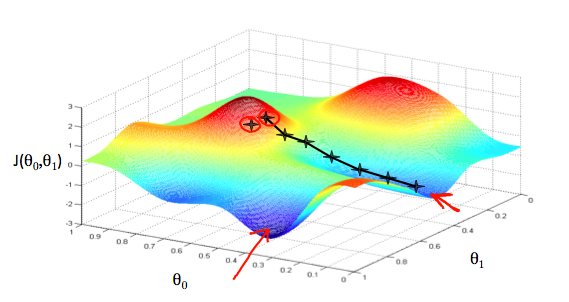
Gradient descent :

So we have our hypothesis function and we have a way of measuring how well it fits into the data. Gradient descent is used to estimate the parameters in the hypothesis function.

Imagine that we graph our hypothesis function based on its fields *θ0* and *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.

* In simple terms, we draw a graph between *θ0* , *θ*1 and J(*θ0* , *θ*1) where they lie on x, y, z (vertical axis) respectively.

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(local optimums).

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.

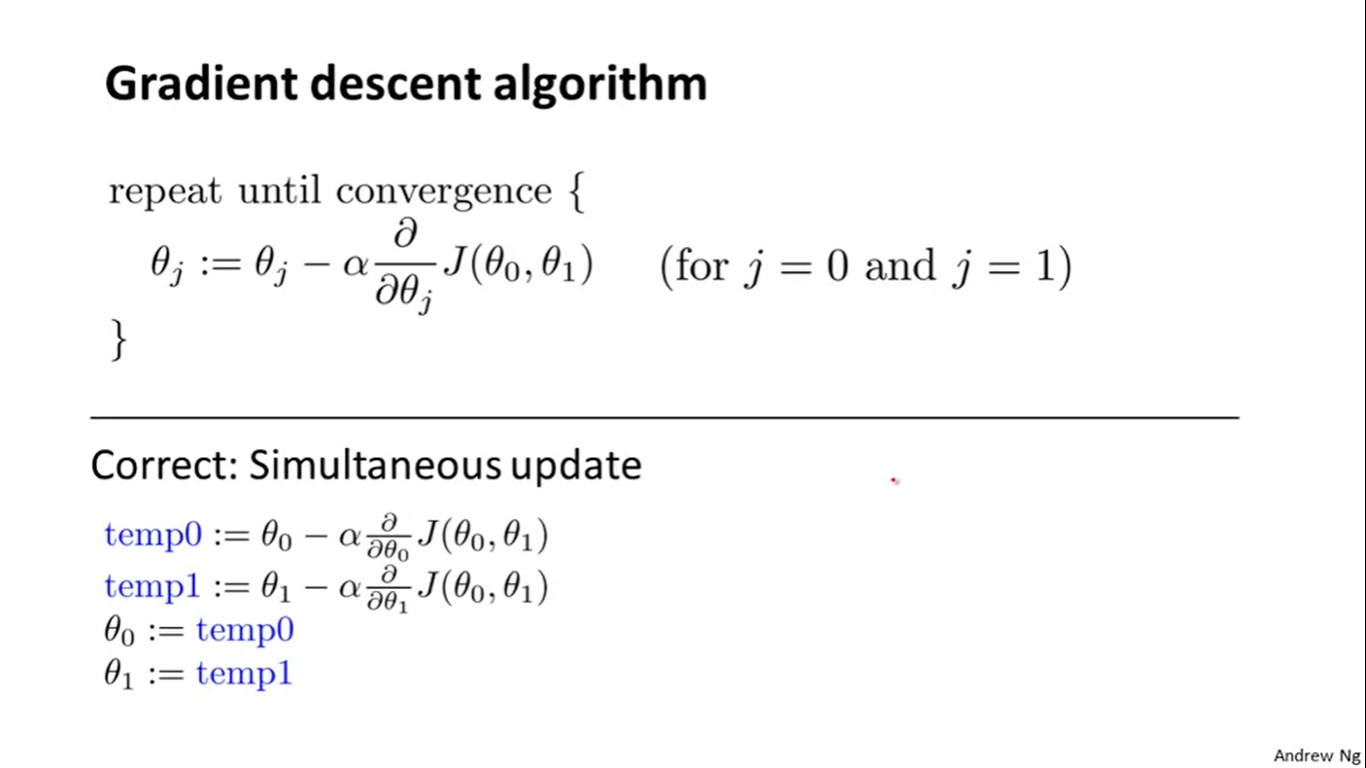
We make steps down the cost function in the direction with the steepest descent. The size of each step is determined by the parameter α, which is called the learning rate.



