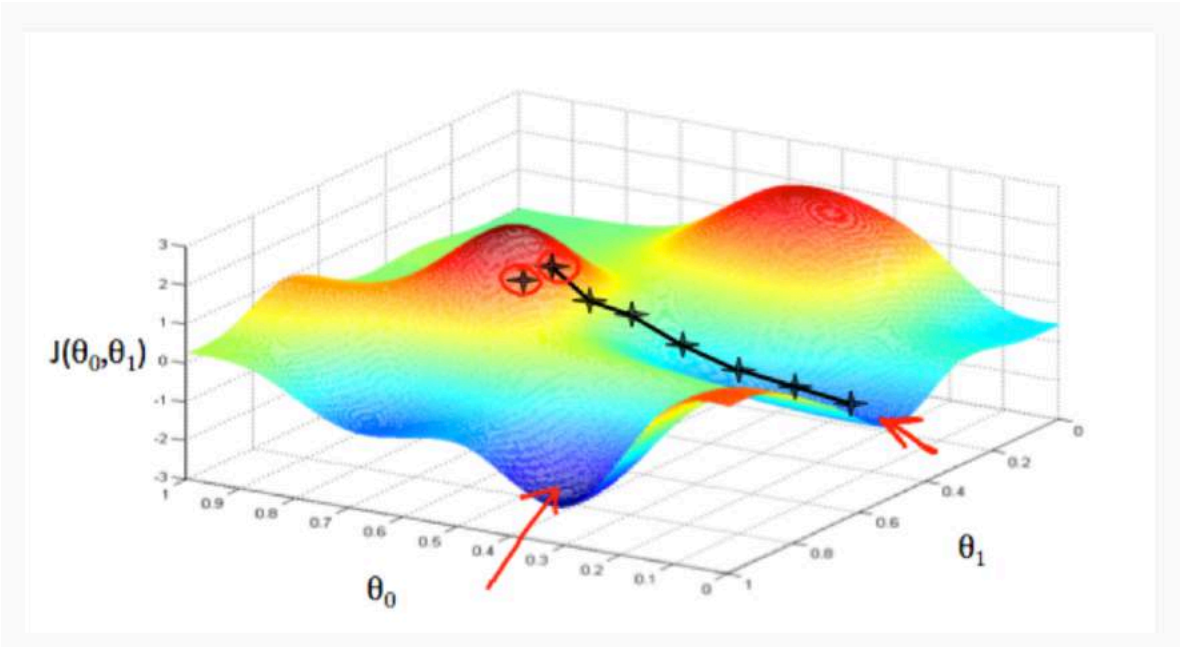
intuitions you need to apply gradient descent.

Gradient Descent (Transcript)

So we have our hypothesis function and we have a way of measuring how well it fits into the data. Now we need to estimate the parameters in the hypothesis function. That's where gradient descent comes in.

Imagine that we graph our hypothesis function based on its fields θ0 and θ1 (actually we are graphing the cost function as a function of the parameter estimates). We are not graphing x and y itself, but the parameter range of our hypothesis function and the cost resulting from selecting a particular set of parameters.

We put θ0 on the x axis and θ1 on the y axis, with the cost function on the vertical z axis. The points on our graph will be the result of the cost function using our hypothesis with those specific theta parameters. The graph below depicts such a setup.



We will know that we have succeeded when our cost function is at the very bottom of the pits in our graph, i.e. when its value is the minimum. The red arrows show the minimum points in the graph.

The way we do this is by taking the derivative (the tangential line to a function)

of our cost function. The slope of the tangent is the derivative at that point and it will give us a direction to move towards. We make steps down the cost function in the direction with the steepest descent. The size of each step is determined by the parameter α, which is called the learning rate.

For example, the distance between each 'star' in the graph above represents a step determined by our parameter α. A smaller α would result in a smaller step and a larger α results in a larger step. The direction in which the step is taken is determined by the partial derivative of J(θ0,θ1). Depending on where one starts on the graph, one could end up at different points. The image above shows us two different starting points that end up in two different places.

The gradient descent algorithm is:

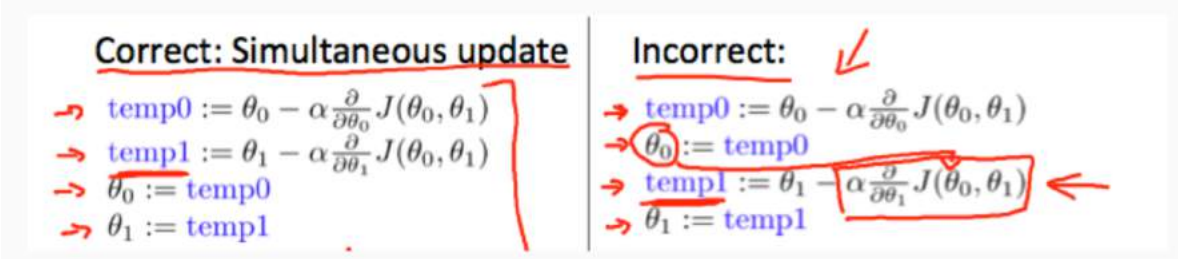
repeat until convergence:

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$

where

j=0,1 represents the feature index number.

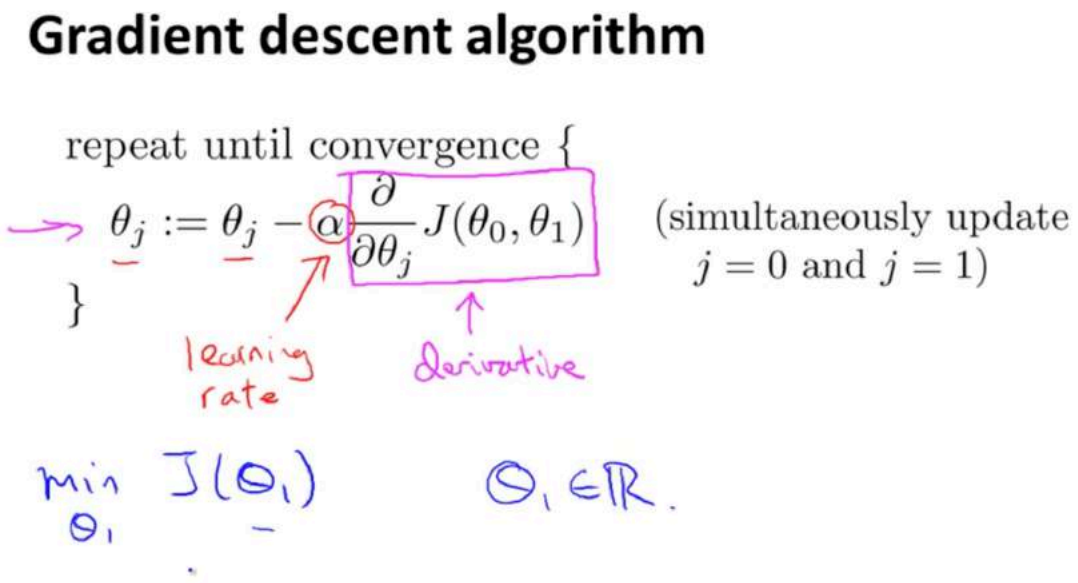
At each iteration j, one should simultaneously update the parameters θ1,θ2,...,θn. Updating a specific parameter prior to calculating another one on the j(th) iteration would yield to a wrong implementation.



Gradient Descent Intuition (Video)

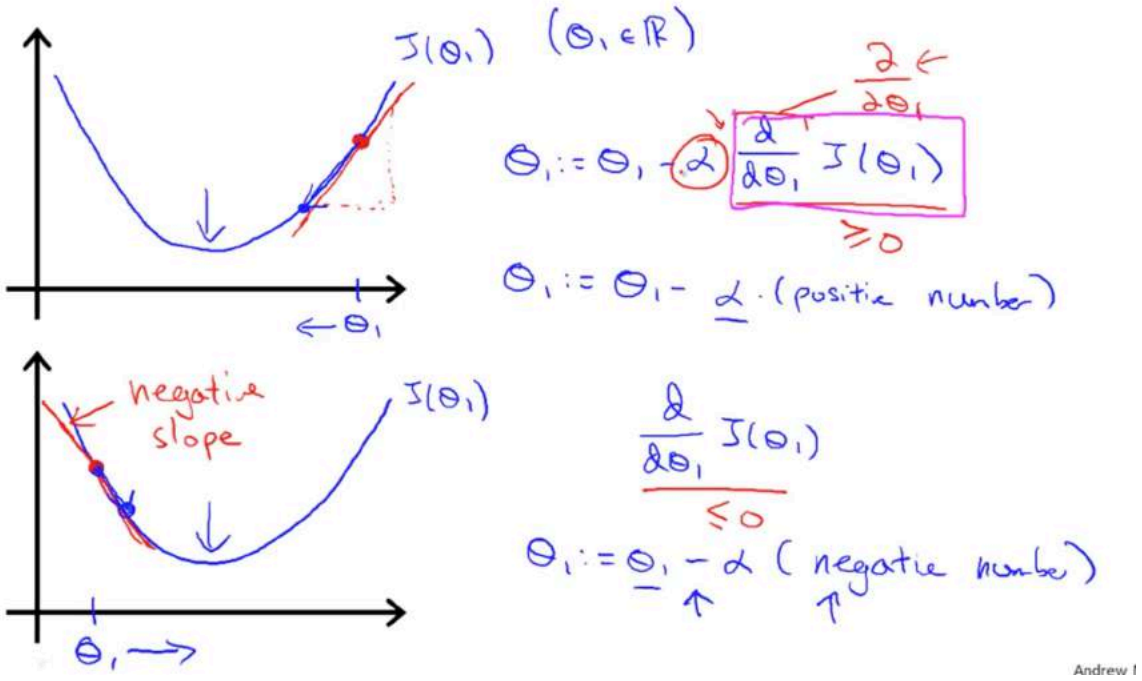
In the previous video, we gave a mathematical definition of gradient descent.

Let's delve deeper and in this video get better intuition about what the algorithm is doing and why the steps of the gradient descent algorithm might make sense. Here's a gradient descent algorithm that we saw last time and just to remind you this parameter, or this term alpha is called the learning rate. And it controls how big a step we take when updating my parameter theory j. And this second term here is the derivative term And what I wanna do in this video is give you that intuition about what each of these two terms is doing and why when put together, this entire update makes sense. In order to convey these intuitions, what I want to do is use a slightly simpler example, where we want to minimize the function of just one parameter. So say we have a cost function, j of just one parameter, theta one, like we did a few videos back, where theta one is a real number. So we can have one d plots, which are a little bit simpler to look at. Let's try to understand what gradient decent would do on this function.



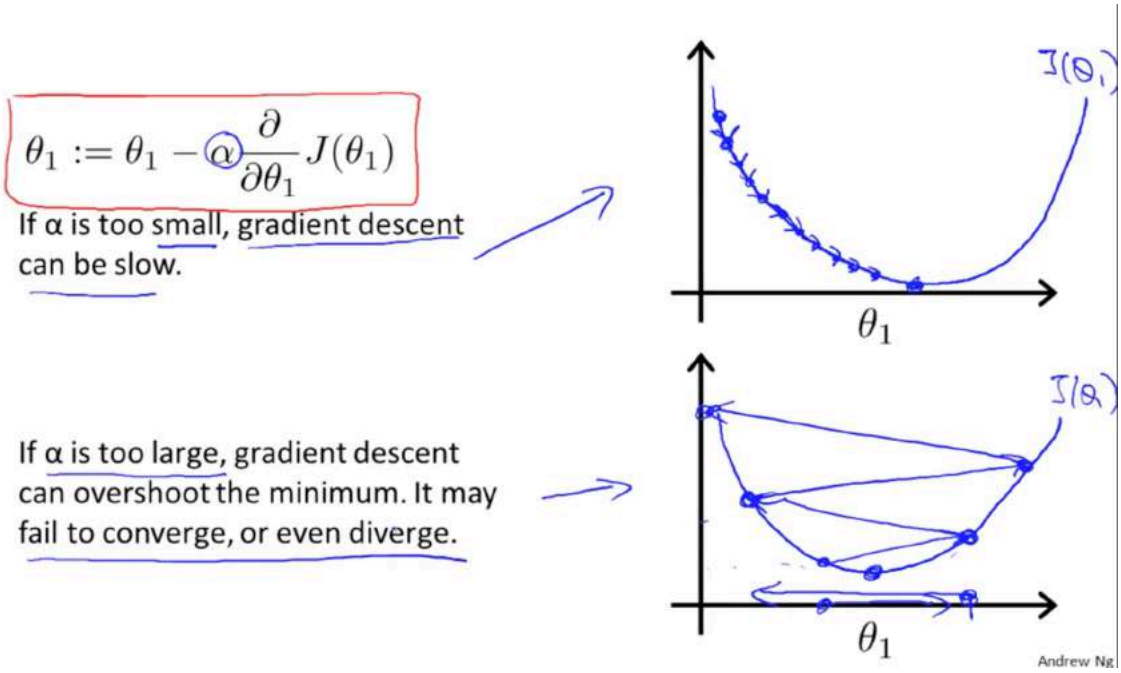
So let's say, here's my function, J of theta 1. And so that's mine. And where theta 1 is a real number. All right? Now, let's have in this slide its grade in descent with theta one at this location. So imagine that we start off at that point on my function. What grade in descent would do is it will update. Theta one gets updated as theta one minus alpha times d d theta one J of theta one, right? And as an aside, this derivative term, right, if you're wondering why I changed the notation from these partial derivative symbols. If you don't know what the difference is between these partial derivative symbols and the dd theta, don't worry about it. Technically in mathematics you call this a partial derivative and call this a derivative, depending on the number of parameters in the function J. But that's a mathematical technicality. And so for the purpose of this lecture, think of these partial symbols and d, d theta 1, as exactly the same thing. And don't worry about what the real difference is. I'm gonna try to

