3. Gradient Descent

August 5, 2021

1 What is Gradient Descent

Gradient descent is an algorithm used to minimize a cost function. Although previously we considered only linear regression problems, the gradient descent algorithm can be applied to other functions as well.

The algorithm works like this (again considering only 2 parameter values):

1. Start with an initial guess of the parameters
2. Keep changing the parameter values with a small change (or step) to reduce the cost function $J(\theta_0, \theta_1)$ until we arrive at a global minimum

We can depict the above steps visually

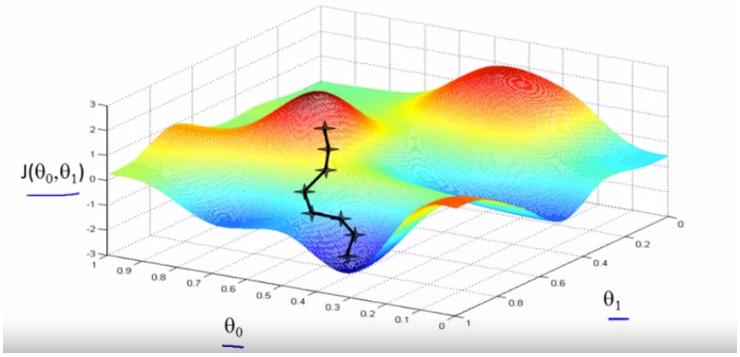


Figure 1: We start at some arbitrary values for θ_0 and θ_1, change our guess repeatedly until the cost function yields the smallest value it can possibly take on

Our initial value for (θ_0, θ_1) is not always going to guarantee we arrive at the global minimum. In the figure below, the initial guess of the parameters lead us to find a local minima

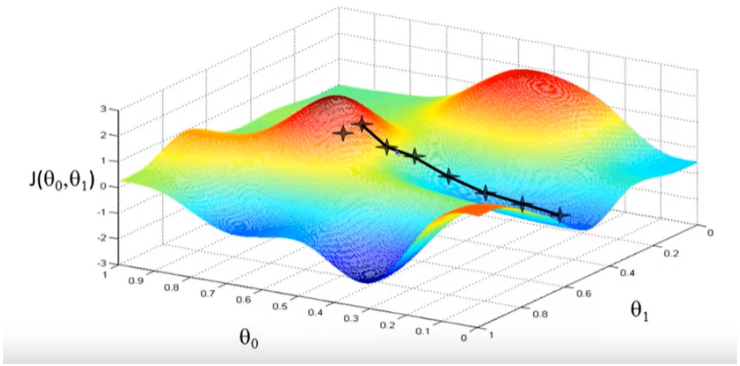


Figure 2: Inital guess of parameters lead us to find a local minima rather than the desired global minimum

The property of the gradient descent algorithm to lead to different minimas based on the initial guess of the parameters is a property we will explore in section 2.1

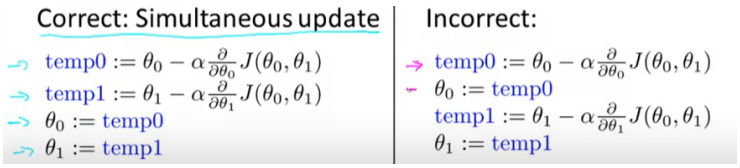
The mathematical definition of the gradient descent algorithm is given below

$$\theta_j \quad := \quad \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1); (j = 0, j = 1) \tag{1}$$

repeated until convergence
where:

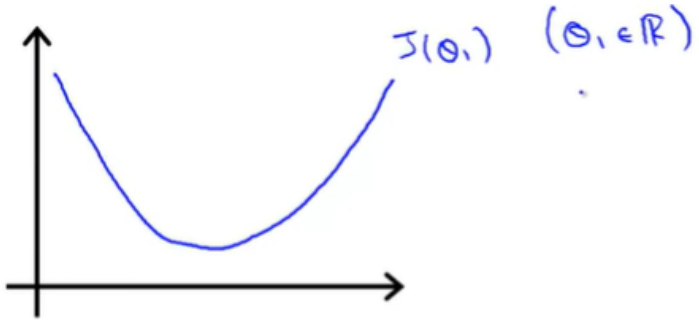
1. := is the assignment operator
2. α is called the learning rate and controls how big a step we take to find the minimum (larger α means biger steps, conversely, smaller α means we take smaller steps)
3. The derivative of the cost function will be explained later