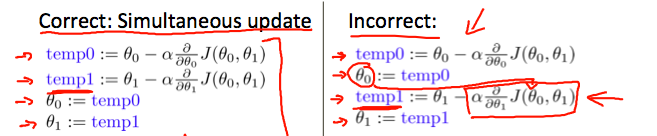
For example, the distance between each 'star' in the graph above represents a step determined by our parameter α. A smaller α would result in a smaller step and a larger α results in a larger step. The direction in which the step is taken is determined by the partial derivative of *J*(*θ*0,*θ*1).

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:

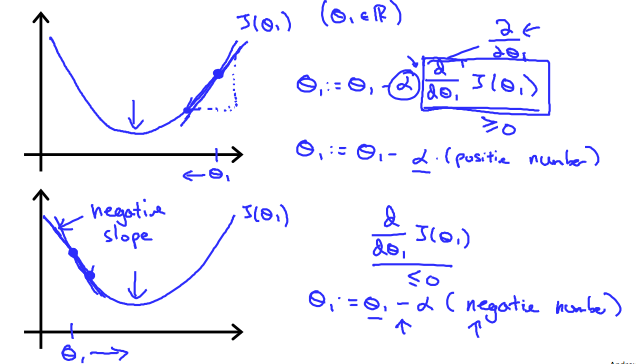


where, j=0,1 represents the feature index number. After each iteration *θ0 , θ1 . . . θn* gets updated. The incorrect updation order is also mentioned :

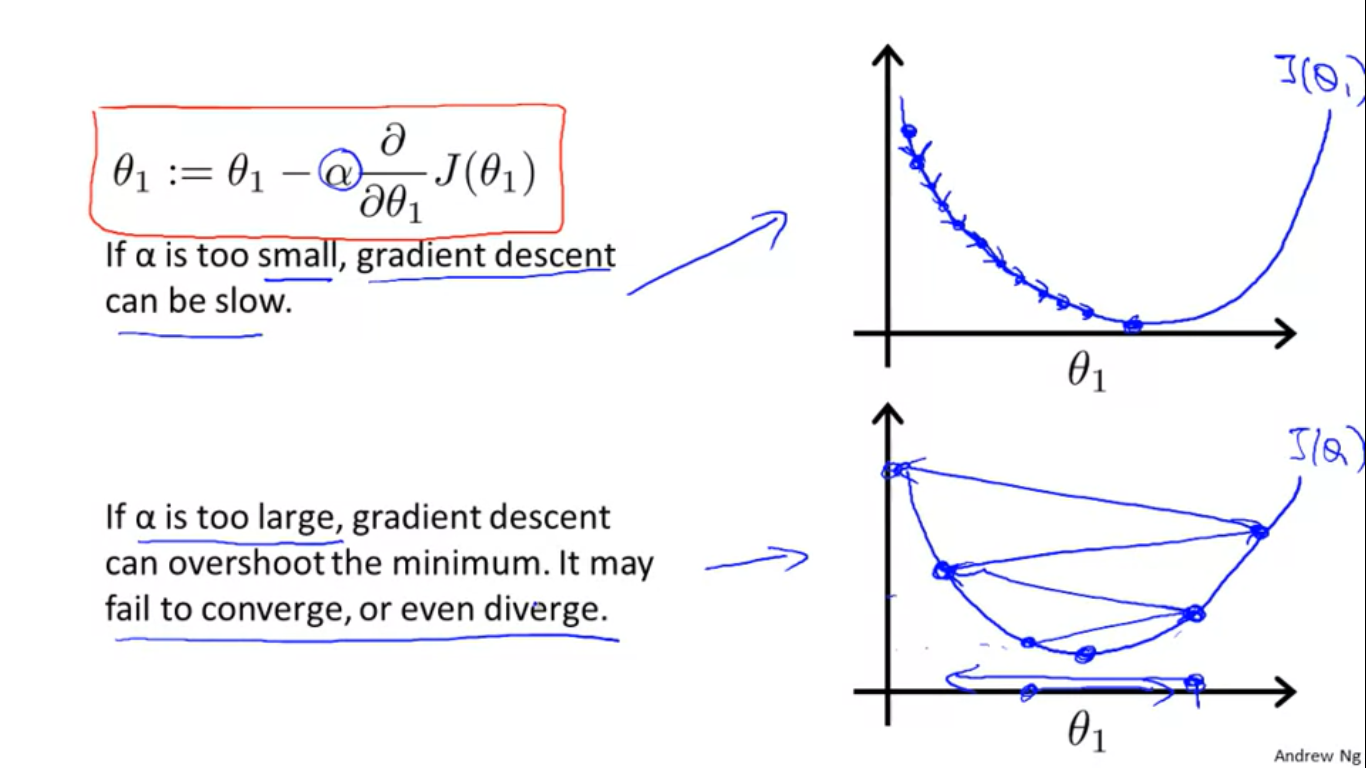


Gradient descent (Intuition):