use the mathematically precise notation, but for our purposes these two notations are really the same thing. And so let's see what this equation will do. So we're going to compute this derivative, not sure if you've seen derivatives in calculus before, but what the derivative at this point does, is basically saying, now let's take the tangent to that point, like that straight line, that red line, is just touching this function, and let's look at the slope of this red line. That's what the derivative is, it's saying what's the slope of the line that is just tangent to the function. Okay, the slope of a line is just this height divided by this horizontal thing. Now, this line has a positive slope, so it has a positive derivative. And so my update to theta is going to be theta 1, it gets updated as theta 1, minus alpha times some positive number. Okay. Alpha the the learning, is always a positive number. And, so we're going to take theta one is updated as theta one minus something. So I'm gonna end up moving theta one to the left. I'm gonna decrease theta one, and we can see this is the right thing to do cuz I actually wanna head in this direction. You know, to get me closer to the minimum over there. So, gradient descent so far says we're going the right thing. Let's look at another example. So let's take my same function J, let's try to draw from the same function, J of theta 1. And now, let's say I had to say initialize my parameter over there on the left. So theta 1 is here. I glare at that point on the surface. Now my derivative term DV theta one J of theta one when you value into that this point, we're gonna look at right the slope of that line, so this derivative term is a slope of this line. But this line is slanting down, so this line has negative slope. Right. Or alternatively, I say that this function has negative derivative, just means negative slope at that point. So this is less than equals to 0, so when I update theta, I'm gonna have theta. Just update this theta of minus alpha times a negative number. And so I have theta 1 minus a negative number which means I'm actually going to increase theta, because it's minus of a negative number, means I'm adding something to theta. And what that means is that I'm going to end up increasing theta until it's not here, and increase theta wish again seems like the thing I wanted to do to try to get me closer to the minimum.



So this whole theory of intuition behind what a derivative is doing, let's take a look at the rate term alpha and see what that's doing. So here's my gradient descent update mural, that's this equation. And let's look at what could happen if alpha is either too small or if alpha is too large. So this first example, what happens if alpha is too small? So here's my function J, J of theta. Let's all start here. If alpha is too small, then what I'm gonna do is gonna multiply my update by some small number, so end up taking a baby step like that. Okay, so this one step. Then from this new point, I'm gonna have to take another step. But if alpha's too small, I take another little baby step. And so if my learning rate is too small I'm gonna end up taking these tiny tiny baby steps as you try to get to the minimum. And I'm gonna need a lot of steps to get to the minimum and so if alpha is too small gradient descent can be slow because it's gonna take these tiny tiny baby steps and so it's gonna need a lot of steps before it gets anywhere close to the global minimum. Now how about if our alpha is too large? So, here's my function Jf filter, turns out that alpha's too large, then gradient descent can overshoot the minimum and may even fail to convert or even divert, so here's what I mean. Let's say it's all our data there, it's actually close to minimum. So the derivative points to the right, but if alpha is too big, I want to take a huge step. Remember, take a huge step like that. So it ends up taking a huge step, and now my cost functions have strong roots. Cuz it starts off with this value, and now, my values are strong in verse. Now my derivative points to the left, it says I should decrease data. But if my learning is too big, I may take a huge step going from here all the way to out there. So we end up being over there, right? And if my is too big, we can take another huge step on the next elevation and kind of overshoot and overshoot and so on, until you already notice I'm actually getting further and further away from the minimum.

So if alpha is to large, it can fail to converge or even diverge.

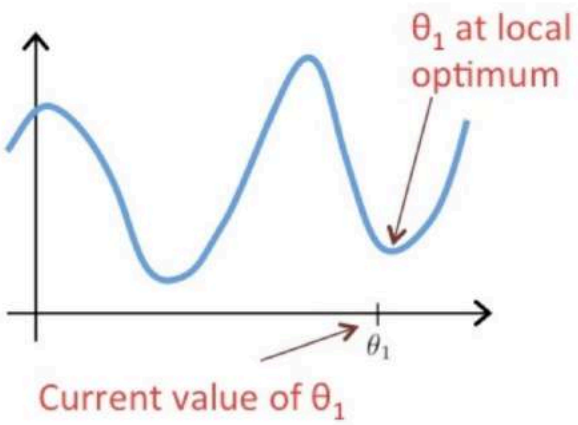


Question:

Now, I have another question for you. So this a tricky one and when I was first learning this stuff it actually took me a long time to figure this out. What if your parameter theta 1 is already at a local minimum, what do you think one step of gradient descent will do?

Suppose θ_1 is at a local optimum of $J(\theta_1)$, such as shown in the figure.

What will one step of gradient descent $\theta_1 := \theta_1 - \alpha \frac{d}{d\theta_1} J(\theta_1)$ do?

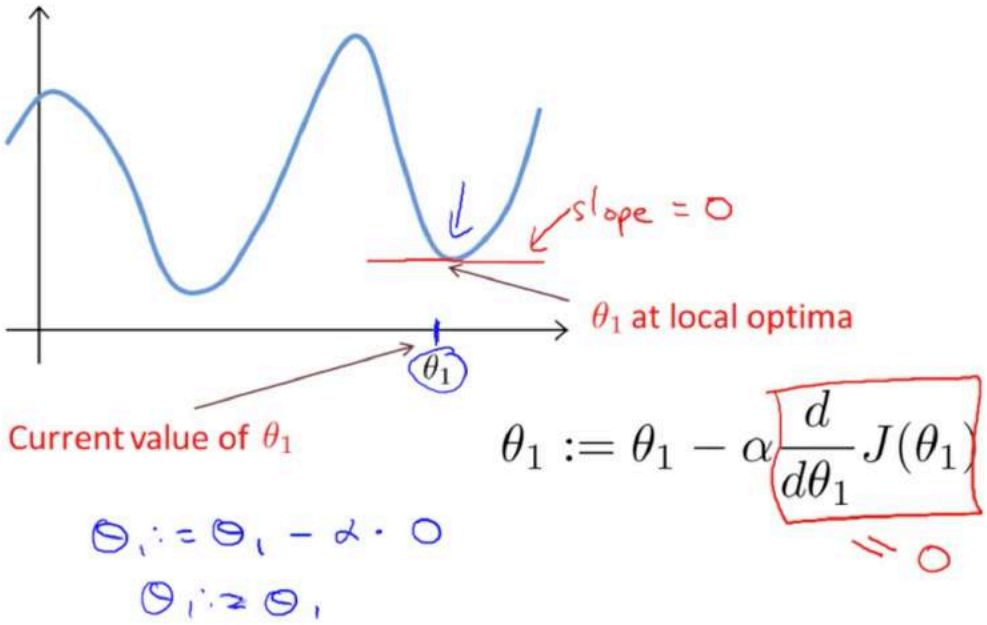


- (•) Leave θ_1 unchanged

Correct

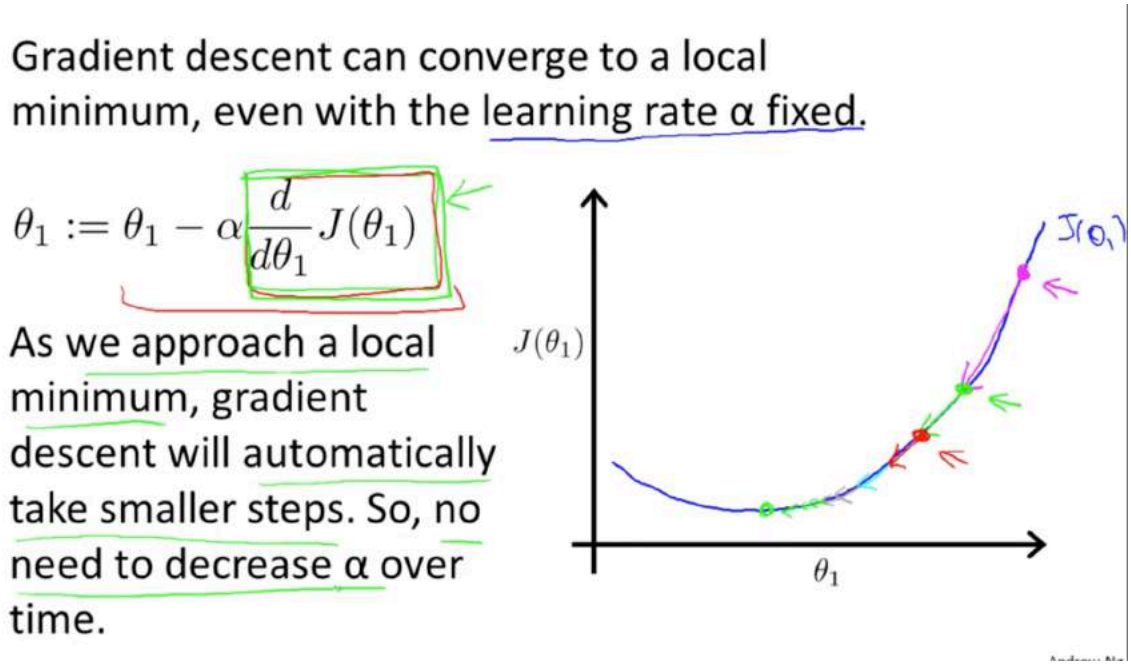
- ( ) Change θ_1 in a random direction
- ( ) Move θ_1 in the direction of the global minimum of $J(\theta_1)$
- ( ) Decrease θ_1

Explanation:
So let's suppose you initialize theta 1 at a local minimum. So, suppose this is your initial value of theta 1 over here and is already at a local optimum or the local minimum. It turns out the local optimum, your derivative will be equal to zero. So for that slope, that tangent point, so the slope of this line will be equal to zero and thus this derivative term is equal to zero. And so your gradient descent update, you have theta one cuz I updated this theta one minus alpha times zero. And so what this means is that if you're already at the local optimum it leaves theta 1 unchanged cause its updates as theta 1 equals theta 1. So if your parameters are already at a local minimum one step with gradient descent does absolutely nothing it doesn't your parameter which is what you want because it keeps your solution at the local optimum.



This also explains why gradient descent can converse the local minimum even with the learning rate alpha fixed. Here's what I mean by that let's look in the

example. So here's a cost function J of theta that maybe I want to minimize and let's say I initialize my algorithm, my gradient descent algorithm, out there at that magenta point. If I take one step in gradient descent, maybe it will take me to that point, because my derivative's pretty steep out there. Right? Now, I'm at this green point, and if I take another step in gradient descent, you notice that my derivative, meaning the slope, is less steep at the green point than compared to at the magenta point out there. Because as I approach the minimum, my derivative gets closer and closer to zero, as I approach the minimum. So after one step of descent, my new derivative is a little bit smaller. So I wanna take another step in the gradient descent. I will naturally take a somewhat smaller step from this green point right there from the magenta point. Now with a new point, a red point, and I'm even closer to global minimum so the derivative here will be even smaller than it was at the green point. So I'm gonna another step in the gradient descent. Now, my derivative term is even smaller and so the magnitude of the update to theta one is even smaller, so take a small step like so. And as gradient descent runs, you will automatically take smaller and smaller steps. Until eventually you're taking very small steps, you know, and you finally converge to the to the local minimum. So just to recap, in gradient descent as we approach a local minimum, gradient descent will automatically take smaller steps. And that's because as we approach the local minimum, by definition the local minimum is when the derivative is equal to zero. As we approach local minimum, this derivative term will automatically get smaller, and so gradient descent will automatically take smaller steps. This is what so no need to decrease alpha or the time.



