. Machine learning is one of the most exciting recent technologies.

Every time you use a web search engine like Google or Bing to search the internet, one of the reasons that works so well is because a learning algorithm, one implemented by Google or Microsoft, has learned how to rank web pages. Every time you use Facebook or Apple's photo typing application and it recognizes your friends' photos, that's also machine learning. Every time you read your email and your spam filter saves you from having to wade through tons of spam email, that's also a learning algorithm.

For me one of the reasons I'm excited is the AI dream of someday building machines as intelligent as you or me. We're a long way away from that goal, but many AI researchers believe that the best way to towards that goal is through learning algorithms that try to mimic how the human brain learns. I'll tell you a little bit about that too in this class. In this class you learn about state-of-the-art machine learning algorithms. But it turns out just knowing the algorithms and knowing the math isn't that much good if you don't also know how to actually get this stuff to work on problems that you care about. So, we've also spent a lot of time developing exercises for you to implement each of these algorithms and see how they work fot yourself. So why is machine learning so prevalent today? It turns out that machine learning is a field that had grown out of the field of AI, or artificial intelligence. We wanted to build intelligent machines and it turns out that there are a few basic things that we could program a machine to do such as how to find the shortest path from A to B. But for the most part we just did not know how to write AI programs to do the more interesting things such as web search or photo tagging or email anti-spam. There was a realization that the only way to do these things was to have a machine learn to do it by itself. So, machine learning was developed as a new capability for computers and today it touches many segments of industry and basic science.

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. [Helicopter whirling]

Handwriting recognition. It turns out one of the reasons it's so inexpensive today to route a piece of mail across the countries, in the US and internationally, is that when you write an envelope like this, it turns out there's a learning algorithm that has learned how to read your handwriting so that it can automatically route this envelope on its way, and so it costs us a few cents to send this thing thousands of miles. And in fact if you've seen the fields of natural language processing or computer vision, these are the fields of AI pertaining to understanding language or understanding images. Most of natural language processing and most of computer vision today is applied machine learning. Learning algorithms are also widely used for self- customizing programs. 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.

We'll talk about how researches are using this to make progress towards the big AI dream. A few months ago, a student showed me an article on the top twelve IT skills. The skills that information technology hiring managers cannot say no to. It was a slightly older article, but at the top of this list of the twelve most desirable IT skills was machine learning.

What is machine learning? Even among machine learning practitioners there isn't a well-accepted definition of what is and what isn't machine learning. But let me show you a couple of examples of the ways that people have tried to define it. Here's the definition of what is machine learning does to Arthur Samuel. He defined machine learning as the field of study that gives computers the ability to learn without being explicitly programmed. Samuel's claim to fame was that back in the 1950's, he wrote a checkers playing program. And the amazing thing about this checkers playing program, was that Arthur Samuel himself, wasn't a very good checkers player. But what he did was, he had to program for it to play 10's of 1000's of games against itself. And by watching what sorts of board positions tended to lead to wins, and what sort of board positions tended to lead to losses. The checkers playing program learns over time what are good board positions and what are bad board positions. And eventually learn to play checkers better than Arthur Samuel himself was able to. This was a remarkable result. Although Samuel himself turned out not to be a very good checkers player. But because the computer has the patience to play tens of thousands of games itself. No human, has the patience to play that many games. By doing this the computer was able to get so much checkers-playing experience that it eventually became a better checkers player than Arthur Samuel himself. This is somewhat informal definition, and an older one. Here's a slightly more recent definition by Tom Mitchell, who's a friend out of Carnegie Mellon. So Tom defines machine learning by saying that, a well posed learning problem is defined as follows.

He says, 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. I actually think he came up with this definition just to make it rhyme. For the checkers playing example the experience e, will be the experience of having the program play 10's of 1000's of games against itself. The task t, will be the task of playing checkers. And the performance measure p, will be the probability that it wins the next game of checkers against some new opponent.

. Let's say your email program watches which emails you do or do not flag as spam. So in an email client like this you might click this spam button to report some email as spam, but not other emails and. Based on which emails you mark as spam, so your e-mail program learns better how to filter spam e-mail. What is the task T in this setting? In a few seconds, the video will pause. And when it does so, you can use your mouse to select one of these four radio buttons to let, to let me know which of these four you think is the right answer to this question. That might be a performance measure P. And so, our task performance on the task our system's performance on the task T, on the performance measure P will improve after the experience E.

There are several different types of learning algorithms. The main two types are what we call supervised learning and unsupervised learning. But it turns out that in supervised learning, the idea is that we're going to teach the computer how to do something, whereas in unsupervised learning we're going let it learn by itself. You will also hear other buzz terms such as reinforcement learning and recommender systems. These are other types of machine learning algorithms that we'll talk about later but the two most used types of learning algorithms are probably supervised learning and unsupervised learning.

