## 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
* Unsupervised learning.

## 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

## 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).

## 

## Linear Regression with One Variable

Linear regression predicts a real-valued output based on an input value. We discuss the application of linear regression to housing price prediction, present the notion of a cost function, and introduce the gradient descent method for learning.

## Model Representation

## 

## 

## 

## 

## 

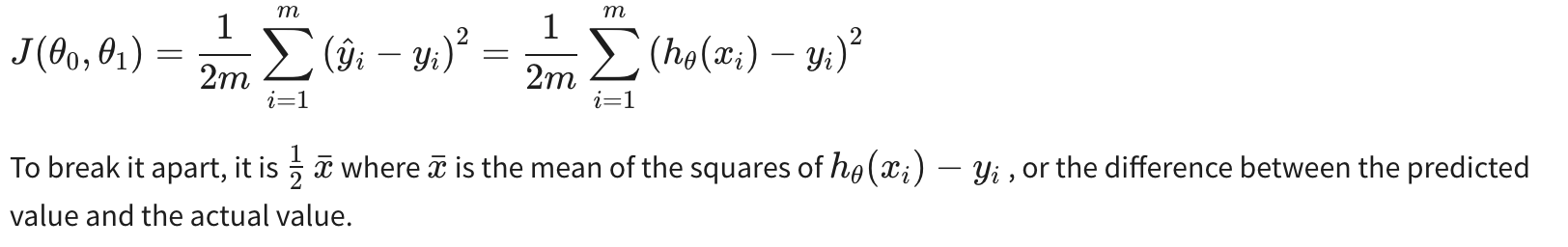
## 

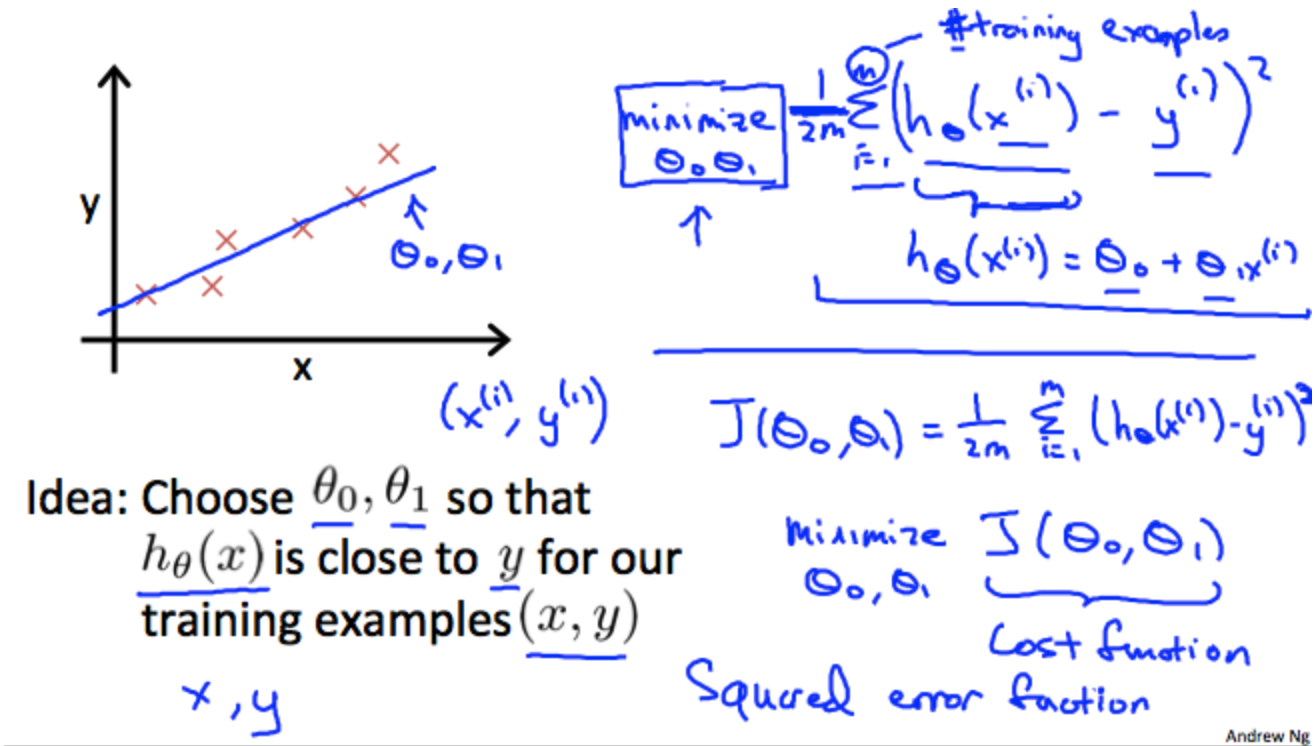
## 

## 

## 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.

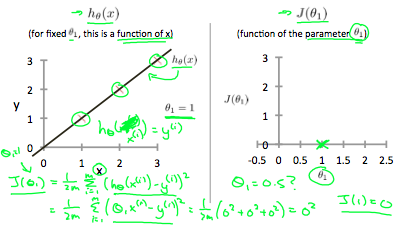
This function is otherwise called the "Squared error function", or "Mean squared error". The mean is halved (½) as a convenience for the computation of the gradient descent, as the derivative term of the square function will cancel out the ½ term. The following image summarizes what the cost function does:



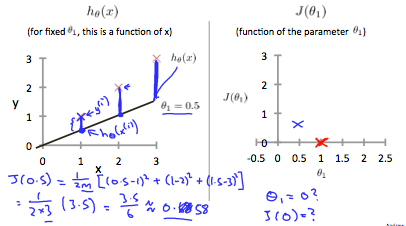
Cost Function - Intuition I

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.

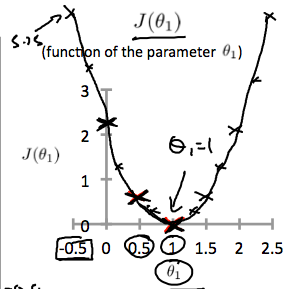
Our objective is to get the best possible line. The best possible line will be such so that the average squared vertical distances of the scattered points from the line will be the least. Ideally, the line should pass through all the points of our training data set. In such a case, the value of *J*(*θ*0,*θ*1) will be 0. The following example shows the ideal situation where we have a cost function of 0.



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



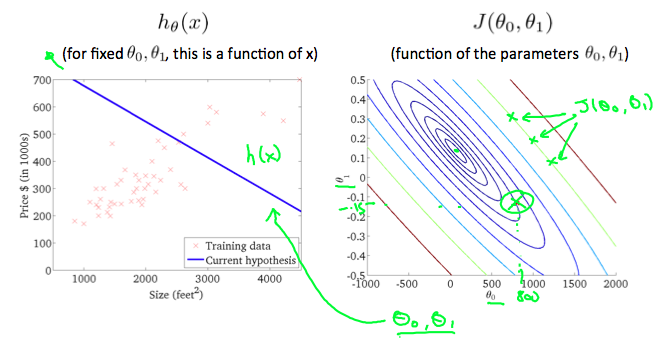
This increases our cost function to 0.58. Plotting several other points yields to the following graph:



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

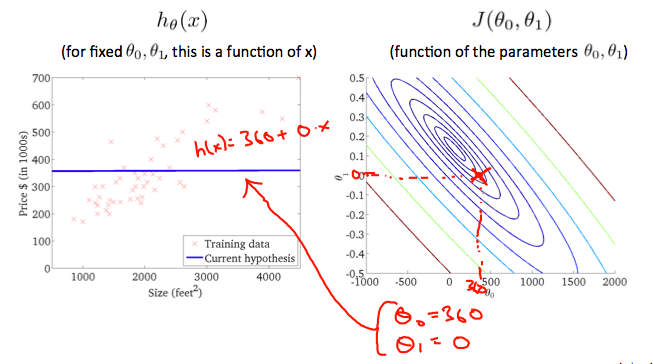
Cost Function - Intuition II

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.