So that's the gradient descent algorithm and you can use it to try to minimize any cost function J, not the cost function J that we defined for linear regression. In the next video, we're going to take the function J and set that

back to be exactly linear regression's cost function, the square cost function that we came up with earlier. And taking gradient descent and this great cause function and putting them together. That will give us our first learning algorithm, that'll give us a linear regression algorithm.

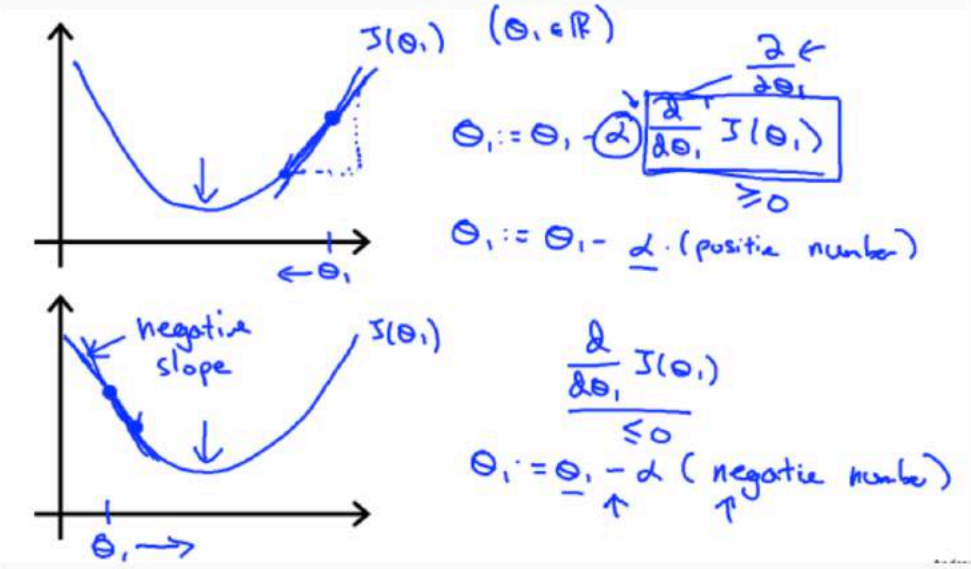
Gradient Descent Intuition (Transcript)

In this video we explored the scenario where we used one parameter θ1 and plotted its cost function to implement a gradient descent. Our formula for a single parameter was :

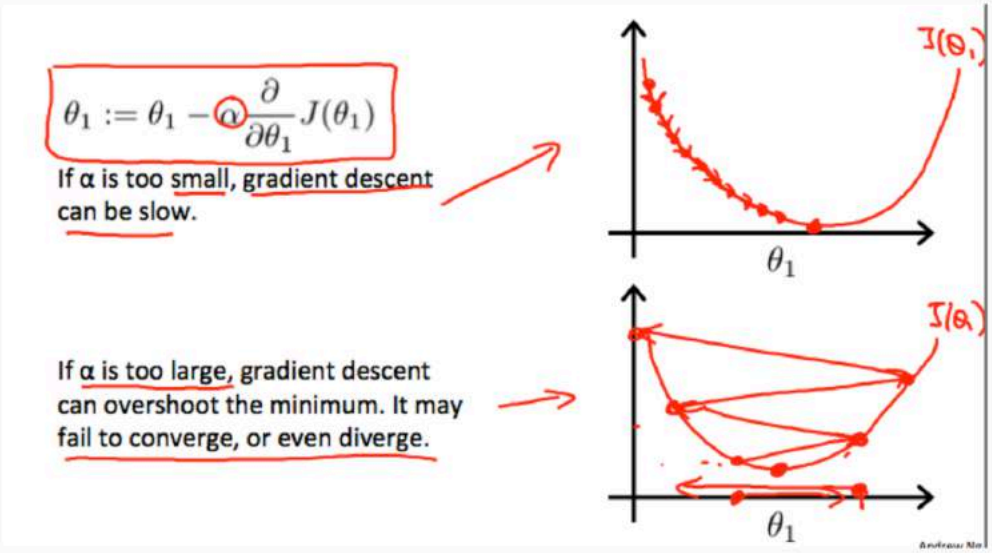
Repeat until convergence:

$$\theta_1 := \theta_1 - \alpha \frac{d}{d\theta_1} J(\theta_1)$$

Regardless of the slope's sign for $\frac{d}{d\theta_1} J(\theta_1)$, θ_1 eventually converges to its minimum value. The following graph shows that when the slope is negative, the value of θ_1 increases and when it is positive, the value of θ_1 decreases.



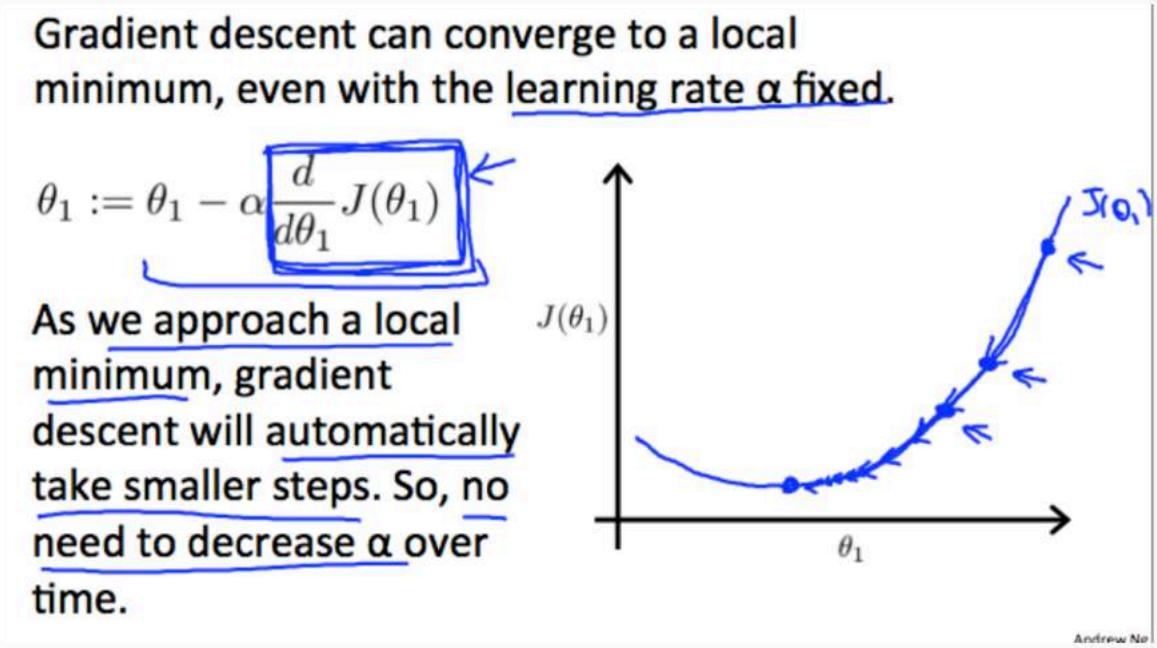
On a side note, we should adjust our parameter α to ensure that the gradient descent algorithm converges in a reasonable time. Failure to converge or too much time to obtain the minimum value imply that our step size is wrong.



How does gradient descent converge with a fixed step size α?

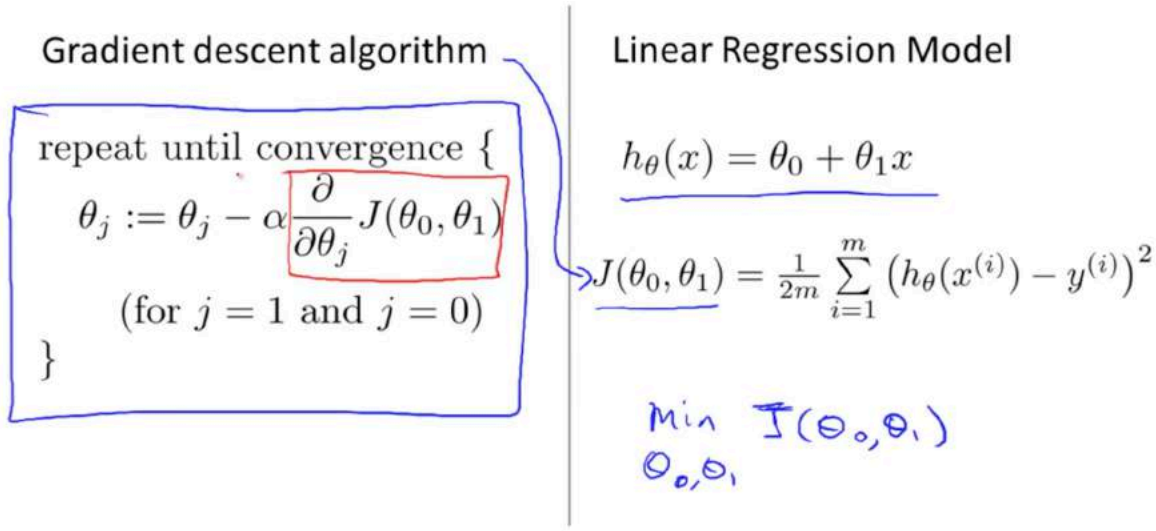
The intuition behind the convergence is that $\frac{d}{d\theta_1} J(\theta_1)$ approaches 0 as we approach the bottom of our convex function. At the minimum, the derivative will always be 0 and thus we get:

$$\theta_1 := \theta_1 - \alpha * 0$$



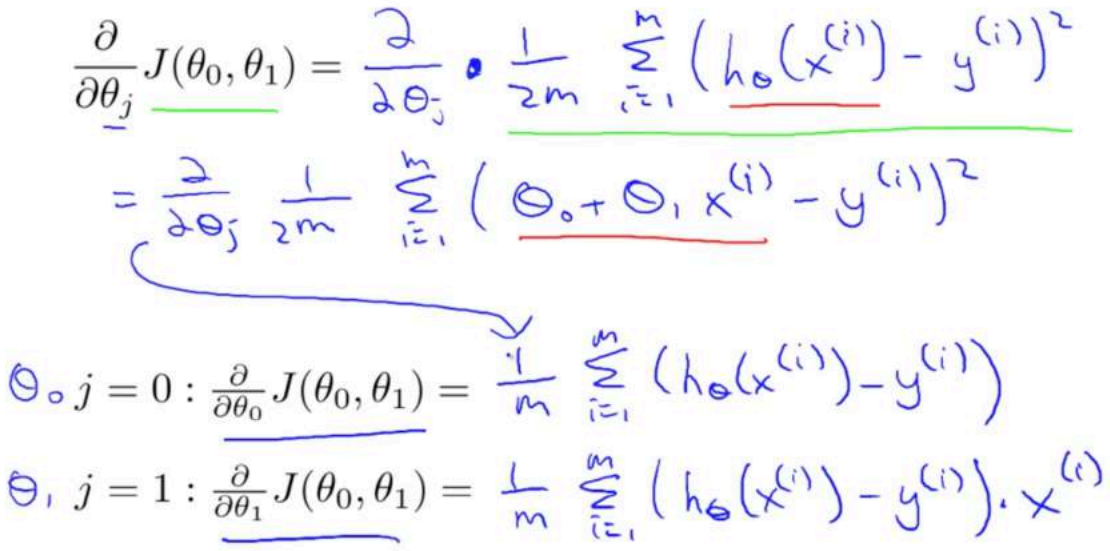
Gradient Descent for Linear Regression (Video)

In previous videos, we talked about the gradient descent algorithm and we talked about the linear regression model and the squared error cost function. In this video we're gonna put together gradient descent with our cost function, and that will give us an algorithm for linear regression or putting a straight line to our data. So this was what we worked out in the previous videos. This gradient descent algorithm which you should be familiar and here's the linear regression model with our linear hypothesis and our squared error cost function. What we're going to do is apply gradient descent to minimize our squared error cost function. Now in order to apply gradient descent, in order to, you know, write this piece of code, the key term we need is this derivative term over here.



So you need to figure out what is this partial derivative term and plugging in the definition of the cause function j. This turns out to be this. Sum from y equals 1 though m. Of this squared error cost function term. And all I did here was I just, you know plug in the definition of the cost function there. And simplifying a little bit more, this turns out to be equal to this. Sigma i equals one through m of theta zero plus theta one x i minus Yi squared. And all I did there was I took the definition for my hypothesis and plugged it in there. And turns out we need to figure out what is this partial derivative for two cases for J equals 0 and J equals 1. So we want to figure out what is this partial derivative for both the theta 0 case and the theta 1 case. And I'm just going to write out the answers. It turns out this first term is, simplifies to 1/M sum from

over my training step of just that of X(i)- Y(i) and for this term partial derivative let's write the theta 1, it turns out I get this term. Minus Y(i) times X(i). Okay and computing these partial derivatives, so we're going from this equation. Right going from this equation to either of the equations down there. Computing those partial derivative terms requires some multivariate calculus. If you know calculus, feel free to work through the derivations yourself and check that if you take the derivatives, you actually get the answers that I got. But if you're less familiar with calculus, don't worry about it and it's fine to just take these equations that were worked out and you won't need to know calculus or anything like that, in order to do the homework so let's implement gradient descent and get back to work.



