Roshan Poudel - HW2 Part 4

Gradient Descent

What is Gradient?

When we talk about a single-variable function such as f(x) = 2x2, we can calculate its slope at certain point by evaluating the derivative of the function at that point. For eg: the slope of the above function at point, say (1,2) would be: - STEP 1: $f'(x) = 4x$ - STEP 2: Evaluating the derivative of the function at x=1: $f'(1) = 4 * 1 = 4$ So, the slope of the function $f(x) = 2x^2$ at the point $(1, 2)$ is 4. This means that for a small change in x around $x = 1$, the corresponding change in $f(x)$ is 4 times that change in x.

Now let's extend this concept to a multivariable function like f(x,y). The exact same thing (calculating derivative) we do, but we do it with partial derivaties which means calculating derivatives wrt one variable while keeping the other variable(s) constant. That gives us the slope of the tangent plane (similar to the slope of tangent line in single-variable function) at certain locus. That slope of the tangent plane is the gradient of the function. The gradient (which is a vector of partial derivatives wrt input variables) points in the direction of maximum increase of the function. If we calculate the magnitude of the gradient vector, then we get the rate of change. The larger the magnitude, the steeper the function is at that point.

Gradient descent in deep learning

In the deep learning model, our goal is to minimize the value of the loss function by repeatedly adjusting weight matrices until we hit the global minimum of the loss function (i.e match the predicted output to the true output as closely as possible with minimum loss). How do we adjust the weights? Do we increase or decrease the coefficients? That's where gradient descent comes in. So, we find the gradient of the loss function locally to determine which direction and how much to move until we hit that minimum loss. If we know the gradient at a certain point, adjusting the weight opposite to the direction of gradient, takes the loss to the lower region of the curvature where we tend closer to the global minimum.

To summarize, Gradient descent is used to minimize a loss function by updating the parameters of a neural network model. The algorithm works by computing the gradient of the loss with respect to the model parameters, and then updating the parameters in the direction of the negative gradient, so as to reduce the value of the loss. The magnitude of the update step is determined by a learning rate, which determines the speed at which the model parameters are changed.

The process is repeated iteratively until the loss reaches a minimum or a stopping criterion is met.

Back Propagation

Backpropagation is an efficient method for computing the gradients of complex neural network models. It uses the derivatives of basic operations, such as addition, relu, or tensor product, to calculate the gradient of the entire network, which is composed of many simple tensor operations chained together.

The algorithm starts with a forward pass, where the input is passed through the network to make a prediction. The error between the predicted output and the actual output is then calculated using the loss function, and this error is used to update the weights of the network in a backward pass, hence the name "backpropagation".

During the backward pass, the error is propagated backward through the network, layer by layer, and the chain rule of calculus is used to calculate the gradient of the error with respect to each weight in the network. This gradient is then used to adjust the weights of the network in a direction that reduces the error.

The gradient of the error with respect to each weight is calculated by multiplying the gradients of each intermediate step in the calculation, which in essence is 'chain rule'.

Global Minumum, Local Minimum, and Momentum

Global Minimum: The global minimum refers to the point in the graph of function where the function has the minimum value over the entire parameter space. The goal of the optimization algorithm used to train the neural network is to find the global minimum of the loss function, which represents the best solution for the neural network.

Local Minimum: A local minimum is a minimum value that is only valid within a restricted region of the function's parameter space. In other words, a local minimum refers to a point in the graph of the function where the function has a minimum value only within a ceratin region, but this value may not be the global minimum of the function. During the gradient descent in the loss function, the optimization algorithm can get stuck in a suboptimal solution that is local minimum, and not the global minimum of the loss function. To avoid this, gradient descent is done using momentum.

Momentum: It is the technique to speed up the convergence to overcome local minima. Momentum basically keeps track of previous direction of travel. In the context of gradient descent, the optimization algorithm calculates the gradient of the loss function with respect to the weights and updates the weights in the direction of the negative gradient. With momentum, the optimization algorithm calculates the gradient of the loss function with respect to the weights and also keeps track of the moving average (average of the gradients over time). The moving average is then used to update the weights, instead of just using the current gradient. This helps to provide the smoothed representation of gradient over time which helps the error value converge efficiently and avoid getting stuck in local minima.