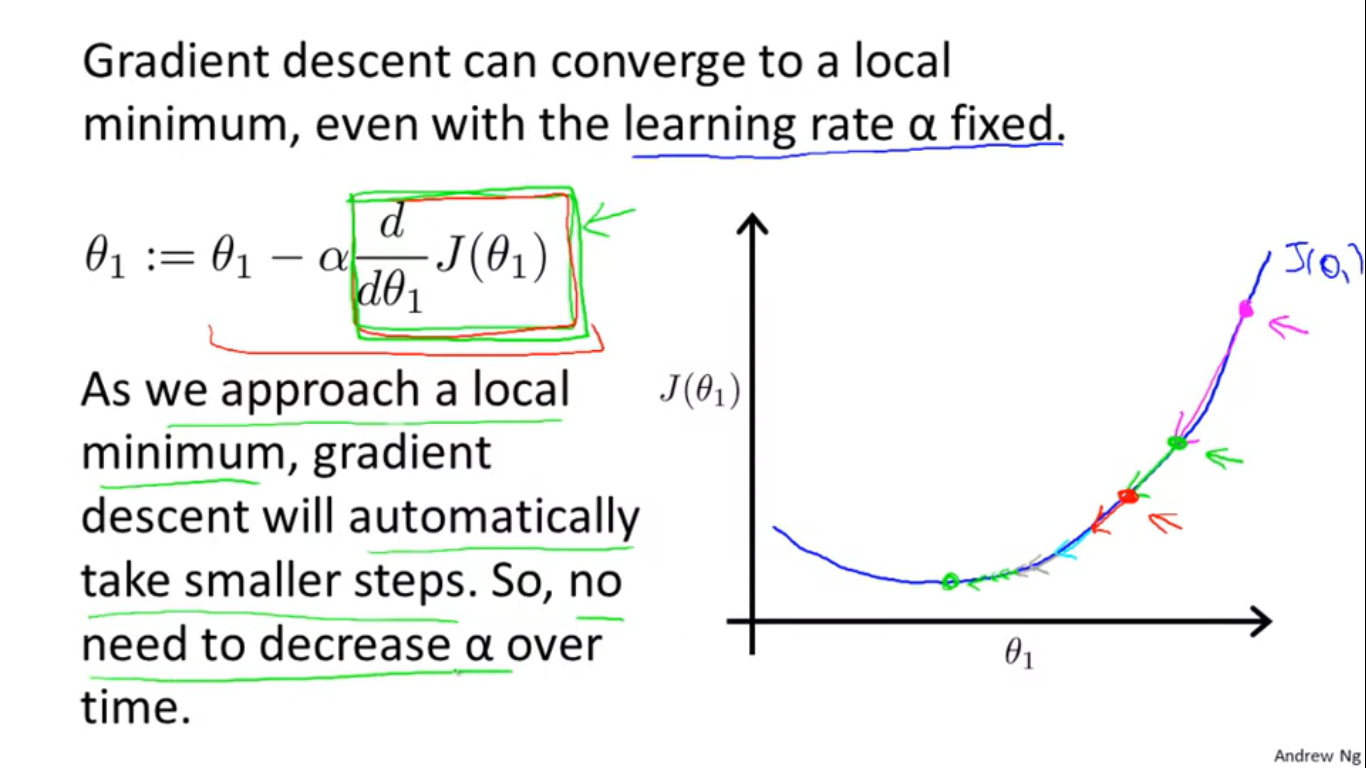
Regardless of the slope's sign for d/d*θ1* [*J*(*θ1*)] , *θ1* eventually converges to its minimum value. The following graph shows that when the slope is negative, the value of *θ1* increases and when it is positive, the value of *θ1* decreases.



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.