So armed with these definitions or armed with what we worked out to be the derivatives which is really just the slope of the cost function j we can now plug them back in to our gradient descent algorithm. So here's gradient descent for linear regression which is gonna repeat until convergence, theta 0 and theta 1 get updated as you know this thing minus alpha times the derivative term. So this term here. So here's our linear regression algorithm. This first term here. That term is of course just the partial derivative with respect to theta zero, that we worked out on a previous slide. And this second term here, that term is just a partial derivative in respect to theta 1, that we worked out on the previous line. And just as a quick reminder, you must, when implementing gradient descent. There's actually this detail that you should be implementing it so the update theta 0 and theta 1 simultaneously.

Gradient descent algorithm

$\frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$

repeat until convergence {

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)$$

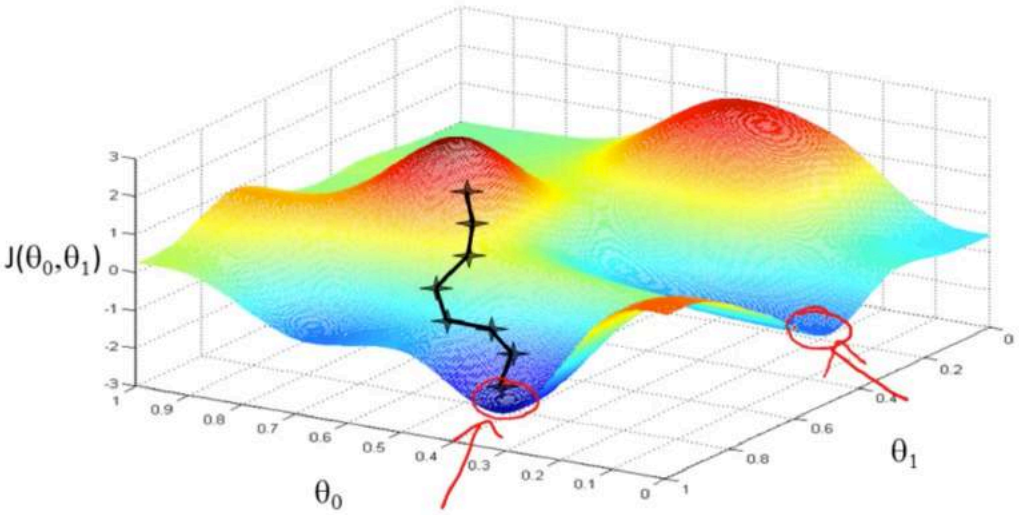
$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right) \cdot x^{(i)}$$

}

update θ_0 and θ_1 simultaneously

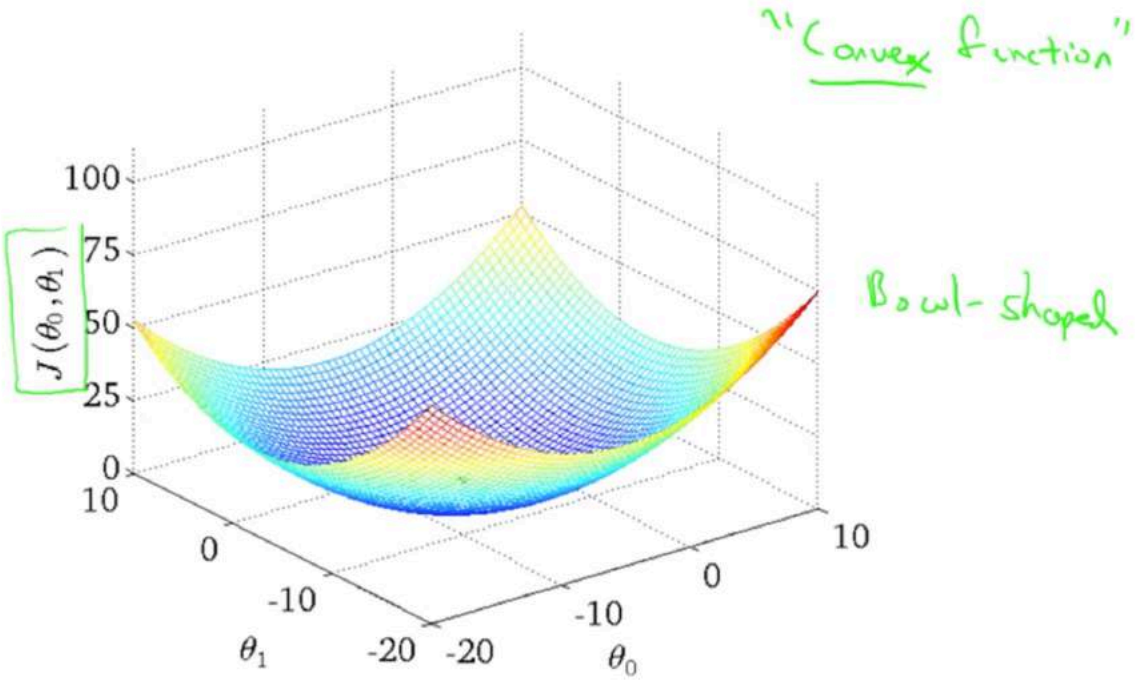
$\frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$

So. Let's see how gradient descent works. One of the issues we saw with gradient descent is that it can be susceptible to local optima. So when I first explained gradient descent I showed you this picture of it going downhill on the surface, and we saw how depending on where you initialize it, you can end up at different local optima. You will either wind up here or here.

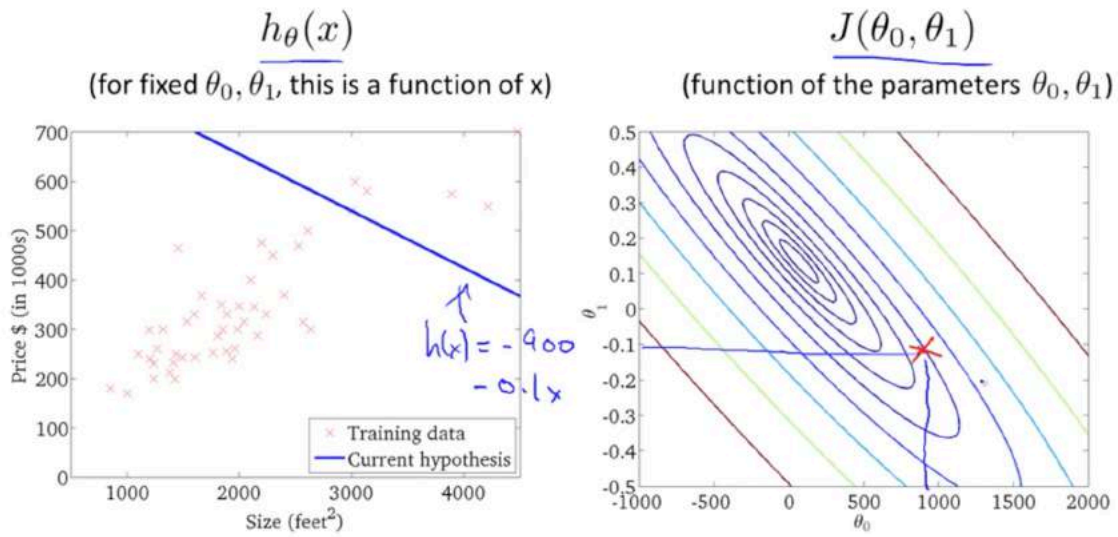


But, it turns out that that the cost function for linear regression is always going to be a bow shaped function like this. The technical term for this is that this is called a convex function. And I'm not gonna give the formal definition for what is a convex function, C, O, N, V, E, X. But informally a convex function means a bowl shaped function and so this function doesn't have any local optima

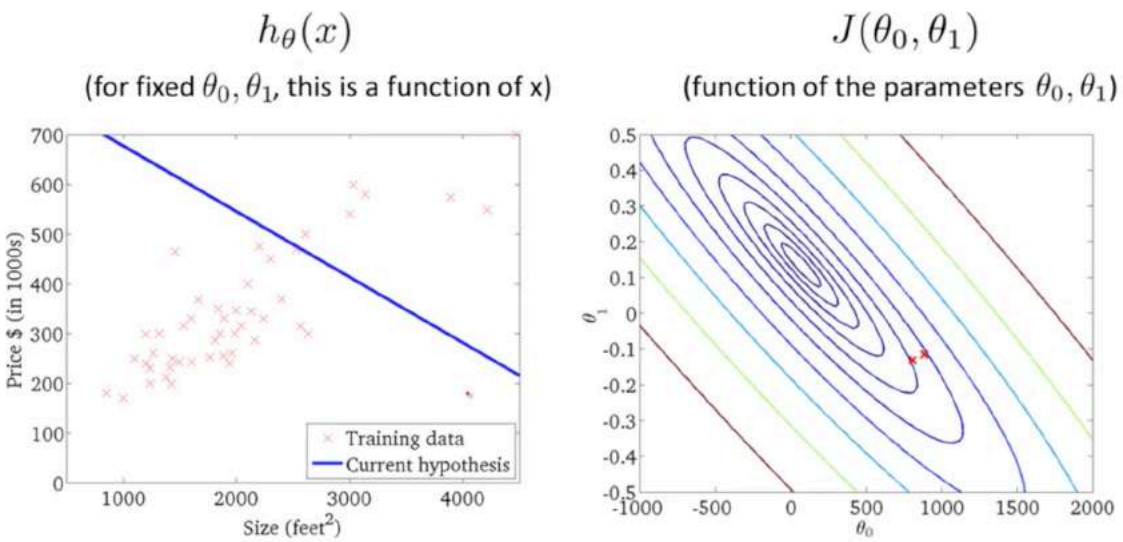
except for the one global optimum. And does gradient descent on this type of cost function which you get whenever you're using linear regression it will always converge to the global optimum. Because there are no other local optimum, global optimum.



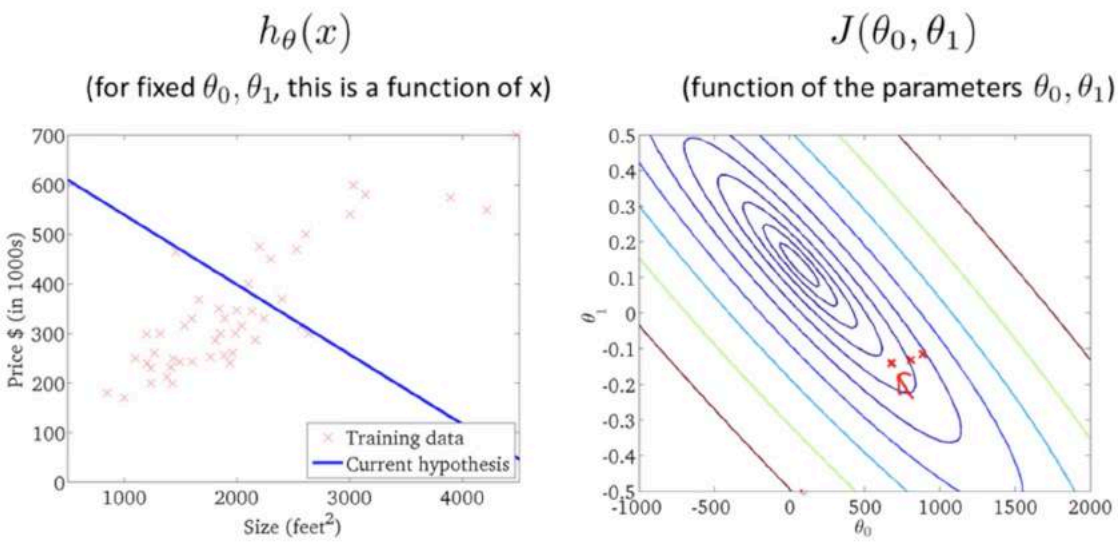
So now let's see this algorithm in action. As usual, here are plots of the hypothesis function and of my cost function j. And so let's say I've initialized my parameters at this value. Let's say, usually you initialize your parameters at zero, zero. Theta zero and theta equals zero. But for the demonstration, in this physical infrontation I've initialized you know, theta zero at 900 and theta one at about -0.1 okay. And so this corresponds to h(x)=-900-0.1x, [the intercept should be +900] is this line, out here on the cost function.