Keep in mind that the mathematical definition of the gradient descent function says to simultaneously update the parameter values

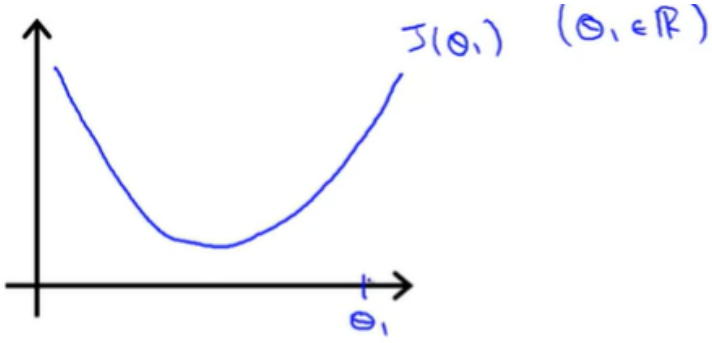


2 Gradient Descent Intuition 1

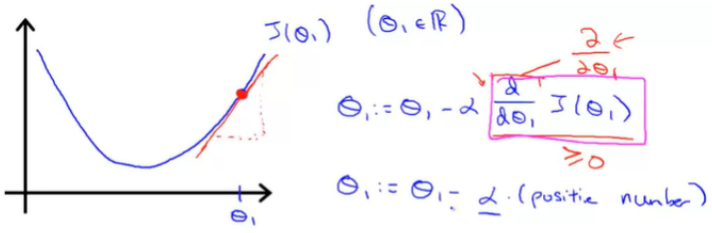
Consider a cost function that contains only 1 parameter θ_1



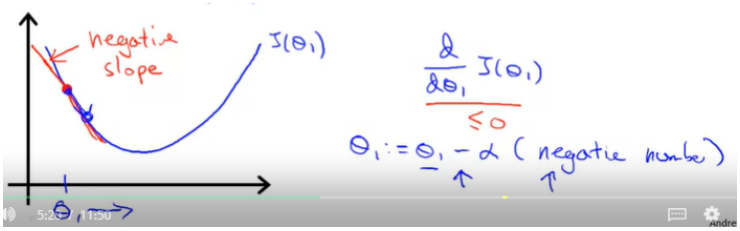
Now imagine we have an initial value of θ_1 as shown in the figure below



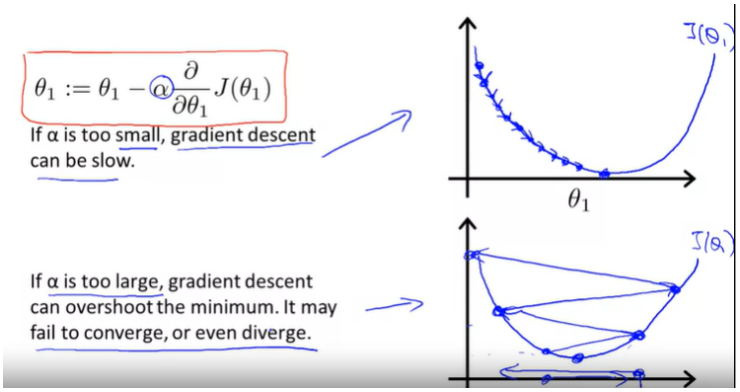
For the 1 variable cost function, the gradient descent algorithm becomes a simple derivative. Finding the derivative at the initial point will yield us the slope/gradient of a tangent to the point on the graph (yields a positive value). Thus the new value for θ_1 will tend towards the minimum of the cost function.