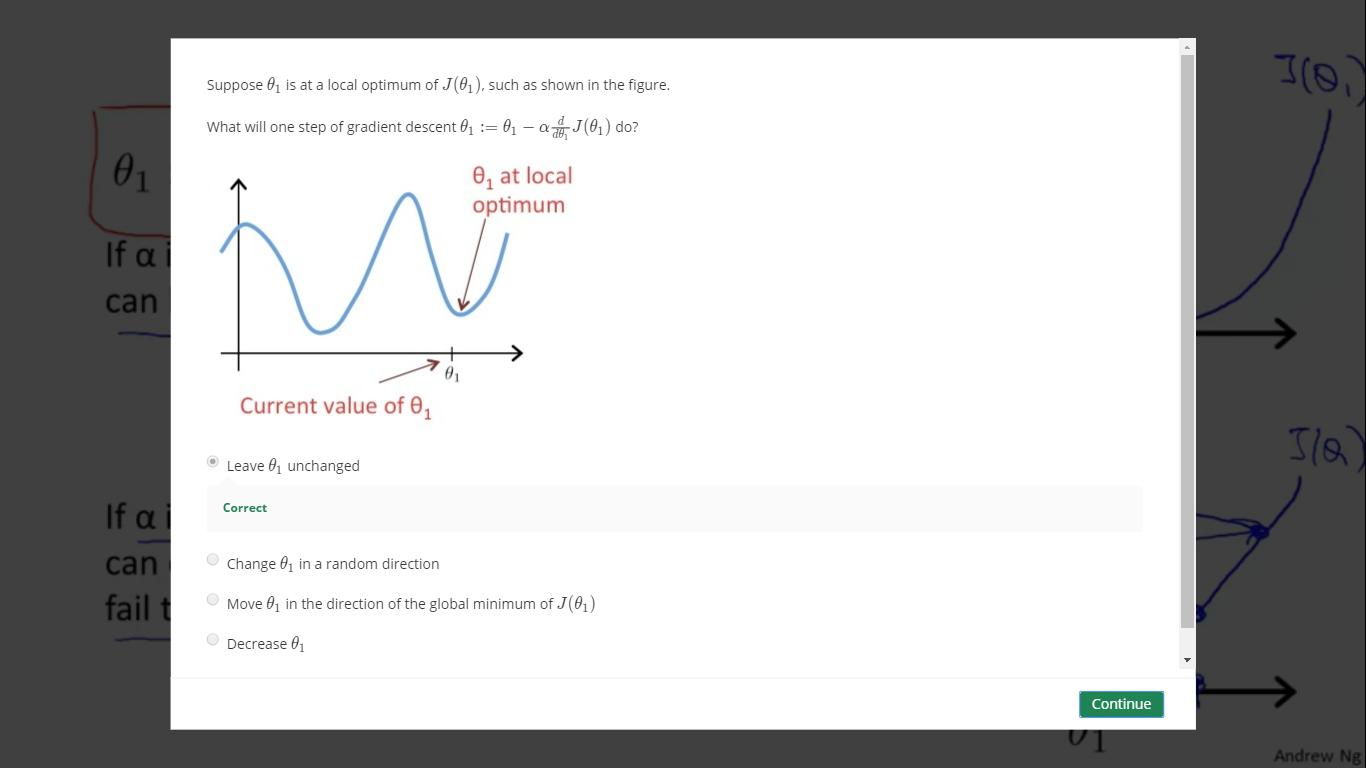






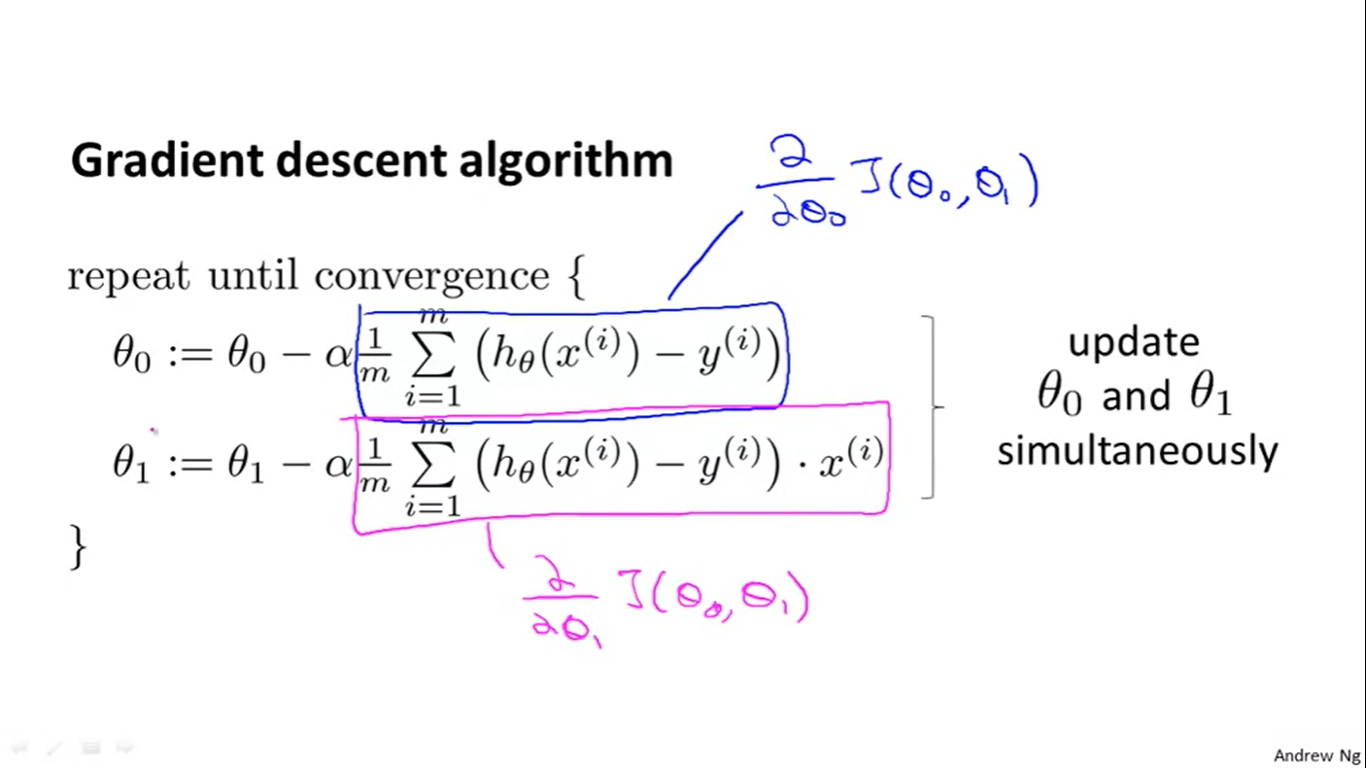
The green boxed term is the derivative.If the derivative is 0 then we have arrived at local minima.