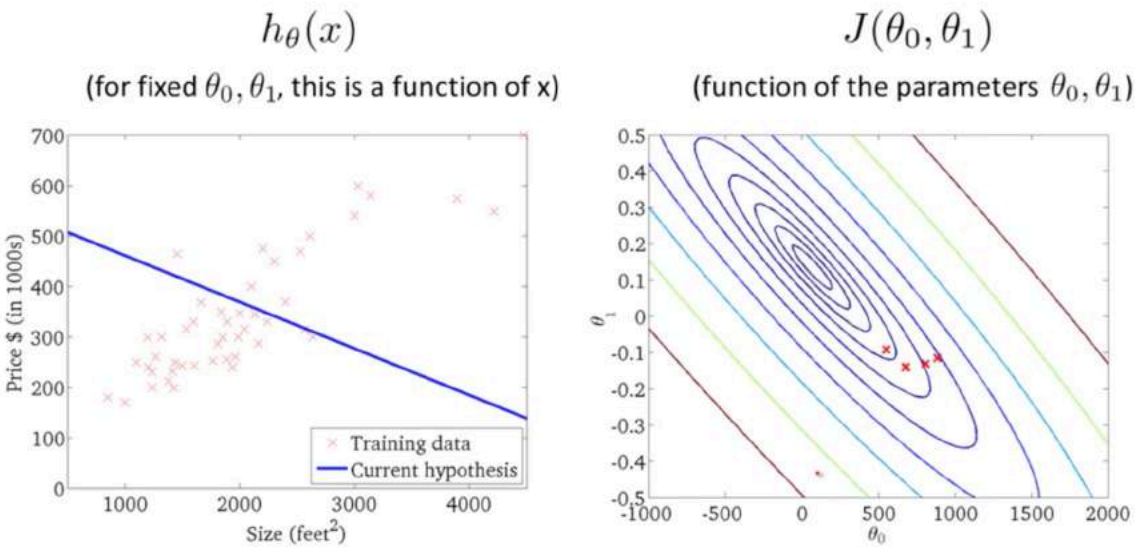
Now, if we take one step in gradient descent, we end up going from this point out here, over to the down and left, to that second point over there. And you notice that my line changed a little bit



And as I take another step of gradient descent, my line on the left will change. Right? And I've also moved to a new point on my cost function.

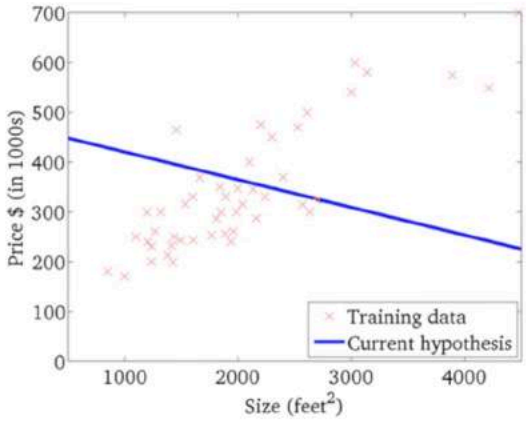


And as I take further steps of gradient descent, I'm going down in cost. So my parameters and such are following this trajectory.



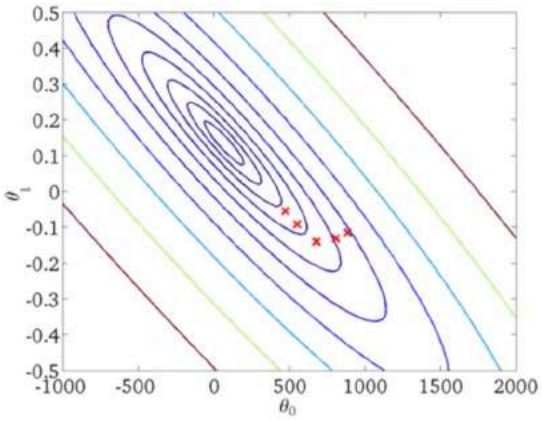
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



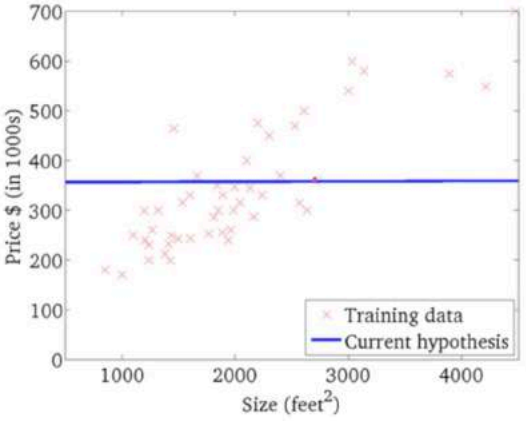
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



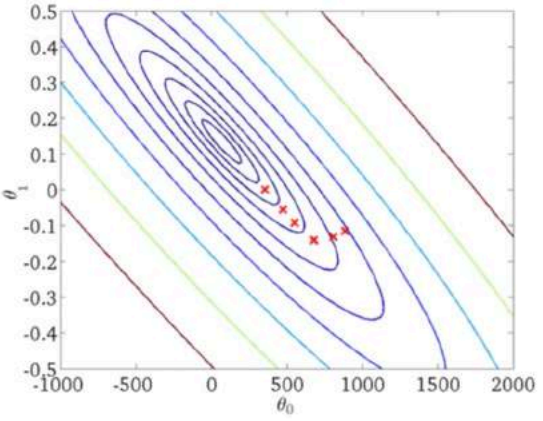
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



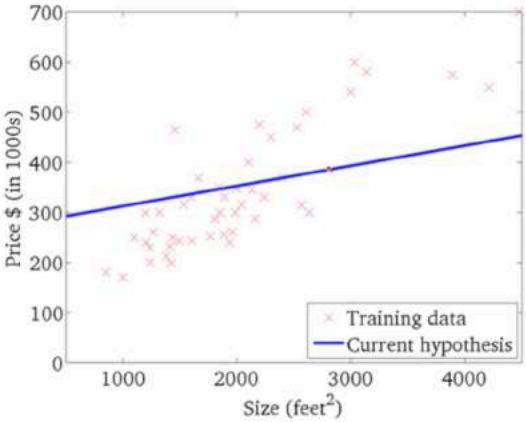
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



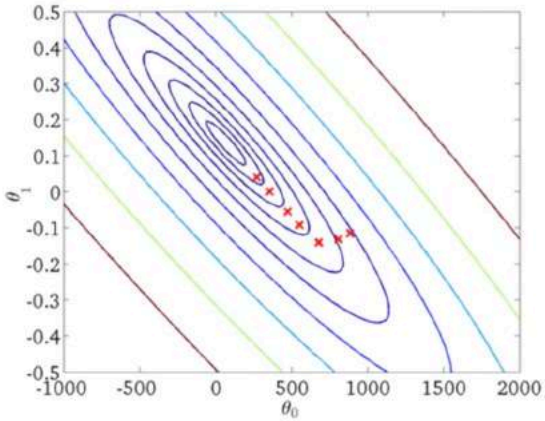
$h_\theta(x)$

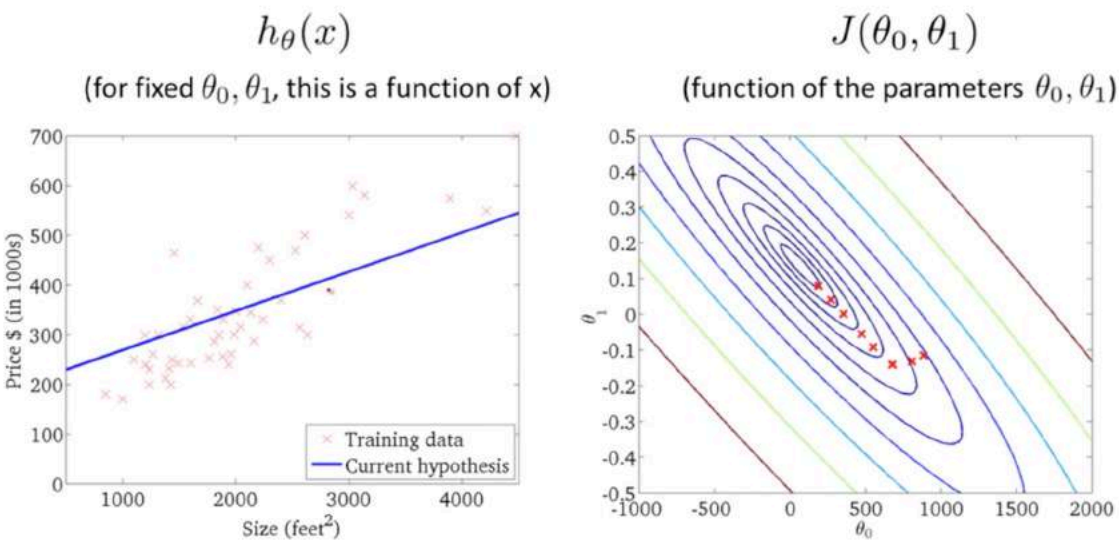
(for fixed θ_0, θ_1, this is a function of x)



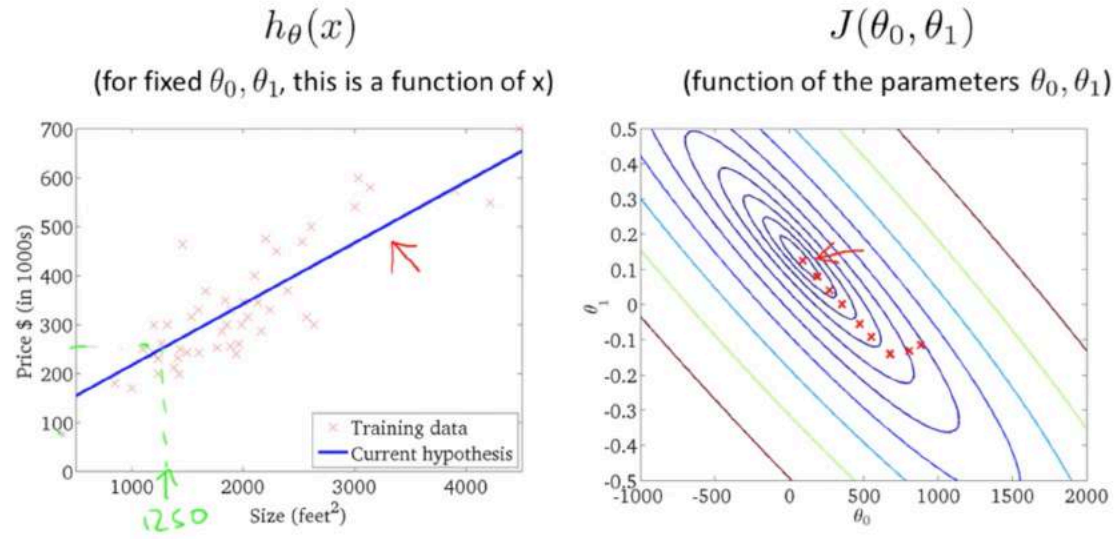
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



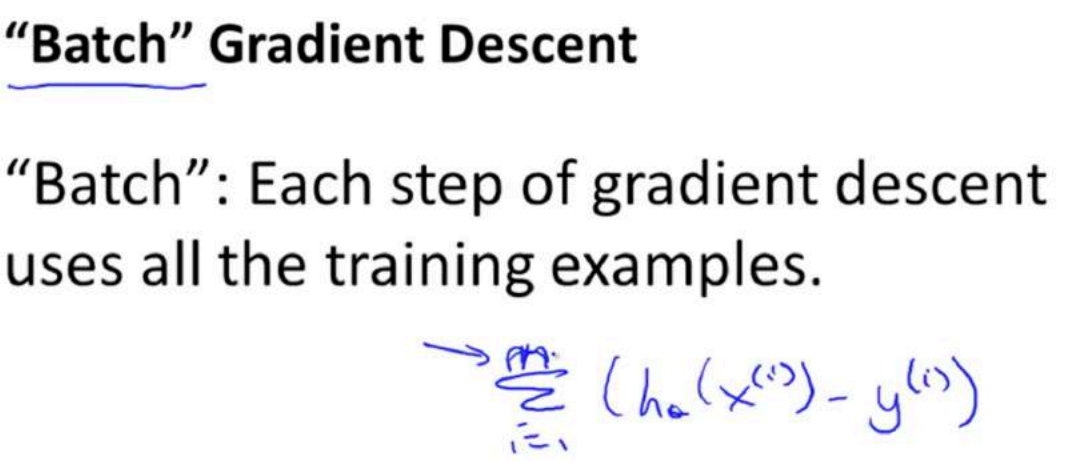


And if you look on the left, this corresponds with hypotheses. That seem to be getting to be better and better fits to the data until eventually I've now wound up at the global minimum and this global minimum corresponds to this hypothesis, which gets me a good fit to the data. And so that's gradient descent, and we've just run it and gotten a good fit to my data set of housing prices. And you can now use it to predict, you know, if your friend has a house size 1250 square feet, you can now read off the value and tell them that I don't know maybe they could get $250,000 for their house.