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.

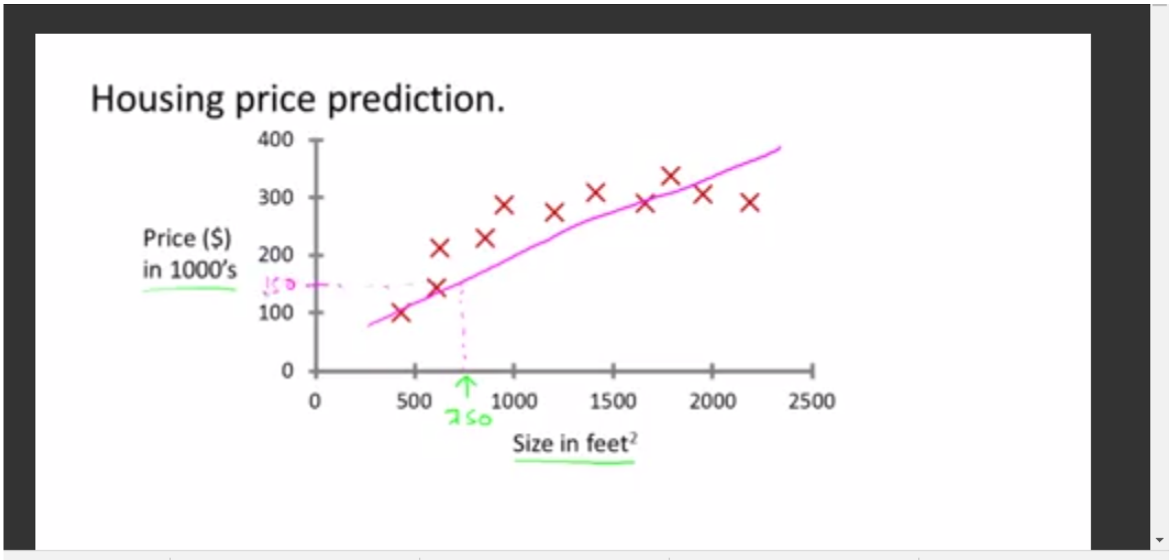
E = the experience of playing many games of checkers

T = the task of playing checkers.

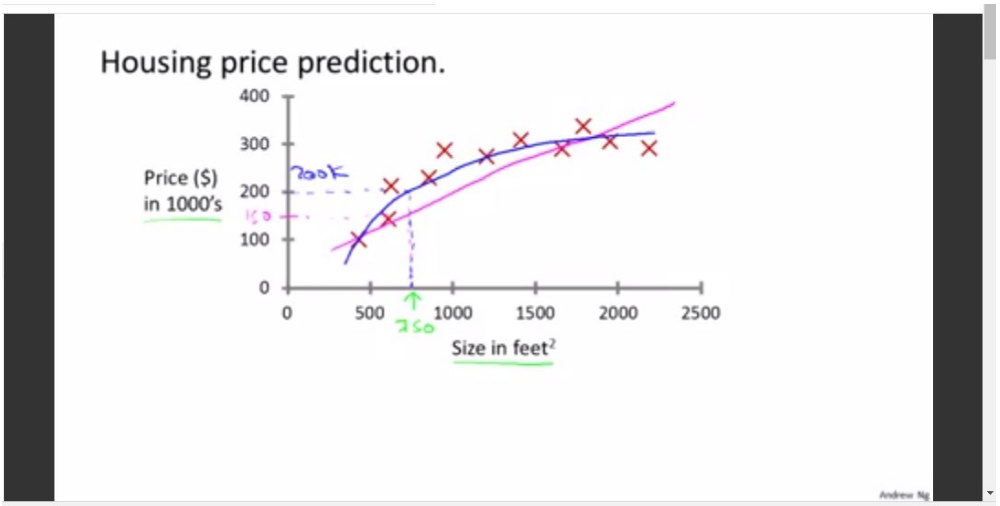
P = the probability that the program will win the next game.