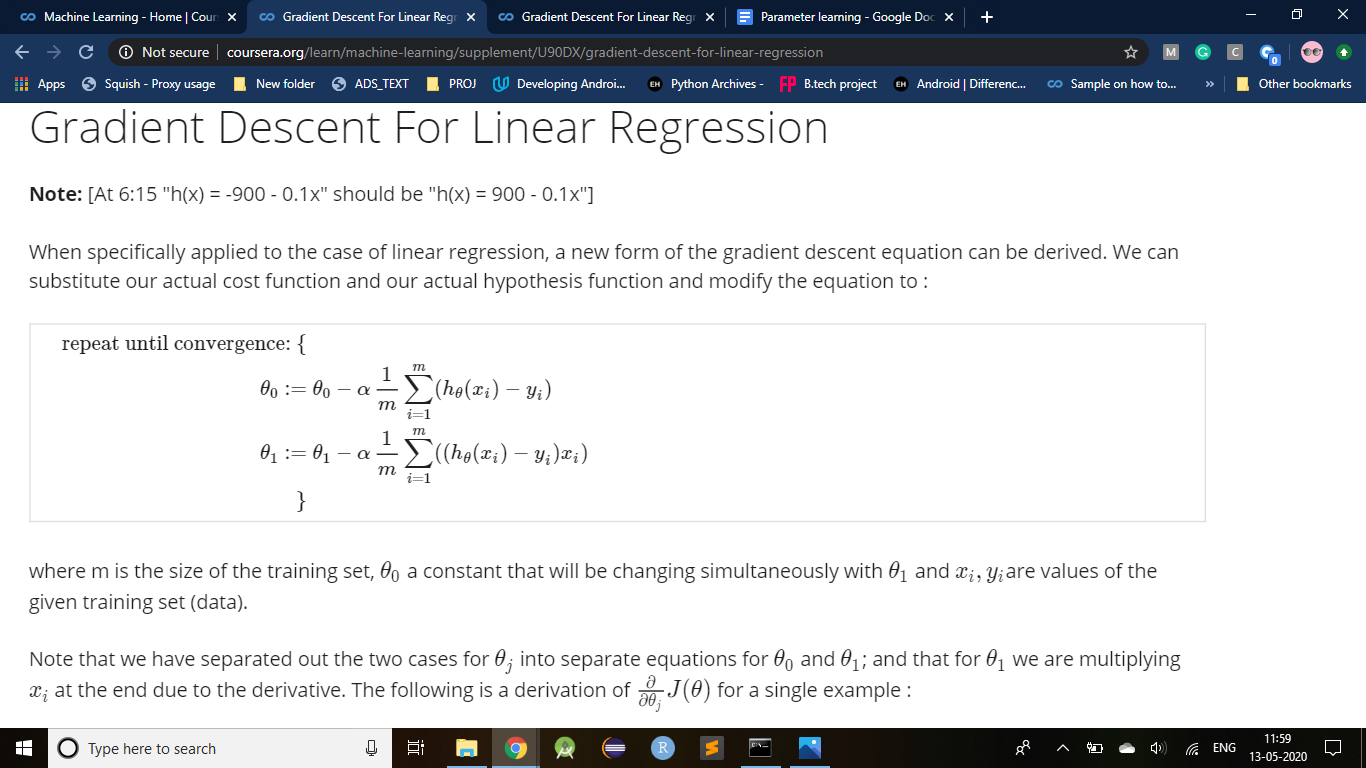
Question :