Finally just to give this another name it turns out that the algorithm that we just went over is sometimes called batch gradient descent. And it turns out in machine learning I don't know I feel like us machine learning people were not

always great at giving names to algorithms. But the term batch gradient descent refers to the fact that in every step of gradient descent, we're looking at all of the training examples. So in gradient descent, when computing the derivatives, we're computing the sums [INAUDIBLE]. So ever step of gradient descent we end up computing something like this that sums over our m training examples and so the term batch gradient descent refers to the fact that we're looking at the entire batch of training examples. And again, it's really not a great name, but this is what machine learning people call it. And it turns out that there are sometimes other versions of gradient descent that are not batch versions, but they are instead. Do not look at the entire training set but look at small subsets of the training sets at a time. And we'll talk about those versions later in this course as well. But for now using the algorithm we just learned about or using batch gradient descent you now know how to implement gradient descent for linear regression.



Question

Which of the following are true statements? Select all that apply.

- [ ] To make gradient descent converge, we must slowly decrease α over time.
- [ ] Gradient descent is guaranteed to find the global minimum for any function $J(\theta_0, \theta_1)$.
- [x] Gradient descent can converge even if α is kept fixed. (But α cannot be too large, or else it may fail to converge.)
- [x] For the specific choice of cost function $J(\theta_0, \theta_1)$ used in linear regression, there are no local optima (other than the global optimum).

So that's linear regression with gradient descent. If you've seen advanced linear algebra before, so some of you may have taken a class in advanced linear algebra. You might know that there exists a solution for numerically solving for the minimum of the cost function j without needing to use an iterative algorithm like gradient descent. Later in this course we'll talk about that method as well that just solves for the minimum of the cost function j without needing these multiple steps of gradient descent. That other method is called the **normal equations method**. But in case you've heard of that method it turns out that gradient descent will scale better to larger data sets than that normal equation method. And now that we know about gradient descent we'll be able to use it in lots of different contexts and we'll use it in lots of different machine learning problems as well.

So congrats on learning about your first machine learning algorithm. We'll later have exercises in which we'll ask you to implement gradient descent and hopefully see these algorithms right for yourselves. But before that I first want to tell you in the next set of videos. The first one to tell you about a generalization of the gradient descent algorithm that will make it much more powerful. And I guess I'll tell you about that in the next video.

Gradient Descent for Linear Regression (Transcript)

Note: [At 6:15 "h(x) = -900 - 0.1x" should be "h(x) = 900 - 0.1x"]

When specifically applied to the case of linear regression, a new form of the gradient descent equation can be derived. We can substitute our actual cost function and our actual hypothesis function and modify the equation to :

repeat until convergence: {

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i)$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left((h_\theta(x_i) - y_i) x_i \right)$$

}

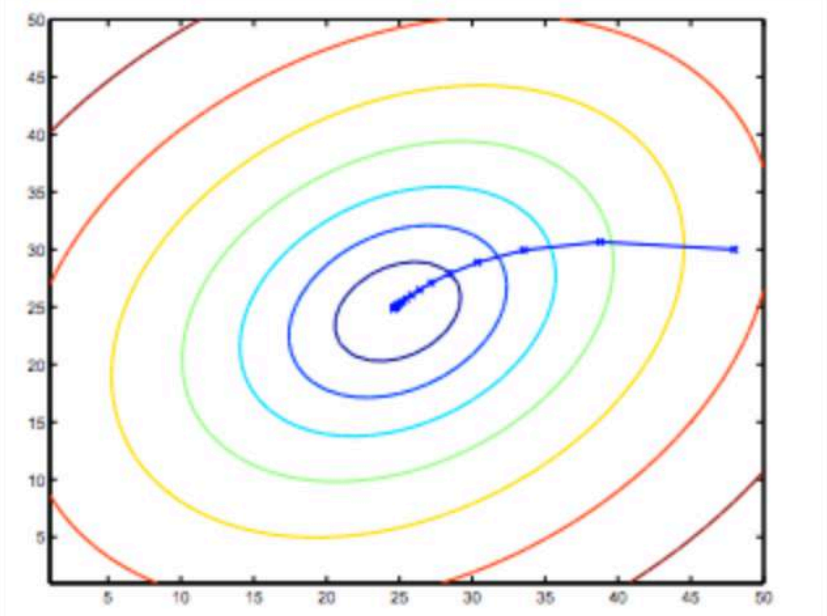
where m is the size of the training set, θ_0 a constant that will be changing simultaneously with θ_1 and x_i, y_i are values of the given training set (data).

Note that we have separated out the two cases for θ_j into separate equations for θ_0 and θ_1; and that for θ_1 we are multiplying x_i at the end due to the derivative. The following is a derivation of $\frac{\partial}{\partial \theta_j} J(\theta)$ for a single example :

$$\begin{aligned}
\frac{\partial}{\partial \theta_j} J(\theta) &= \frac{\partial}{\partial \theta_j} \frac{1}{2} (h_\theta(x) - y)^2 \\
&= 2 \cdot \frac{1}{2} (h_\theta(x) - y) \cdot \frac{\partial}{\partial \theta_j} (h_\theta(x) - y) \\
&= (h_\theta(x) - y) \cdot \frac{\partial}{\partial \theta_j} \left(\sum_{i=0}^{n} \theta_i x_i - y \right) \\
&= (h_\theta(x) - y) \, x_j
\end{aligned}$$

The point of all this is that if we start with a guess for our hypothesis and then repeatedly apply these gradient descent equations, our hypothesis will become more and more accurate.

So, this is simply gradient descent on the original cost function J. This method looks at every example in the entire training set on every step, and is called batch gradient descent. Note that, while gradient descent can be susceptible to local minima in general, the optimization problem we have posed here for linear regression has only one global, and no other local, optima; thus gradient descent always converges (assuming the learning rate α is not too large) to the global minimum. Indeed, J is a convex quadratic function. Here is an example of gradient descent as it is run to minimize a quadratic function.



The ellipses shown above are the contours of a quadratic function. Also shown is the trajectory taken by gradient descent, which was initialized at (48,30).

The x's in the figure (joined by straight lines) mark the successive values of θ that gradient descent went through as it converged to its minimum.

Review

QUIZ: Linear Regression with one variable

1.

1. Consider the problem of predicting how well a student does in her second year of college/university, given how well she did in her first year.

Specifically, let x be equal to the number of "A" grades (including A-. A and A+ grades) that a student receives in their first year of college (freshmen year). We would like to predict the value of y, which we define as the number of "A" grades they get in their second year (sophomore year).

Refer to the following training set of a small sample of different students' performances (note that this training set may also be referenced in other questions in this quiz). Here each row is one training example. Recall that in linear regression, our hypothesis is $h_\theta(x) = \theta_0 + \theta_1 x$, and we use m to denote the number of training examples.

x	y
3	4
2	1
4	3
0	1

For the training set given above, what is the value of m? In the box below, please enter your answer (which should be a number between 0 and 10).

4

2.

2. Many substances that can burn (such as gasoline and alcohol) have a chemical structure based on carbon atoms; for this reason they are called hydrocarbons. A chemist wants to understand how the number of carbon atoms in a molecule affects how much energy is released when that molecule combusts (meaning that it is burned). The chemist obtains the dataset below. In the column on the right, "kJ/mol" is the unit measuring the amount of energy released.

Name of molecule	Number of hydrocarbons in molecule (x)	Heat release when burned (kJ/mol) (y)
methane	1	-890
ethene	2	-1411
ethane	2	-1560
propane	3	-2220
cyclopropane	3	-2091
butane	4	-2878
pentane	5	-3537
benzene	6	-3268
cycloexane	6	-3920
hexane	6	-4163
octane	8	-5471
napthalene	10	-5157

You would like to use linear regression ($h_\theta(x) = \theta_0 + \theta_1 x$) to estimate the amount of energy released (y) as a function of the number of carbon atoms (x). Which of the following do you think will be the values you obtain for θ_0 and θ_1? You should be able to select the right answer without actually implementing linear regression.

- $\theta_0 = -1780.0, \theta_1 = 530.9$
- $\theta_0 = -1780.0, \theta_1 = -530.9$
- $\theta_0 = -569.6, \theta_1 = 530.9$
- $\theta_0 = -569.6, \theta_1 = -530.9$

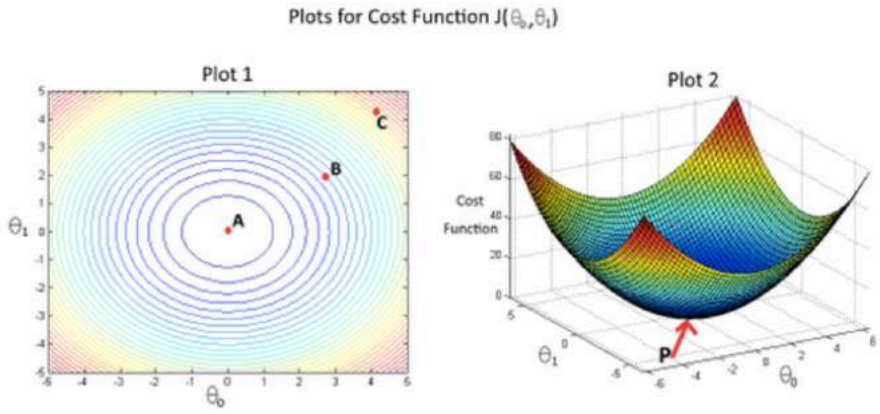
3.

3. Suppose we set $\theta_0 = -1, \theta_1 = 0.5$. What is $h_\theta(4)$?

1

4.

4. In the given figure, the cost function $J(\theta_0, \theta_1)$ has been plotted against θ_0 and θ_1, as shown in 'Plot 2'. The contour plot for the same cost function is given in 'Plot 1'. Based on the figure, choose the correct options (check all that apply).



- [ ] If we start from point B, gradient descent with a well-chosen learning rate will eventually help us reach at or near point C, as the value of cost function $J(\theta_0, \theta_1)$ is minimum at point C.
- [ ] If we start from point B, gradient descent with a well-chosen learning rate will eventually help us reach at or near point A, as the value of cost function $J(\theta_0, \theta_1)$ is maximum at point A.
- [ ] Point P (The global minimum of plot 2) corresponds to point C of Plot 1.
- [x] If we start from point B, gradient descent with a well-chosen learning rate will eventually help us reach at or near point A, as the value of cost function $J(\theta_0, \theta_1)$ is minimum at A.
- [x] Point P (the global minimum of plot 2) corresponds to point A of Plot 1.

5.

5. Suppose that for some linear regression problem (say, predicting housing prices as in the lecture), we have some training set, and for our training set we managed to find some θ_0, θ_1 such that $J(\theta_0, \theta_1) = 0$.

Which of the statements below must then be true? (Check all that apply.)