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

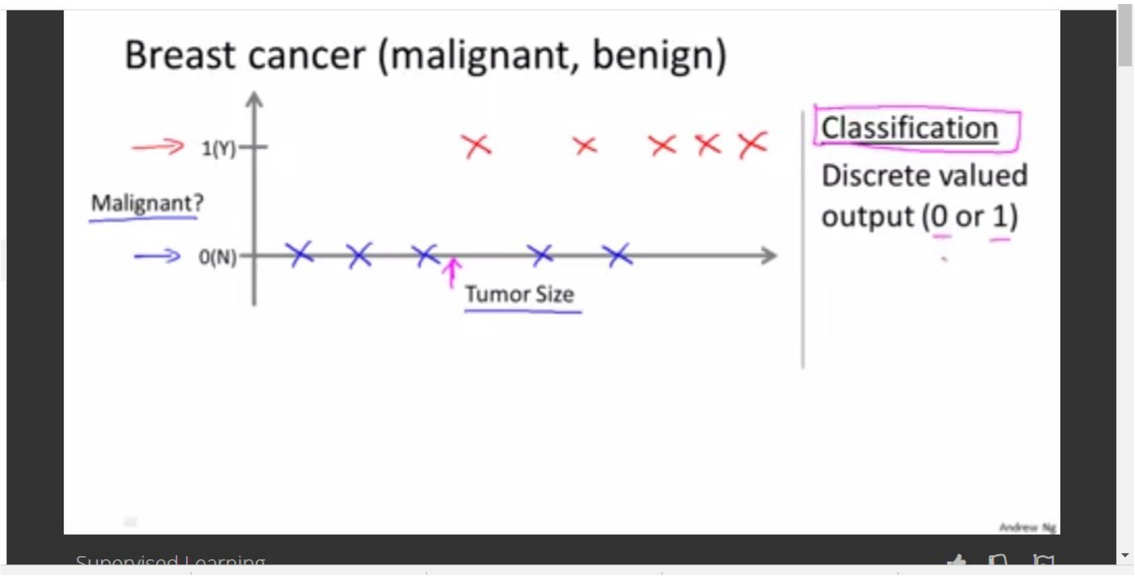
Supervised learning and Unsupervised learning.



Housing price prediction.



Housing price prediction.



Breast cancer (malignant, benign)

Classification

Discrete valued output (0 or 1)

Supervised Learning

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:

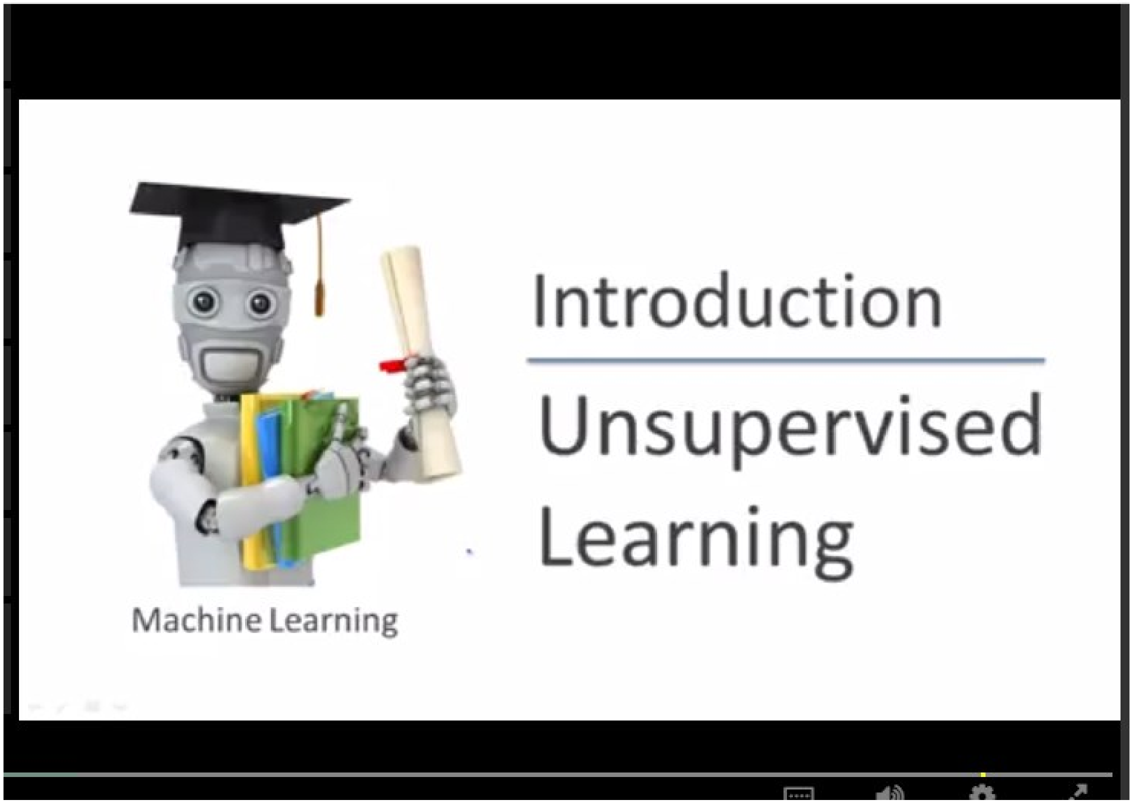
Given data about the size of houses on the real estate market, try to predict their price. Price as a function of size is a continuous output, so this is a regression problem.

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.



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.

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 groups of people that all know each other are?

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. 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.

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.

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.

Unsupervised Learning

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.