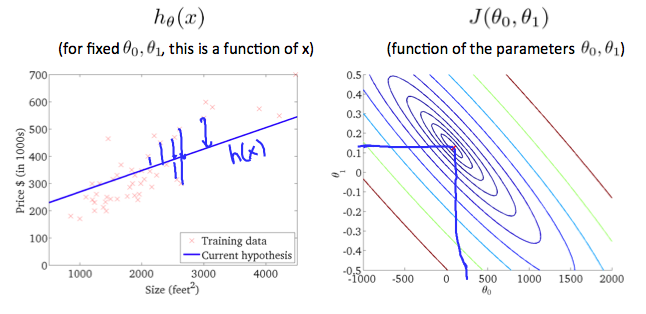


Taking any color and going along the 'circle', one would expect to get the same value of the cost function. For example, the three green points found on the green line above have the same value for*J*(*θ*0,*θ*1) and as a result, they are found along the same line. The circled x displays the value of the cost function for the graph on the left when

*θ*0= 800 and*θ*1= -0.15. Taking another h(x) and plotting its contour plot, one gets the following graphs:



When*θ*0= 360 and*θ*1= 0, the value of*J*(*θ*0,*θ*1) in the contour plot gets closer to the center thus reducing the cost function error. Now giving our hypothesis function a slightly positive slope results in a better fit of the data.



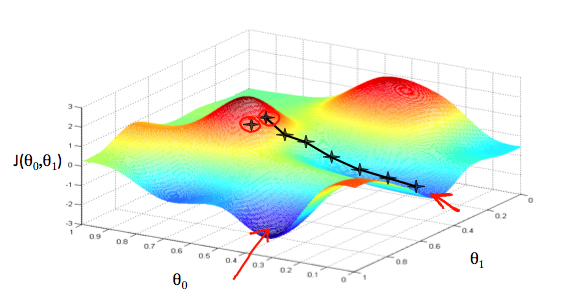
The graph above minimizes the cost function as much as possible and consequently, the result of *θ*1 and *θ*0 tend to be around 0.12 and 250 respectively. Plotting those values on our graph to the right seems to put our point in the center of the innermost 'circle'.

## Gradient Descent

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 *θ*0and*θ*1(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.

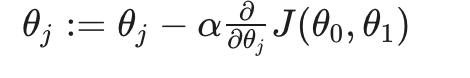
We put *θ*0 on the x axis and *θ*1 on the y axis, with the cost function on the vertical z axis. The points on our graph will be the result of the cost function using our hypothesis with those specific theta parameters. The graph below depicts such a setup.



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.

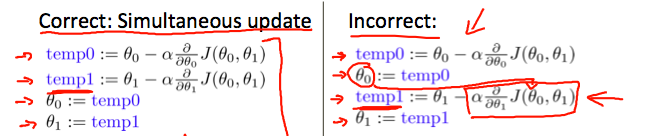
The way we do this is by taking the derivative (the tangential line to a function) of our cost function. The slope of the tangent is the derivative at that point and it will give us a direction to move towards. We make steps down the cost function in the direction with the steepest descent. The size 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*(*θ*0,*θ*1). 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:

**

where j=0,1 represents the feature index number.

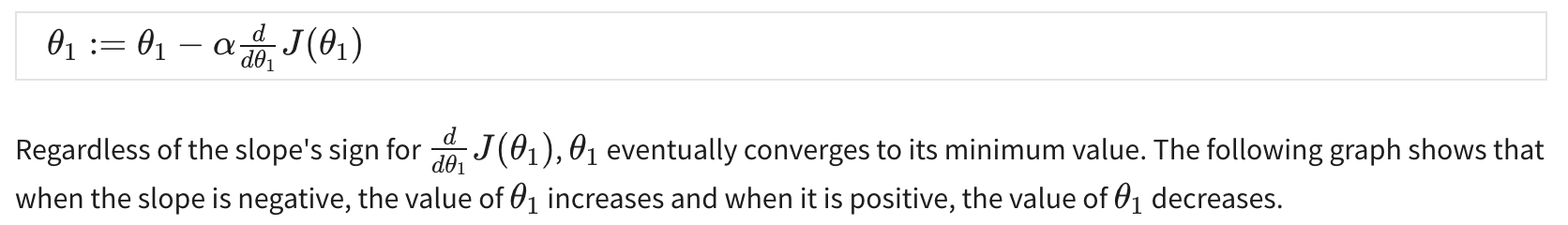
At each iteration j, one should simultaneously update the parameters*θ*1,*θ*2,...,*θn*. Updating a specific parameter prior to calculating another one on the*j*(*th*)iteration would yield a wrong implementation.

