Gradient Descent

Authors: Lai Jiang, Bryant Leal, Yiming Jin. (PDF)

After learning about linear regression and its proof, we have dived into an alternative method that will reach the same answer that linear regression can solve. The method is called gradient descent which will be further developed into a much less computationally intensive method called Stochastic Gradient Descent. It is vital that we begin this supplement first with a general idea of gradient descent before diving into SGD.

What is gradient descent?

Gradient descent is an optimization algorithm which finds the coefficient of a function(f) that minimizes a cost function (e.g., MSE, Cross-entropy cost, Hellinger distance)

We can think of gradient desecent as a bowl and the bowl's inner surface is a plot of the cost function. a random position on the inner surface of bowl is the cost of the current values of coefficients. The bottom of the bowl is the cost of the optimal coefficients.

The goal is to try different values for the coefficient, evaluate their cost and select new coefficients at a defined learning rate that have a slightly better (lower) cost.

Repeating this process enough times will lead to the bottom of the bowl and you will know the values of the coefficients that result in the minimum cost. Note that as the cost become closer to zero, the steps size becomes smaller and smaller.

> Below is a simplified mathematical representation of the steps to conduct gradient descent. Let's use SSE as a cost function:

Step 1: Initialize the weights(a & b) with random values and calculate Error (SSE)

Function: $\hat{Y} = \hat{\alpha}_0 + \hat{\beta}_1 X$

Loss Function: $\frac{1}{2}\sum(Y - \hat{Y})^2$

Step 2: Calculate the gradient i.e. change in SSE when the weights (a & b) are changed by a very small value from their original randomly

initialized value. This helps us move the values of a & b in the direction in which SSE is minimized.

$\frac{\partial SSE}{\partial a} = -2 * (Y - \hat{Y})$

$\frac{\partial SSE}{\partial b} = -2 * (Y - \hat{Y}) * X$

$SSE = (Y - \hat{Y})^2 = (Y - (a + betaX))^2$

Step 3: Adjust the weights with the gradients to reach the optimal values where SSE is minimized

$a = \frac{\partial SSE}{\partial a}$

$b = \frac{\partial SSE}{\partial b}$

$New_a = a - r * \frac{\partial SSE}{\partial a}$

$New_b = b - r * \frac{\partial SSE}{\partial b}$

Where r is the learning rate, think of it as the pace at which we are adjusting the weight. It is important to note that SGD and GD in general are sensitive to the learning rate, where too high of a rate will create steps that will likely miss (or Diverge from) the optimal weights. Too low of a rate may take a long time to reach an optimal solution.

Step 4: Use the new weights for prediction and to calculate the new SSE

Function: $\hat{Y}_{new} = New_a + New_b X$

Loss Function: $\frac{1}{2} \sum (Y_{new} - \hat{Y}_{new})^2$

Step 5: Repeat steps 2, 3 and 4 until further adjustments to weights does not significantly reduce the loss function Sum(Y − Ypred)2

Ultimately, our final answer will yield weights that minimize the loss function. In our simplified example, this means where the derivates of a and b are zero or close to it.

> This is the Gradient Descent Algorithm. This optimization algorithm and its variants form the core of many machine learning algorithms like Neural Networks and even Deep Learning.

Professor also updated us on the covid-19 issue since the number of death surpassed 300k.

Stochastic Gradient Descent (SGD)

Gradient Descent is very computationally demanding especially on big datasets when calculating the derivative of the loss function with respect to each parameter for each point *and* also calculating how big a step to take based on those weights. Therefore, Stochastic Gradient Descent (SGD) comes into play by picking a

random data point from the data set and performing the GD on it - making SGD more scalable. This process continues until the optimal weights are found.

SGD is an itrative method for optimizing an objective function with suitable smoothness properties. Stochastic gradient descent replaces true gradient by "instantaneous" gradient, that reduces error only on the new instance. We explored linear models in class, with τ as (inner loop) iteration number, and n denoting the datapoint being considered:

$W^{(\tau+1)} = W^{\tau} - \eta \nabla E_n = W^{\tau} + \eta(t_n - W^{(\tau)T}\eta(X_n))\eta(X_n)$

Learning rate: η

Error: $t_n - W^{(\tau)T}\eta(X_n)$

Input: X_n

This is also known as Least-mean-squares (LMS) algorithm or the Widrow-Hoff rule.

References

Below are some videos that we found helpful when trying to explain what SGD is trying to do.

Gradient Descent: https://www.youtube.com/watch?v=sDv4f4s2SB8

Stochastics Gradient Descent: https://www.youtube.com/watch?v=vMh0zPT0tLI&feature=youtu.be