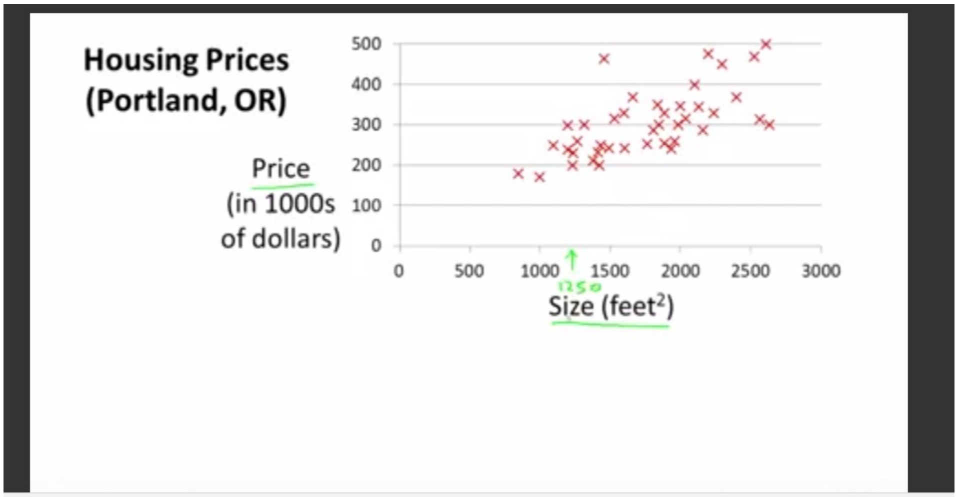
Example:

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.

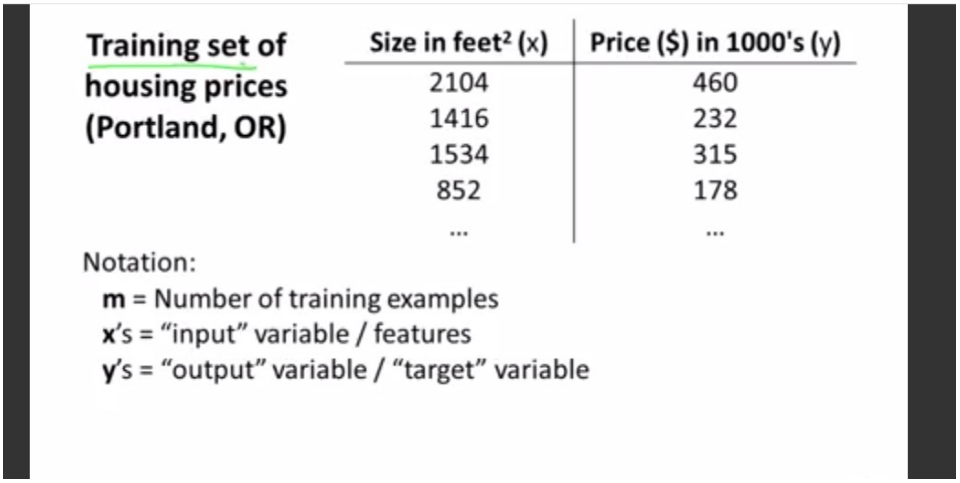
Non-clustering: The "Cocktail Party Algorithm", allows you to find structure in a chaotic environment. (i.e. identifying individual voices and music from a mesh of sounds at a cocktail party).

Model Representation:

Our first learning algorithm will be linear regression. 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 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. So as mentioned, occasionally I'll ask you a question to let you check your understanding and a few seconds in this video a multiple-choice question will pop up in the video. When it does, please use your mouse to select what you think is the right answer. What defined by the training set is. 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<u>theta(x) equals theta<u>0</u></u> plus theta<u>1 of x. And as a shorthand, sometimes instead of writing, you</u> 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.



Housing Prices (Portland, OR)

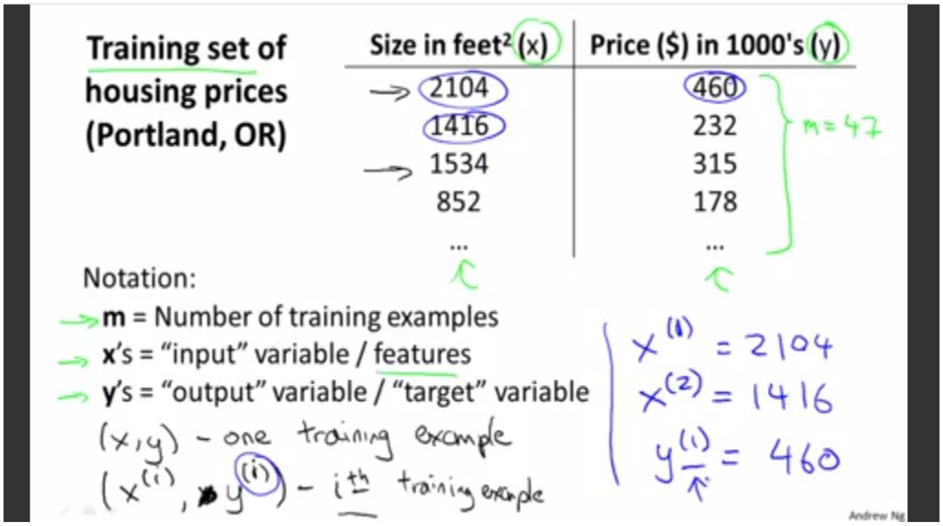


Training set of housing prices (Portland, OR)