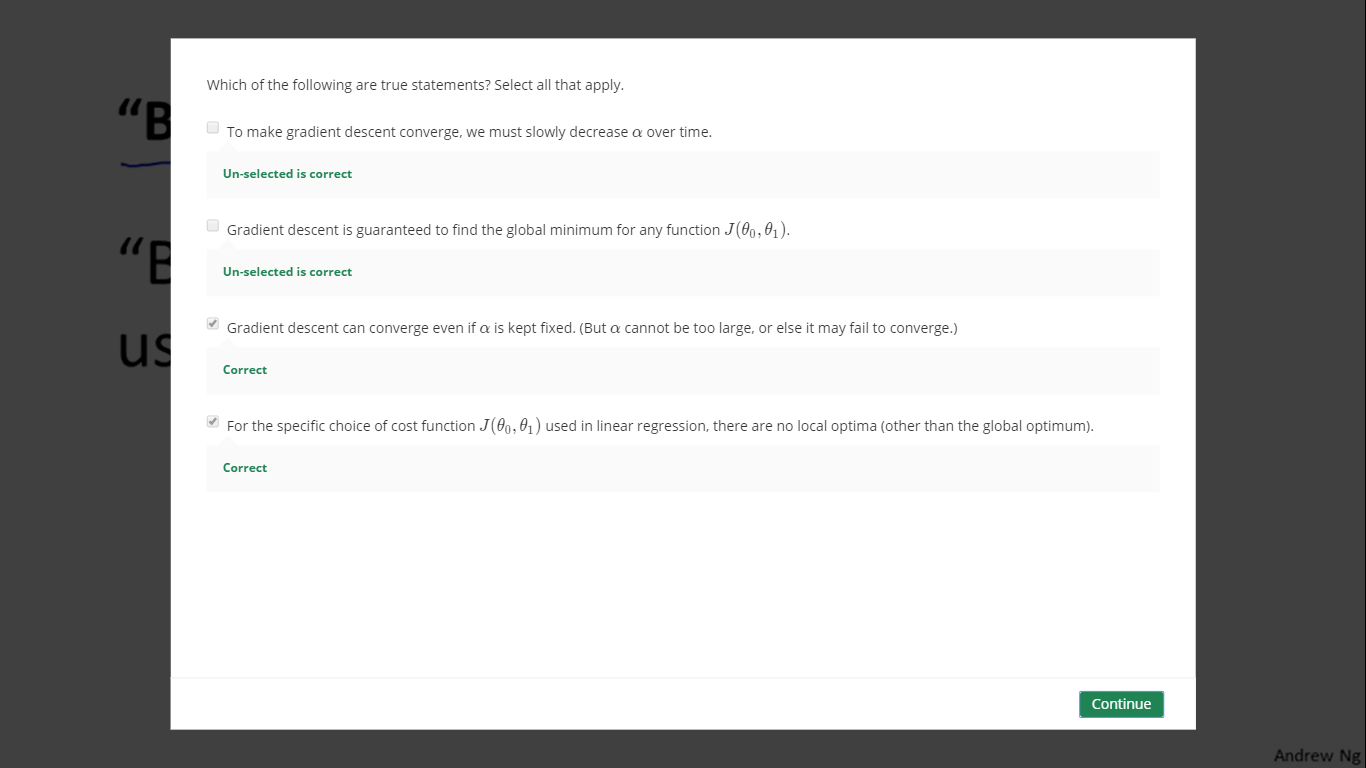
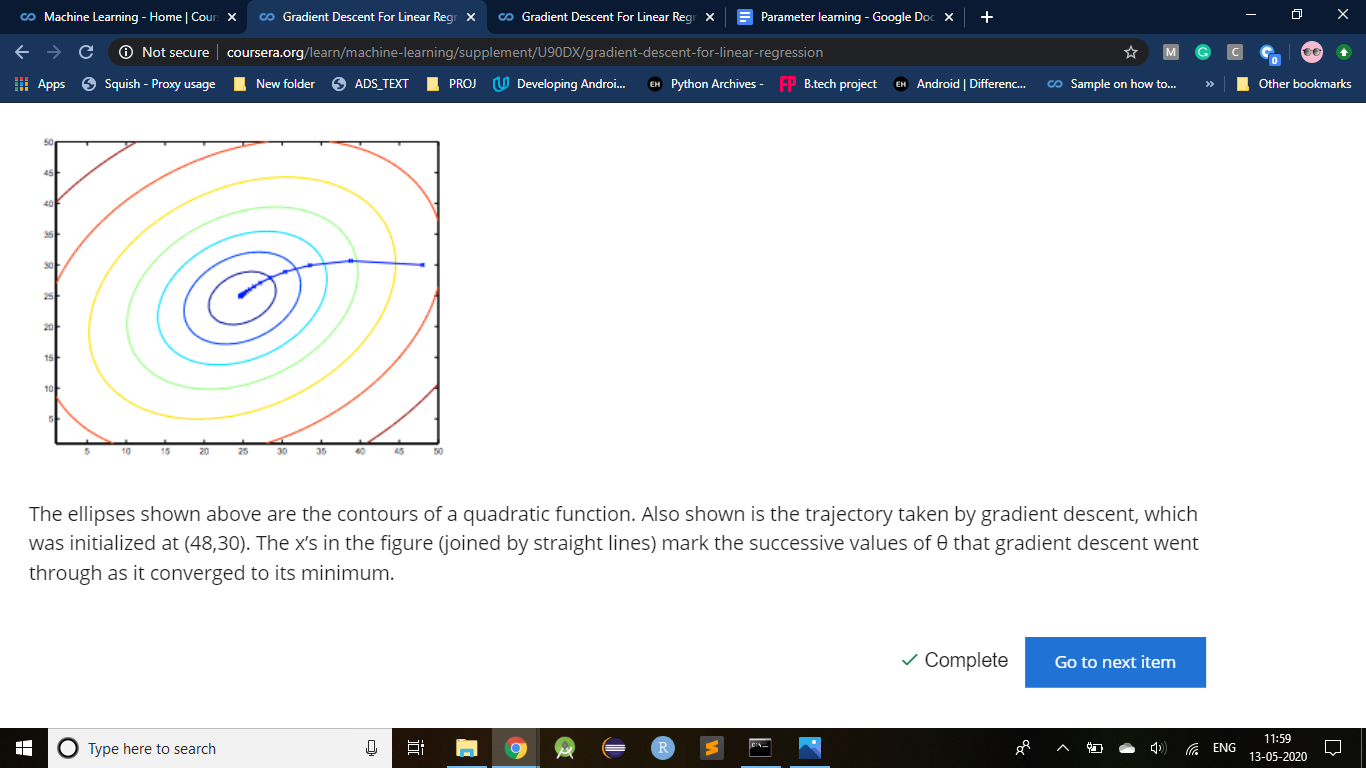