If the initial value was to the left of the minimum the gradient would be negative and we would still be yielded a new value that tends towards the minimum of the cost function

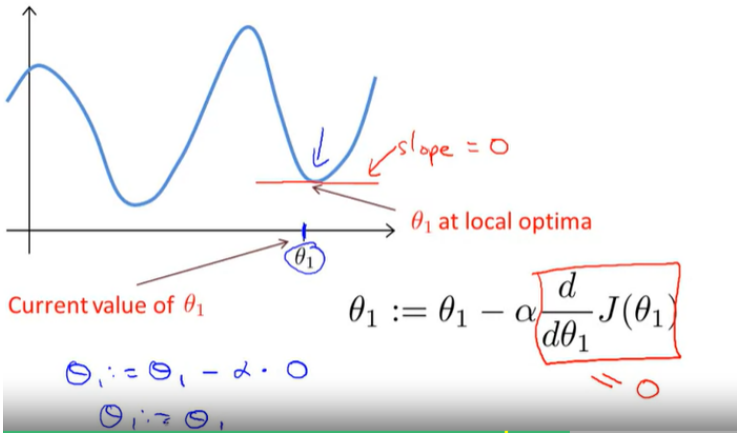


Looking at the learning rate α we can see that the smaller it is, the more changes need to be made to θ_1 to get to the minimum and the bigger it is the greater the change in θ_1 would be leading to the possibility of overshooting our minimum.



2.1 Initial Value of Parameters

Recall previously we said that an initial value of the parameters determine whether we achieve a global min or a local min. Here we explain why. Consider we set the initial value of θ_1 to a value to the right of the local minimum which is the current value of θ_1. Gradient descent would take us to the current local minimum and it would stop (it will not achieve global min). This is because the gradient at this point is 0, the gradient descent algorithm will want to keep this value as is.



If my memory serves me correctly, I believe finding the 2nd derivative of the function determines whether we are at a local min or a global min (more on this in sections to come that look at finding global min)

3 Gradient Descent for Linear Regression

Recall for linear regression the hypothesis function looks as follow $h_\theta(x) = \theta_0 + \theta_1 \times x$ and the cost function is given as $J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^i) - y^i)^2$

Now to find gradient descent for linear regression we substitute in the hypothesis and cost function

$$\begin{aligned} \theta_j &= \theta_j - \alpha \tfrac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) \\ &= \theta_j - \alpha \tfrac{\partial}{\partial \theta_j} [\tfrac{1}{2m} \textstyle\sum_{i=1}^{m} (h_\theta(x^i) - y^i)^2] \\ &= \theta_j - \tfrac{\alpha}{2m} \textstyle\sum_{i=1}^{m} \tfrac{\partial}{\partial \theta_j} (\theta_0 + \theta_1 x^i - y^i)^2 \end{aligned} \tag{2}$$

$$\begin{aligned} \therefore \theta_0 &= \theta_0 - \tfrac{\alpha}{2m} \textstyle\sum_{i=1}^{m} (2)(1)(\theta_0 + \theta_1 x^i - y^i) \\ &= \theta_0 - \tfrac{\alpha}{m} \textstyle\sum_{i=1}^{m} (\theta_0 + \theta_1 x^i - y^i) \end{aligned} \tag{3}$$

and

$$\begin{aligned} \therefore \theta_1 &= \theta_1 - \tfrac{\alpha}{2m} \textstyle\sum_{i=1}^{m} (2)(x^i)(\theta_0 + \theta_1 x^i - y^i) \\ &= \theta_1 - \tfrac{\alpha}{m} \textstyle\sum_{i=1}^{m} (\theta_0 + \theta_1 x^i - y^i) x^i \end{aligned} \tag{4}$$

And here we see the need for the linear regression cost function to have been divided by 2 as the derivatives of the cost function lead to its cancellation with a numerator 2.

3.1 Batch Gradient Descent with Linear Regression

The summing terms of equation 3 and equation 4 says that we use all the training examples in determining the parameters per gradient descent step, this gradient descent algorithm is known as batch gradient descent

There are other kinds of gradient descent algorithms that we will explore later on