Size in feet² (x)	Price ($) in 1000's (y)
2104	460
1416	232
1534	315
852	178
…	…

Notation:

m = Number of training examples
x's = "input" variable / features
y's = "output" variable / "target" variable



Training set of housing prices (Portland, OR)

Size in feet² (x)	Price ($) in 1000's (y)
2104	460
1416	232
1534	315
852	178
…	…

m = 47

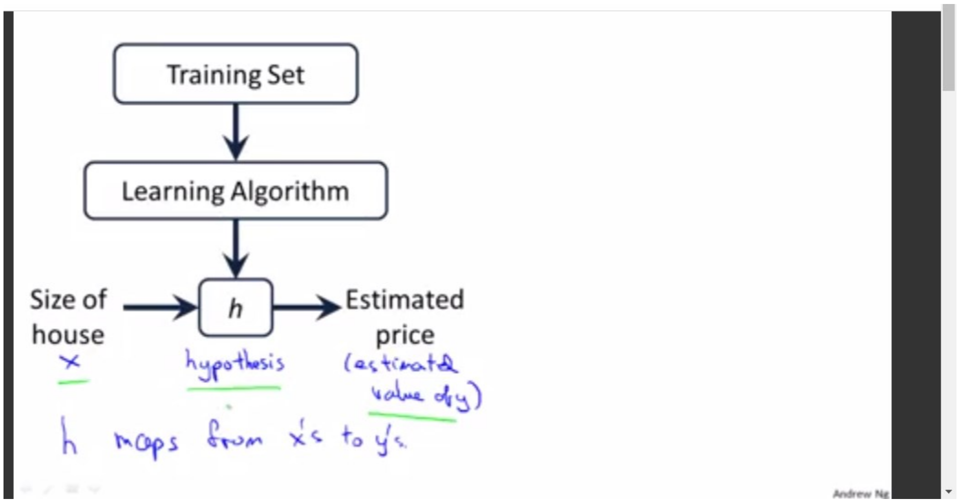
Notation:

m = Number of training examples
x's = "input" variable / features
y's = "output" variable / "target" variable

(x, y) - one training example
$(x^{(i)}, y^{(i)})$ - i^{th} training example

$x^{(1)} = 2104$
$x^{(2)} = 1416$
$y^{(1)} = 460$

Andrew Ng



Training Set → Learning Algorithm → h

Size of house → h → Estimated price

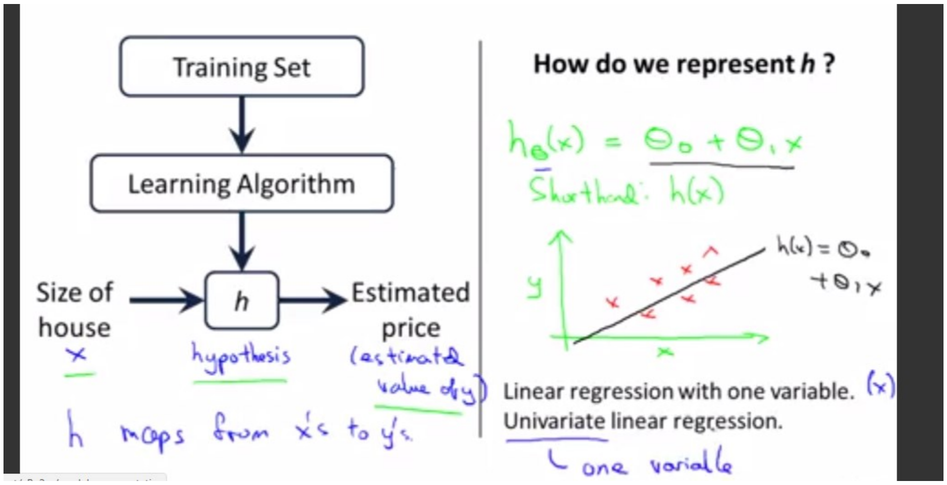
x

hypothesis

(estimated value of y)

h maps from x's to y's.