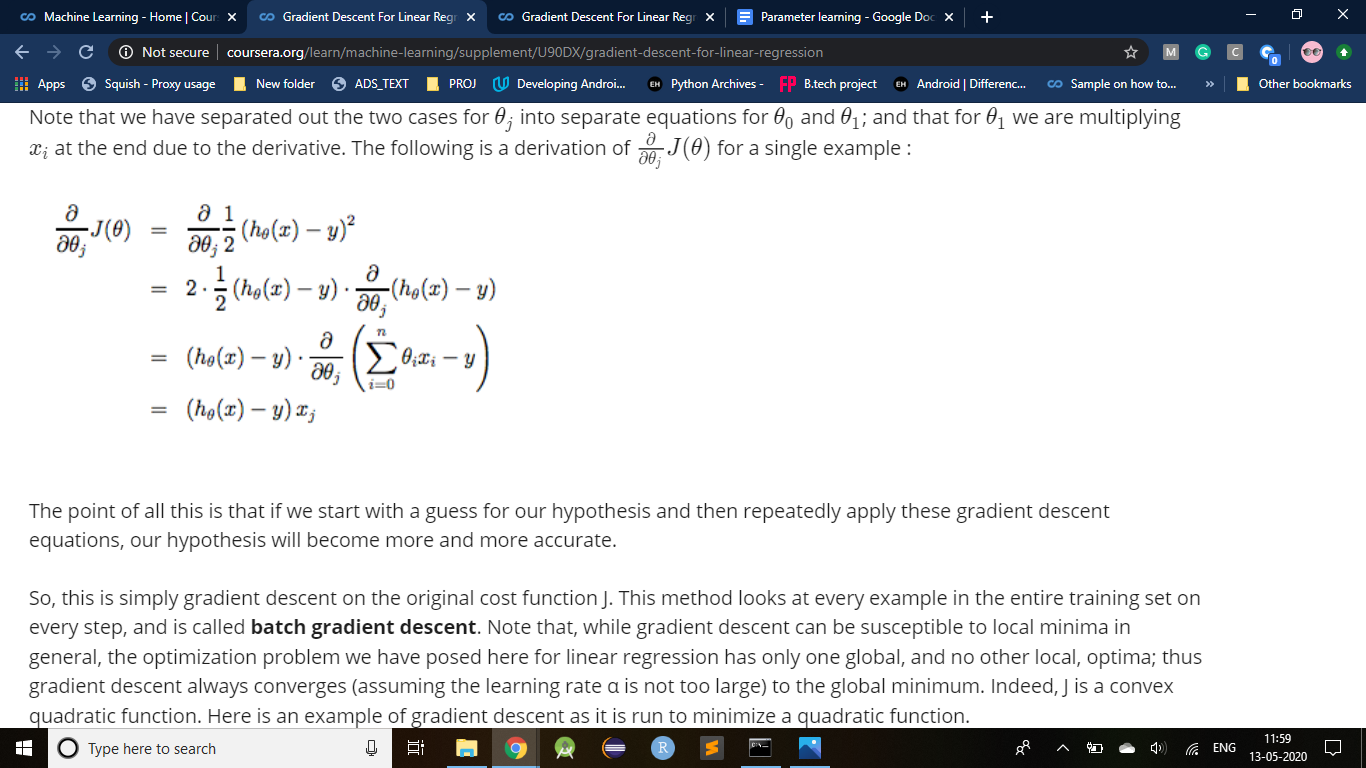
#### Gradient Descent For Linear Regression :