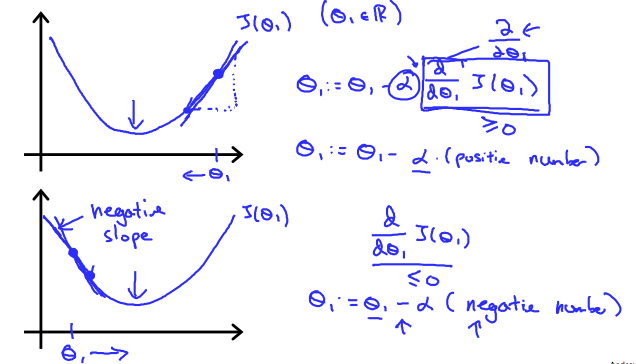


## Gradient Descent Intuition

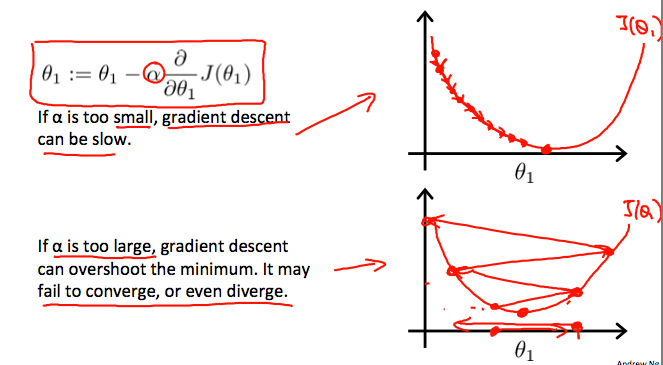
In this video we explored the scenario where we used one parameter*θ*1and plotted its cost function to implement a gradient descent. Our formula for a single parameter was :

Repeat until convergence:



.

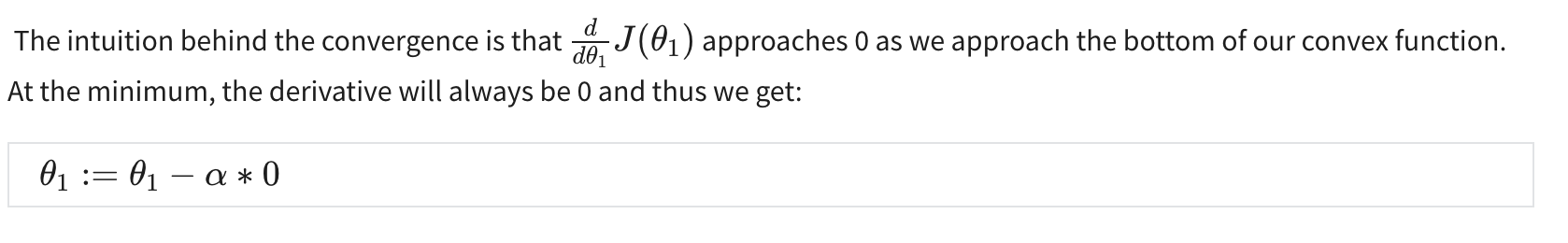
On a side note, we should adjust our parameter *α* to ensure that the gradient descent algorithm converges in a reasonable time. Failure to converge or too much time to obtain the minimum value imply that our step size is wrong.

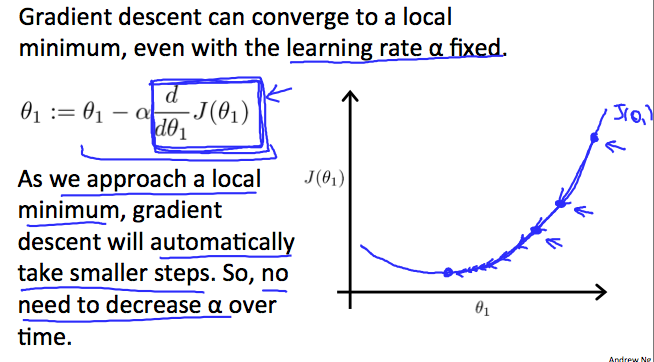


### 

### 

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





## Gradient Descent For Linear Regression