Andrew Ng



Training Set

Learning Algorithm

Size of house → h → Estimated price

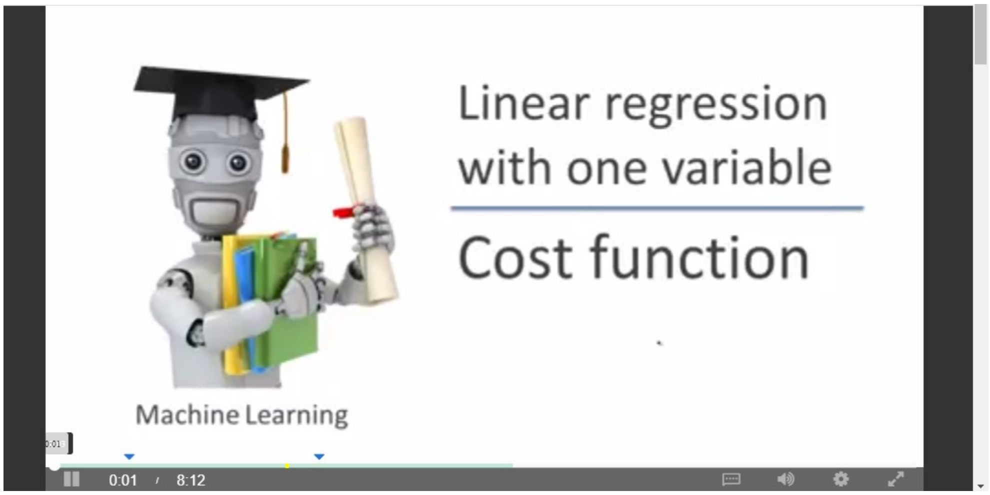
How do we represent *h* ?

$h_\theta(x) = \theta_0 + \theta_1 x$

Shorthand: $h(x)$

Linear regression with one variable. (x)

Univariate linear regression.



Linear regression with one variable

Cost function

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. 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 phat 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.

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 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.

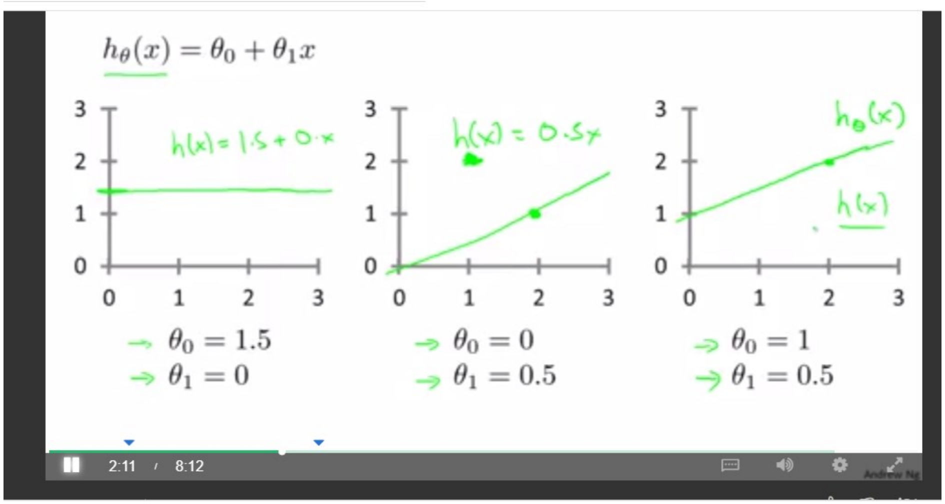
And just to rewrite this out a little bit more cleanly, what I'm going to do is, by convention we usually define a cost function, which is going to be exactly this, that formula I have up here.

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 erros. 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.

7:42

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...

