Neural Network Foundations, Explained: Updating Weights with Gradient Descent & Backpropagation

Reflection on task

Change the iteration number and learning_rate and observe how cost decreases.

Iterations = 100
Learning rate = 0.08
Cost reduced to <1 by iteration 22
<0.1 by iteration 43
cost 1.711074266712627e-05 iteration 199 (for 200 iterations)

Iterations = 100
Learning rate = 0.5
Cost kept increasing and it could be seen that the gradient kept fluctuating between positive and negative meaning the global minimum kept being missed and overshot by a lot.

Iterations = 200
Learning rate = 0.02
Cost < 1 by iteration 6
< 0.1 by iteration 166

Iterations = 200
Learning rate = 0.1
Cost kept increasing.

Iterations = 200
Learning rate = 0.81
Cost < 1 by iteration 28
< 0.1 by iteration 47
cost 1.4900270855963761e-05 iteration 199

Iterations = 200
Learning rate = 0.079
< 1 by iteration 18
< 0.1 by 42

Learning rate of 0.079 is the best value I achieved when experimenting with the learning rate. I kept the iterations at 200 for each experiment and checked by what iteration number the cost reached < 1 and < 0.1.

For a learning rate of 0.079, the gradient descent cost function simulation reached a cost of less than 0.1 by iteration 42 and by iteration 199 the cost was ~1.96E-05.

Background reading

Weights of a neural network get updated after a forward pass of data, this is how it learns. Back Propagation of error is when each neuron's respective error in a neural network is sent backward to them. Weights need to be updated for large neural networks with many layers to conserve computing power and time.

Imagine a cost function used to determine error of predicted values, the lowest point on the cost function would be where the rate of change (gradient) is equal to 0. The process of finding this point of optimal value is called gradient descent.
A model's learning rate indicates how far weights are adjusted to move across the cost function to find the optimal gradient.

Stochastic gradient descent uses random sampling of the data to do cost calculations and back propagate errors, It helps combat local minima by making weight adjustments after every data instance. It has a better chance at finding the global minima (Mayo, 2017).

References

Mayo, M. (2017). Neural Network Foundations, Explained: Updating Weights with Gradient Descent & Backpropagation. KDnuggets. Available at: https://www.kdnuggets.com/2017/10/neural-network-foundations-explained-gradient-descent.html [Accessed 25 Jan. 2025].