Following is the gradient descent algorithm :

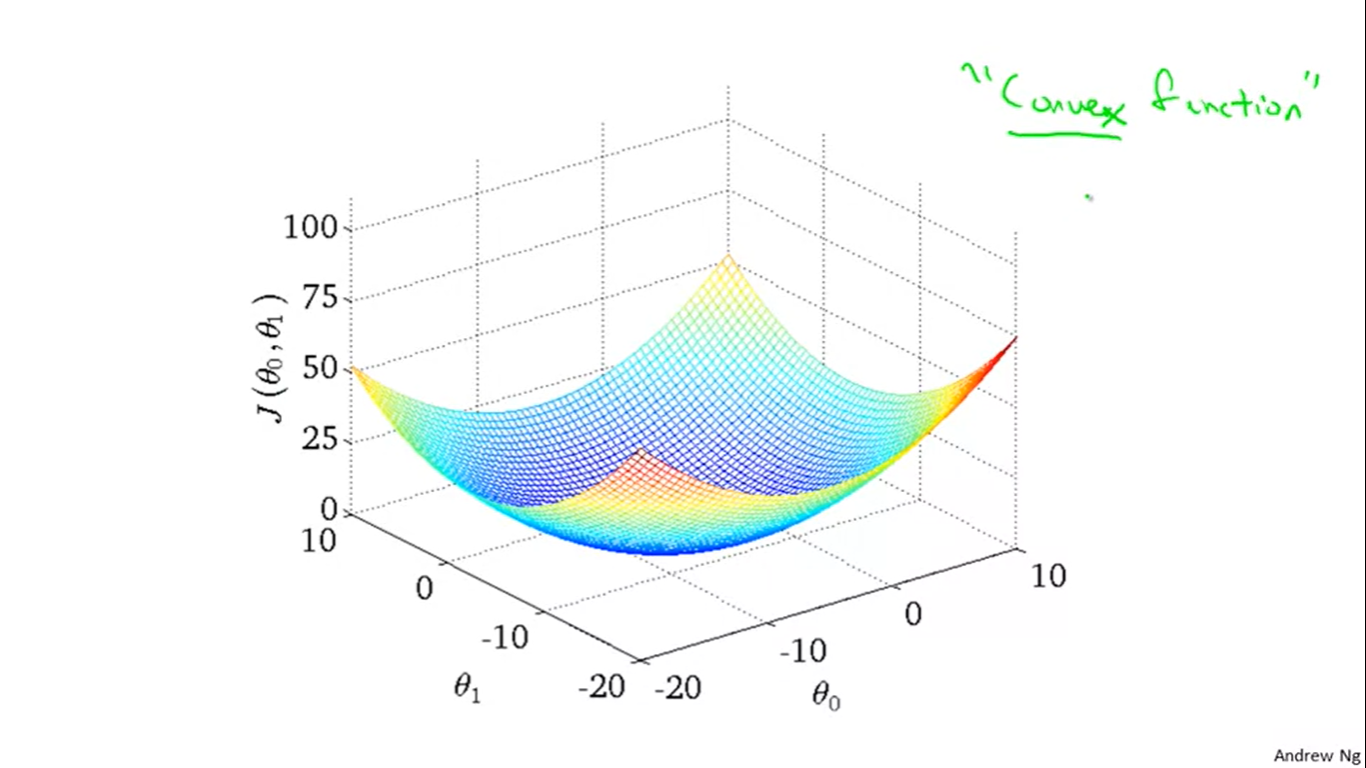




If we start with a guess for our hypothesis & then repeatedly apply gradient descent equations,our hypothesis will be more and more accurate.



Linear regression always has a bowl-shaped curve : (with only one global optimum)



Whereas, on applying gradient descent in general it has many local optimum :