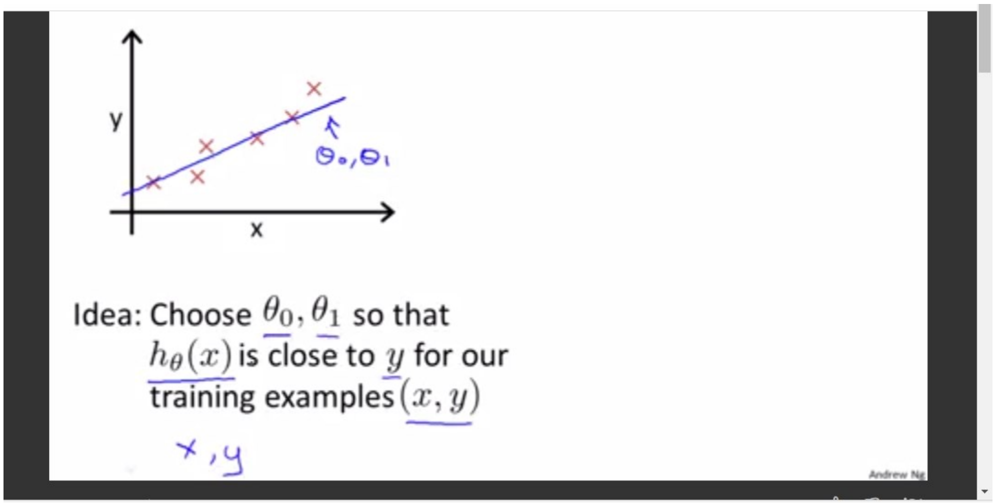


$h_\theta(x) = \theta_0 + \theta_1 x$

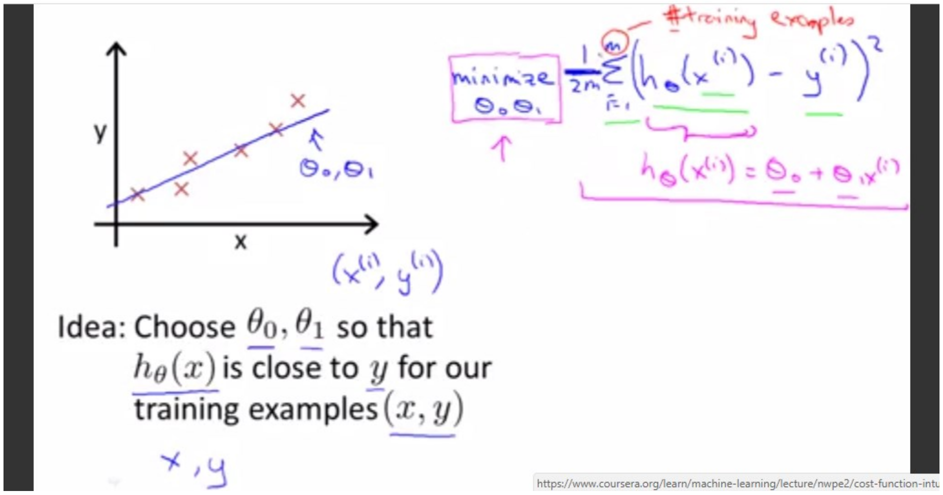
$\theta_0 = 1.5$
$\theta_1 = 0$

$\theta_0 = 0$
$\theta_1 = 0.5$

$\theta_0 = 1$
$\theta_1 = 0.5$



Idea: Choose θ_0, θ_1 so that
$h_\theta(x)$ is close to y for our
training examples (x, y)



$$\boxed{\underset{\Theta_0 \Theta_1}{\text{minimize}}} \frac{1}{2m}\sum_{i=1}^{m}\left(h_\Theta(x^{(i)}) - y^{(i)}\right)^2$$

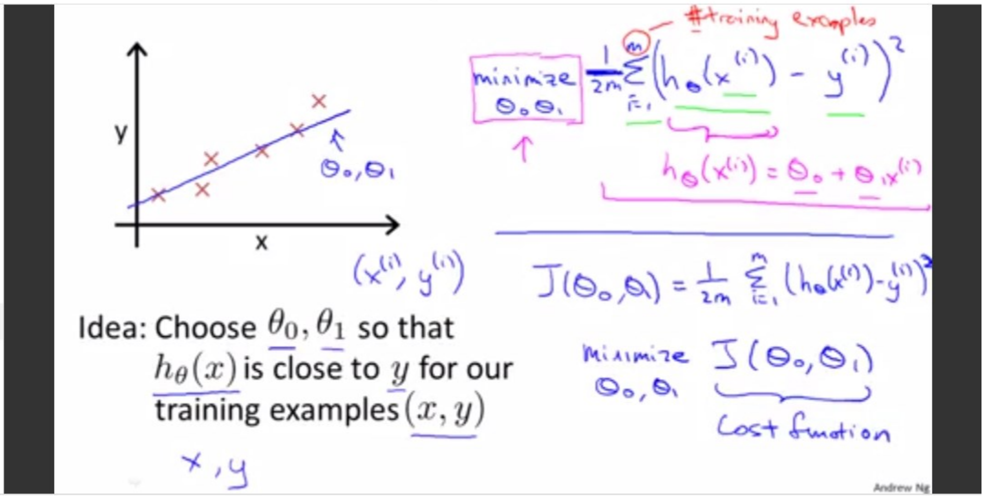
#training examples

$$h_\Theta(x^{(i)}) = \Theta_0 + \Theta_1 x^{(i)}$$

Idea: Choose θ_0, θ_1 so that $h_\theta(x)$ is close to y for our training examples (x, y)

x, y

https://www.coursera.org/learn/machine-learning/lecture/nwpe2/cost-function-intu



$$\boxed{\underset{\Theta_0 \Theta_1}{\text{minimize}}} \frac{1}{2m}\sum_{i=1}^{m}\left(h_\Theta(x^{(i)}) - y^{(i)}\right)^2$$