- [ ] Gradient descent is likely to get stuck at a local minimum and fail to find the global minimum.
- [ ] For this to be true, we must have $y^{(i)} = 0$ for every value of $i = 1, 2, \ldots, m$.
- [x] Our training set can be fit perfectly by a straight line,

  i.e., all of our training examples lie perfectly on some straight line.
- [ ] For this to be true, we must have $\theta_0 = 0$ and $\theta_1 = 0$

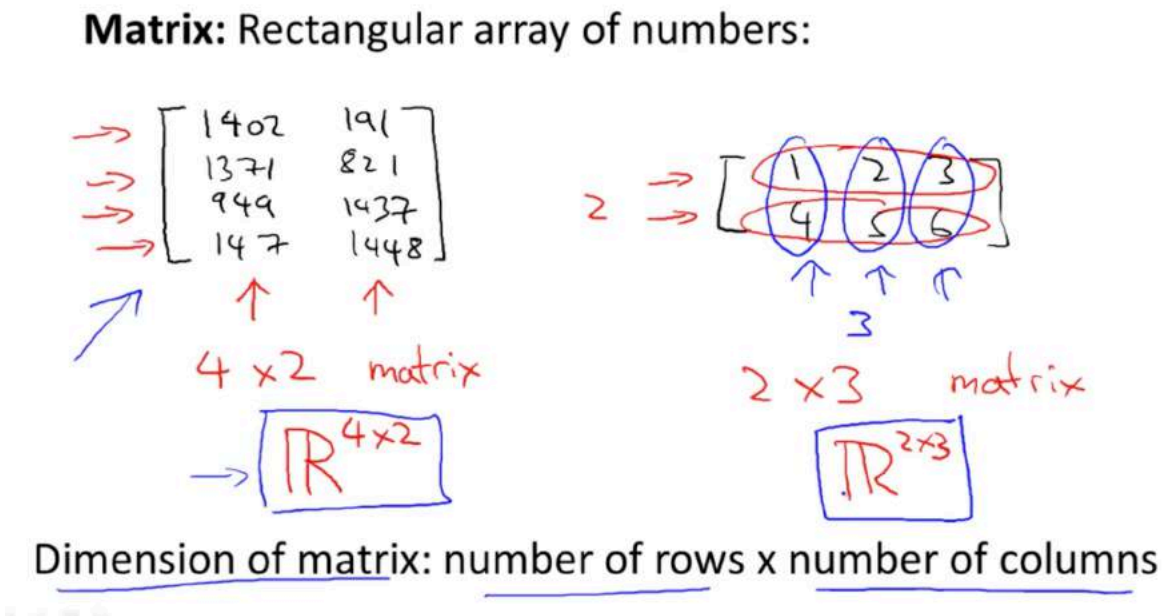
  so that $h_\theta(x) = 0$

Linear Algebra Review

Matrices and Vectors

Let's get started with our linear algebra review. In this video I want to tell you what are matrices and what are vectors. A matrix is a rectangular array of numbers written between square brackets. So, for example, here is a matrix on the right, a left square bracket. And then, write in a bunch of numbers. These could be features from a learning problem or it could be data from somewhere else, but the specific values don't matter, and then I'm going to close it with another right bracket on the right. And so that's one matrix. And, here's another example of the matrix, let's write 3, 4, 5,6. So matrix is just another way for saying, is a 2D or a two dimensional array. And the other piece of

knowledge that we need is that the dimension of the matrix is going to be written as the number of row times the number of columns in the matrix. So, concretely, this example on the left, this has 1, 2, 3, 4 rows and has 2 columns, and so this example on the left is a 4 by 2 matrix - number of rows by number of columns. So, four rows, two columns. This one on the right, this matrix has two rows. That's the first row, that's the second row, and it has three columns. That's the first column, that's the second column, that's the third column So, this second matrix we say it is a 2 by 3 matrix. So we say that the dimension of this matrix is 2 by 3. Sometimes you also see this written out, in the case of left, you will see this written out as R4 by 2 or concretely what people will sometimes say this matrix is an element of the set R 4 by 2. So, this thing here, this just means the set of all matrices that of dimension 4 by 2 and this thing on the right, sometimes this is written out as a matrix that is an R 2 by 3. So if you ever see, 2 by 3. So if you ever see something like this are 4 by 2 or are 2 by 3, people are just referring to matrices of a specific dimension.

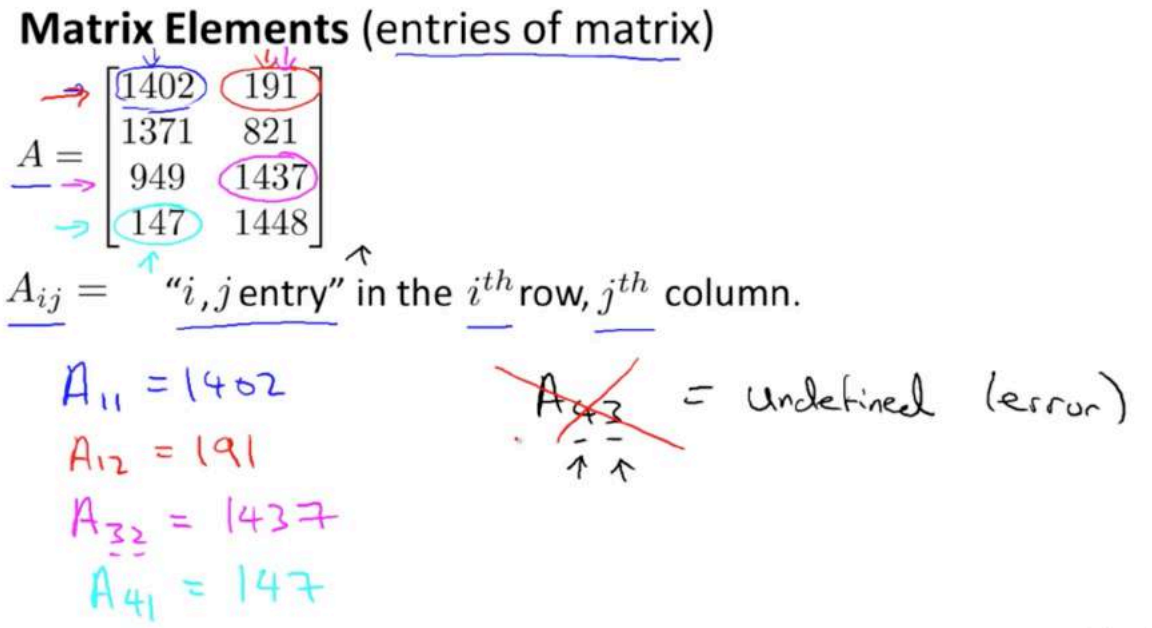


Question

Which of the following statements are true? Check all that apply.

- [x] $\begin{bmatrix} 1 & 2 \\ 4 & 0 \\ 0 & 1 \end{bmatrix}$ is a 3×2 matrix.
- [ ] $\begin{bmatrix} 0 & 1 & 4 & 2 \\ 3 & 4 & 0 & 9 \end{bmatrix}$ is a 4×2 matrix.
- [x] $\begin{bmatrix} 0 & 4 & 2 \\ 3 & 4 & 9 \\ 5 & -1 & 0 \end{bmatrix}$ is a 3×3 matrix.
- [x] $\begin{bmatrix} 1 & 2 \end{bmatrix}$ is a 1×2 matrix.

Next, let's talk about how to refer to specific elements of the matrix. And by matrix elements, other than the matrix I just mean the entries, so the numbers inside the matrix. So, in the standard notation, if A is this matrix here, then A sub-strip IJ is going to refer to the i, j entry, meaning the entry in the matrix in the ith row and jth column. So for example a1-1 is going to refer to the entry in the 1st row and the 1st column, so that's the first row and the first column and so a1-1 is going to be equal to 1, 4, 0, 2. Another example, 8 1 2 is going to refer to the entry in the first row and the second column and so A 1 2 is going to be equal to one nine one. This come from a quick examples. Let's see, A, oh let's say A 3 2, is going to refer to the entry in the 3rd row, and second column, right, because that's 3 2 so that's equal to 1 4 3 7. And finally, 8 4 1 is going to refer to this one right, fourth row, first column is equal to 1 4 7 and if, hopefully you won't, but if you were to write and say well this A 4 3, well, that refers to the fourth row, and the third column that, you know, this matrix has no third column so this is undefined, you know, or you can think of this as an error. There's no such element as 8 4 3, so, you know, you shouldn't be referring to 8 4 3. So, the matrix gets you a way of letting you quickly organise, index and access lots of data. In case I seem to be tossing up a lot of concepts, a lot of new notations very rapidly, you don't need to memorise all of this, but on the course website where we have posted the lecture notes, we also have all of these definitions written down. So you can always refer back, you know, either to these slides, possible coursework, so audible lecture notes if you forget well, A41 was that? Which row, which column was that? Don't worry about memorising everything now. You can always refer back to the written materials on the course website, and use that as a reference. So that's what a matrix is.



Question

Let A be a matrix shown below. A_{32} is one of the elements of this matrix.

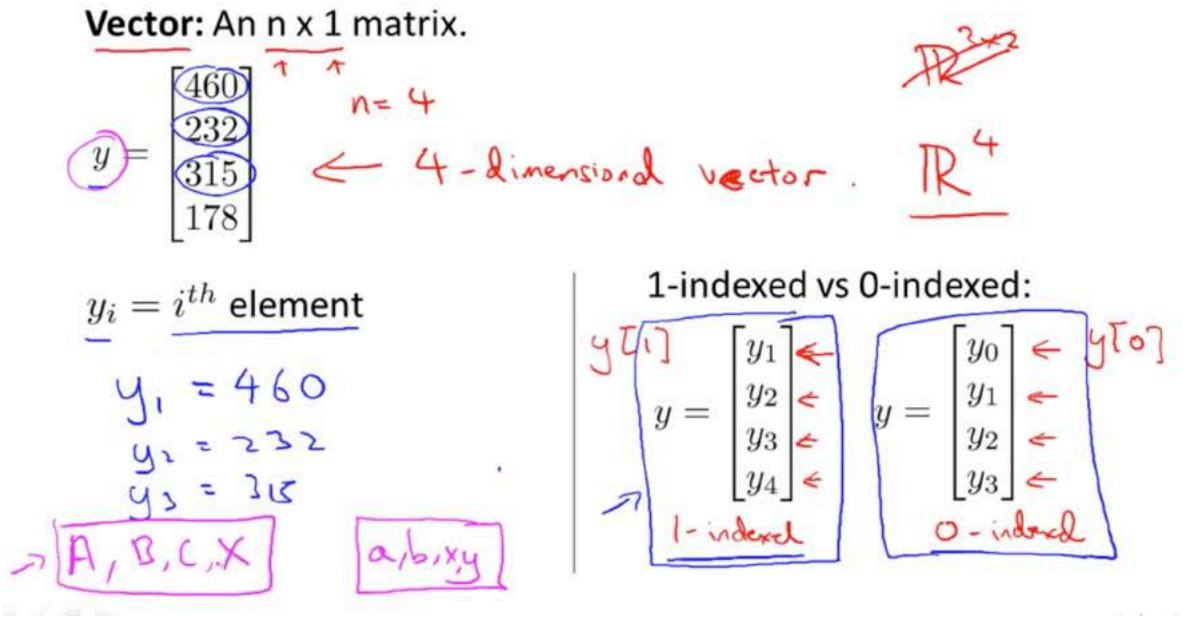
$$A = \begin{bmatrix} 85 & 76 & 66 & 5 \\ 94 & 75 & 18 & 28 \\ 68 & 40 & 71 & 5 \end{bmatrix}$$

What is the value of A_{32}?

- 18
- 28
- 76
- 40 (selected)

Next, let's talk about what is a vector. A vector turns out to be a special case of a matrix. A vector is a matrix that has only 1 column so you have an N x 1 matrix, then that's a remember, right? N is the number of rows, and 1 here is

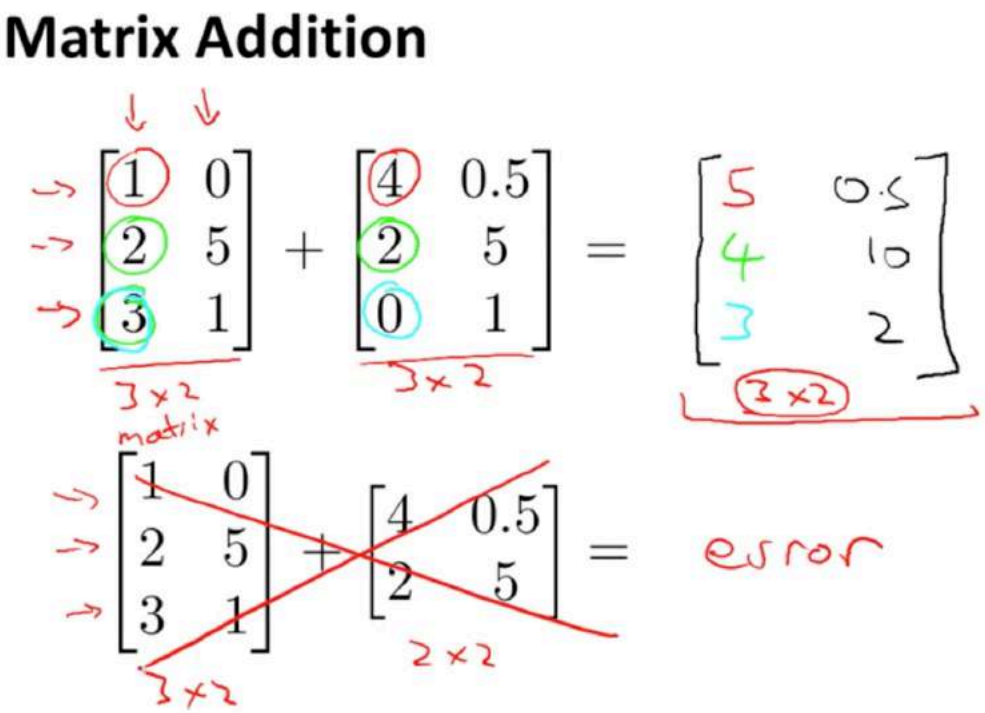
the number of columns, so, so matrix with just one column is what we call a vector. So here's an example of a vector, with I guess I have N equals four elements here. so we also call this thing, another term for this is a four dimensional vector, just means that this is a vector with four elements, with four numbers in it. And, just as earlier for matrices you saw this notation R3 by 2 to refer to 2 by 3 matrices, for this vector we are going to refer to this as a vector in the set R4. So this R4 means a set of four-dimensional vectors. Next let's talk about how to refer to the elements of the vector. We are going to use the notation yi to refer to the ith element of the vector y. So if y is this vector, y subscript i is the ith element. So y1 is the first element, four sixty, y2 is equal to the second element, two thirty two -there's the first. There's the second. Y3 is equal to 315 and so on, and only y1 through y4 are defined consistency 4-dimensional vector. Also it turns out that there are actually 2 conventions for how to index into a vector and here they are. Sometimes, people will use one index and sometimes zero index factors. So this example on the left is a one in that vector where the element we write is y1, y2, y3, y4. And this example in the right is an example of a zero index factor where we start the indexing of the elements from zero. So the elements go from a zero up to y three. And this is a bit like the arrays of some primary languages where the arrays can either be indexed starting from one. The first element of an array is sometimes a Y1, this is sequence notation I guess, and sometimes it's zero index depending on what programming language you use. So it turns out that in most of math, the one index version is more common For a lot of machine learning applications, zero index vectors gives us a more convenient notation. So what you should usually do is, unless otherwise specified, you should assume we are using one index vectors. In fact, throughout the rest of these videos on linear algebra review, I will be using one index vectors. But just be aware that when we are talking about machine learning applications, sometimes I will explicitly say when we need to switch to, when we need to use the zero index vectors as well. Finally, by convention, usually when writing matrices and vectors, most people will use upper case to refer to matrices. So we're going to use capital letters like A, B, C, you know, X, to refer to matrices, and usually we'll use lowercase, like a, b, x, y, to refer to either numbers, or just raw numbers or scalars or to vectors. This isn't always true but this is the more common notation where we use lower case "Y" for referring to vector and we usually use upper case to refer to a matrix. So, you now know what are matrices and vectors. Next, we'll talk about some of the things you can do with them