#training examples

$$h_\Theta(x^{(i)}) = \Theta_0 + \Theta_1 x^{(i)}$$

$$J(\Theta_0, \Theta_1) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\Theta(x^{(i)}) - y^{(i)}\right)^2$$

$$\underset{\Theta_0, \Theta_1}{\text{minimize}} \underbrace{J(\Theta_0, \Theta_1)}_{\text{Cost function}}$$

Idea: Choose θ_0, θ_1 so that $h_\theta(x)$ is close to y for our training examples (x, y)

x, y

Andrew Ng

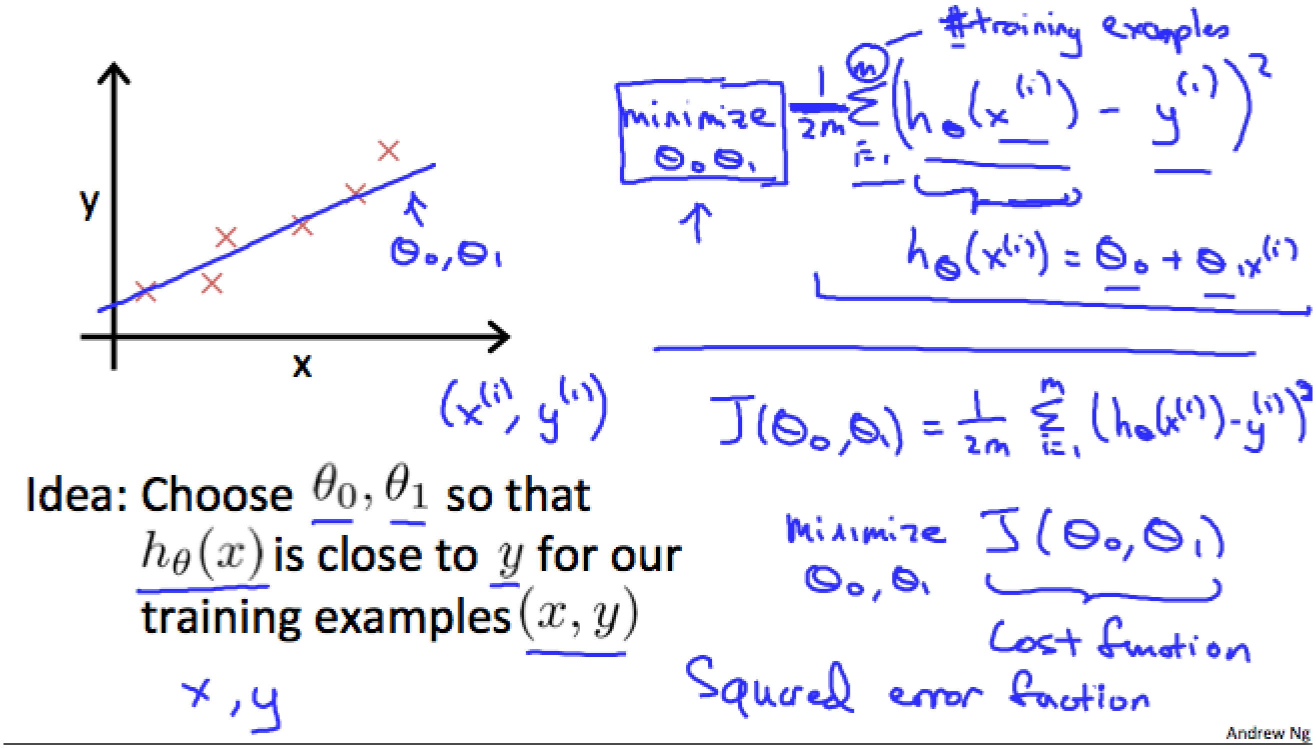
Cost Function

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.

$J(\theta_0,\theta_1)=12m\sum_{i=1m}(y\hat{}_i-y_i)2=12m\sum_{i=1m}(h_\theta(x_i)-y_i)2$

To break it apart, it is $12\,\bar{x}$ where $\bar{x}$ is the mean of the squares of $h_\theta(x_i)-y_i$, 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:



Idea: Choose θ_0, θ_1 so that $h_\theta(x)$ is close to y for our training examples (x, y)

y
x
θ_0, θ_1
$(x^{(i)}, y^{(i)})$
x, y
#training examples
$\boxed{\underset{\theta_0,\theta_1}{\text{minimize}}}\ \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$
$h_\theta(x^{(i)}) = \theta_0 + \theta_1 x^{(i)}$
$J(\theta_0,\theta_1) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)})-y^{(i)}\right)^2$
$\underset{\theta_0,\theta_1}{\text{minimize}}\ J(\theta_0,\theta_1)$
Cost function
Squared error function
Andrew Ng