Addition and Scalar Multiplication

In this video we'll talk about matrix addition and subtraction, as well as how to multiply a matrix by a number, also called Scalar Multiplication. Let's start an example. Given two matrices like these, let's say I want to add them together. How do I do that? And so, what does addition of matrices mean? It turns out that if you want to add two matrices, what you do is you just add up the elements of these matrices one at a time. So, my result of adding two matrices is going to be itself another matrix and the first element again just by taking one and four and multiplying them and adding them together, so I get five. The second element I get by taking two and two and adding them, so I get four; three plus three plus zero is three, and so on. I'm going to stop changing colors, I guess. And, on the right is open five, ten and two. And it turns out you can add only two matrices that are of the same dimensions. So this example is a three by two matrix, because this has 3 rows and 2 columns, so it's 3 by 2. This is also a 3 by 2 matrix, and the result of adding these two matrices is a 3 by 2 matrix again. So you can only add matrices of the same dimension, and the result will be another matrix that's of the same dimension as the ones you just added. Where as in contrast, if you were to take these two matrices, so this one is a 3 by 2 matrix, okay, 3 rows, 2 columns. This here is a 2 by 2 matrix. And because these two matrices are not of the same dimension, you

know, this is an error, so you cannot add these two matrices and, you know, their sum is not well-defined. So that's matrix addition.



Question

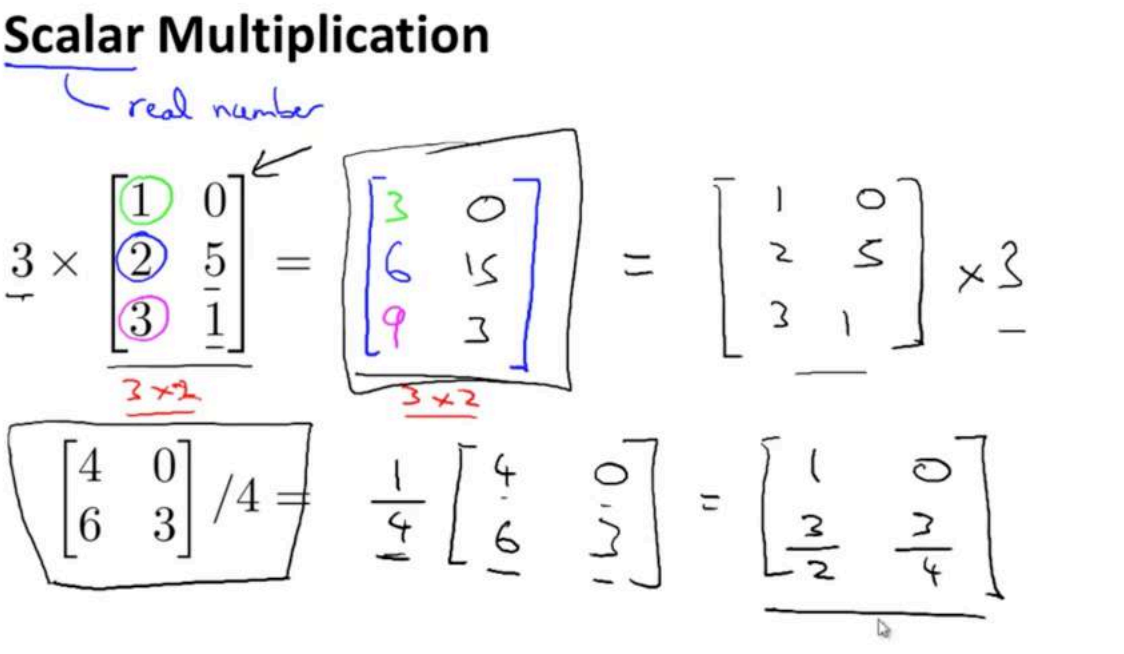
What is

$$\begin{bmatrix} 8 & 6 & 9 \\ 10 & 1 & 10 \end{bmatrix} + \begin{bmatrix} 3 & 10 & 2 \\ 6 & 1 & -1 \end{bmatrix}?$$

- $\begin{bmatrix} 5 & -4 & 7 \\ 4 & 0 & 11 \end{bmatrix}$
- $\begin{bmatrix} 11 & 16 & 11 \\ 16 & 2 & 9 \end{bmatrix}$
- $\begin{bmatrix} 14 & 7 & 8 \\ 13 & 11 & 12 \end{bmatrix}$
- $\begin{bmatrix} 8 & 6 & 9 \\ 10 & 1 & 10 \end{bmatrix}$

Next, let's talk about multiplying matrices by a scalar number. And the scalar is

just a, maybe a overly fancy term for, you know, a number or a real number. Alright, this means real number. So let's take the number 3 and multiply it by this matrix. And if you do that, the result is pretty much what you'll expect. You just take your elements of the matrix and multiply them by 3, one at a time. So, you know, one times three is three. What, two times three is six, 3 times 3 is 9, and let's see, I'm going to stop changing colors again. Zero times 3 is zero. Three times 5 is 15, and 3 times 1 is three. And so this matrix is the result of multiplying that matrix on the left by 3. And you notice, again, this is a 3 by 2 matrix and the result is a matrix of the same dimension. This is a 3 by 2, both of these are 3 by 2 dimensional matrices. And by the way, you can write multiplication, you know, either way. So, I have three times this matrix. I could also have written this matrix and 0, 2, 5, 3, 1, right. I just copied this matrix over to the right. I can also take this matrix and multiply this by three. So whether it's you know, 3 times the matrix or the matrix times three is the same thing and this thing here in the middle is the result. You can also take a matrix and divide it by a number. So, turns out taking this matrix and dividing it by four, this is actually the same as taking the number one quarter, and multiplying it by this matrix. 4, 0, 6, 3 and so, you can figure the answer, the result of this product is, one quarter times four is one, one quarter times zero is zero. One quarter times six is, what, three halves, about six over four is three halves, and one quarter times three is three quarters. And so that's the results of computing this matrix divided by four. Vectors give you the result.

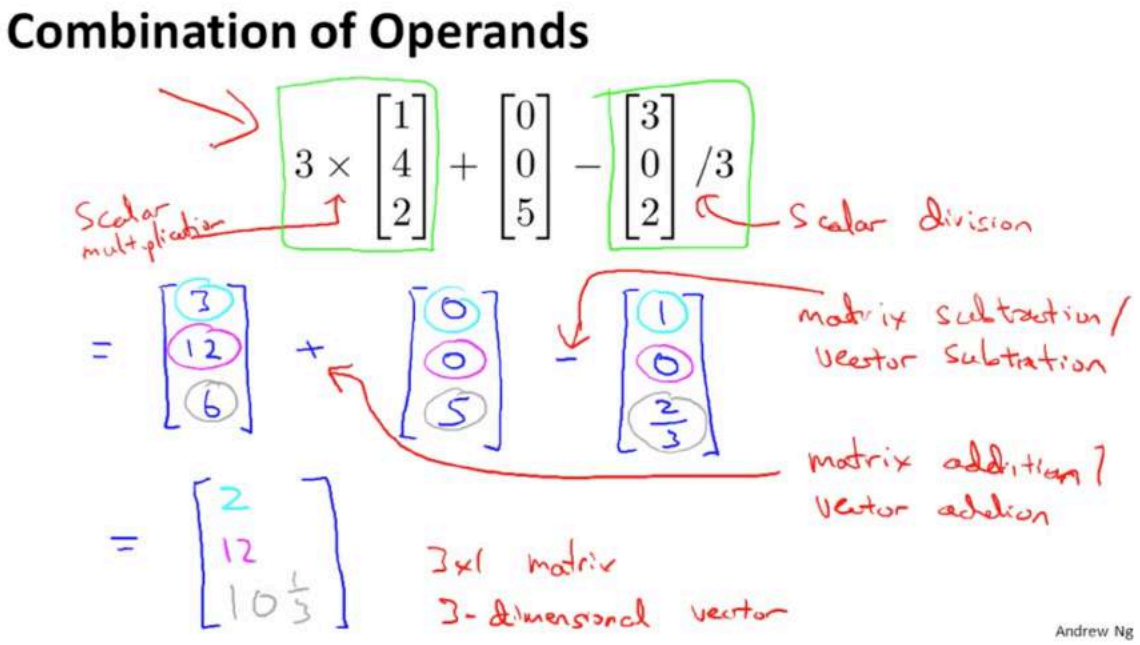


Question

What is $2 \times \begin{bmatrix} 4 & 5 \\ 1 & 7 \end{bmatrix}$?

- (●) $\begin{bmatrix} 8 & 10 \\ 2 & 14 \end{bmatrix}$
- ( ) $\begin{bmatrix} 8 & 5 \\ 1 & 7 \end{bmatrix}$
- ( ) $\begin{bmatrix} 8 & 10 \\ 1 & 7 \end{bmatrix}$
- ( ) $\begin{bmatrix} 4 & 5 \\ 1 & 14 \end{bmatrix}$

Finally, for a slightly more complicated example, you can also take these operations and combine them together. So in this calculation, I have three times a vector plus a vector minus another vector divided by three. So just make sure we know where these are, right. This multiplication. This is an example of scalar multiplication because I am taking three and multiplying it. And this is, you know, another scalar multiplication. Or more like scalar division, I guess. It really just means one zero times this. And so if we evaluate these two operations first, then what we get is this thing is equal to, let's see, so three times that vector is three, twelve, six, plus my vector in the middle which is a 005 minus one, zero, two-thirds, right? And again, just to make sure we understand what is going on here, this plus symbol, that is matrix addition, right? I really, since these are vectors, remember, vectors are special cases of matrices, right? This, you can also call this vector addition This minus sign here, this is again a matrix subtraction, but because this is an n by 1, really a three by one matrix, that this is actually a vector, so this is also vector, this column. We call this matrix a vector subtraction, as well. OK? And finally to wrap this up. This therefore gives me a vector, whose first element is going to be 3+0-1, so that's 3-1, which is 2. The second element is 12+0-0, which is 12. And the third element of this is, what, 6+5-(2/3), which is 11-(2/3), so that's 10 and one-third and see, you close this square bracket. And so this gives me a 3 by 1 matrix, which is also just called a 3 dimensional vector, which is the outcome of this calculation over here.



Question

What is $\begin{bmatrix} 4 \\ 6 \\ 7 \end{bmatrix} /2 - 3 \begin{bmatrix} 2 \\ 1 \\ 0 \end{bmatrix}$?

- $\begin{bmatrix} 4 \\ 0 \\ 5 \end{bmatrix}$
- $\begin{bmatrix} -4 \\ -1 \\ 3 \end{bmatrix}$
- $\begin{bmatrix} -4 \\ 0 \\ 3.5 \end{bmatrix}$
- $\begin{bmatrix} 0 \\ 2 \\ 3.5 \end{bmatrix}$

So far I have only talked about how to multiply matrices and vectors by scalars, by row numbers. In the next video we will talk about a much more interesting step, of taking 2 matrices and multiplying 2 matrices together.