Day 35

DIY Solution

Q1. Discover other major types of Gradient descent.

Answer: There are three gradient descent learning algorithms types, batch gradient descent, stochastic gradient descent, and mini-batch gradient descent.

1. **Batch gradient descent:** Batch gradient descent sums the error for each point in a training set, updating the model only after all training examples have been evaluated. This process is referred to as a training epoch.

   While this batching provides computation efficiency, it can still have an extended processing time for extensive training datasets as it still needs to store all of the data in memory. Batch gradient descent also usually produces a stable error gradient and convergence, but sometimes that convergence point isn't the most ideal, finding the local minimum versus the global one.

2. **Stochastic gradient descent:** Stochastic gradient descent (SGD) runs a training epoch for each example within the dataset, and it updates each training example's parameters one at a time. Since you only need to hold one training example, they are easier to store in memory. While these frequent updates can offer more detail and speed, they can result in losses in computational efficiency when compared to batch gradient descent. Its regular updates can result in noisy

gradients, but this can also help escape the local minimum and find the global one.

3. **Mini-batch gradient descent:** Mini-batch gradient descent combines batch gradient descent and stochastic gradient descent concepts. It splits the training dataset into small batch sizes and performs updates on each batch. This approach strikes a balance between the computational efficiency of batch gradient descent and the speed of stochastic gradient descent.

Q2. The problem with Gradient descent

Answer: Major problems faced by Gradient descent are,

1. **Vanishing Gradient:** If the range of the initial values for the weights is not carefully chosen, and the scope of the importance of the consequences during training is not controlled, a vanishing gradient would occur, which is the foremost hurdle to learning in deep networks. The neural networks are trained using the gradient descent algorithm:

$$w' = w - \eta \delta J / \delta w$$

Where J is the loss of the network on the current training batch, if the $\delta J/\delta w$ is very small, the learning will be prolonged since the changes in w will be minimal. So, if the gradients vanish, the knowledge will be pretty slow.

The vanishing gradient is that, during backpropagation, the gradients of inceptive layers are obtained by multiplying the gradients of concluding layers. So, if the gradients of the concluding layers are less than one, their multiplication vanishes very fast.

To summarize :

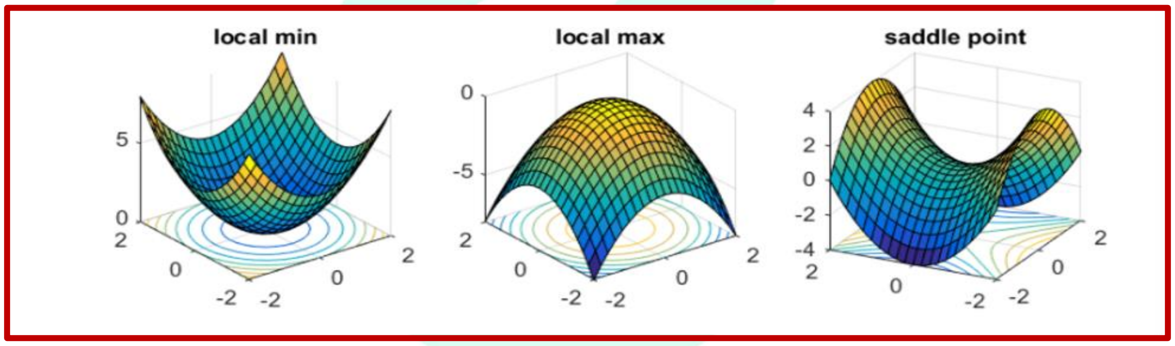
i. The gradient is the gradient of the loss function J concerning each trainable parameter (weights w and biases, b).

ii. A vanishing gradient does not mean the gradient vector is zero (except for numerical underflow), but the angles are so small that the learning will be prolonged.

2. **Exploding Gradient:** Exploding gradient occurs when the derivatives increase as we go backward with every layer during backpropagation. This is the exact opposite of vanishing gradients.

   The architecture becomes unstable due to significant changes in a loss at each update step. The weights grow exponentially and become very large, and derivatives stabilize.

3. **Saddle Point:** A saddle point on the surface of the loss function is a point where, from one perspective, that critical point looks like a local minimum, while from another perspective, it looks like a local maximum.



   The saddle point injects confusion into the learning process. Model learning stops (or becomes extremely slow) at this point, thinking that "minimum" has been achieved since the slope becomes zero (or very close to zero). This critical point is the maximum cost value from the other perspective. The saddle point comes into view when gradient descent runs in multidimensions.

Q3. How do handle these problems?

Answer:

1. **Changing the architecture:** This solution could be used in both the exploding and vanishing gradient problems but requires a good understanding and outcomes of the change. For example, suppose we reduce the number of layers in our network. In that case, we give up some of our model's complexity since having more layers makes the networks more capable of representing complex mappings.
2. **Gradient Clipping for Exploding Gradients:** They carefully monitor and limit the gradients' size, while our model trains are yet another solution. This requires some deep knowledge of how the changes could impact the overall performance.
3. **Careful Weight Initialization:** A more careful initialization of the model parameters for our network is a partial solution since it does not solve the problem completely.

Q4. Limitation of ridge and lasso regression

Answer:

1. **Limitation of Ridge Regression:** Ridge regression decreases the complexity of a model but does not reduce the number of variables since it never leads to a coefficient being zero rather only minimizes it. Hence, this model is not suitable for feature reduction.
2. **Limitation of Lasso Regression:** Lasso sometimes struggles with some data types. If the number of predictors (p) is greater than the number of observations (n), Lasso will pick at most n predictors as non-zero, even if all predictors are relevant (or may be used in the test set). If there are two or more highly collinear

variables, then LASSO regression selects one of them randomly, which is not suitable for the interpretation of data.

Q5. What is the elastic net?

Answer: The lasso regression can cause a slight bias in the model where the prediction depends on a particular variable. In these cases, elastic Net is proved to be better. It combines the regularization of both lasso and Ridge. The advantage of that it does not quickly eliminate the high collinearity coefficient.

The elastic net method overcomes the limitations of the LASSO (most minor absolute shrinkage and selection operator) method, which uses a penalty function based on:

$$|\beta|_1 = \sum_{j=1}^{p} |\beta_j|$$

The use of this penalty function has several limitations. For example, in the "large p, small n" case (high-dimensional data with few examples), the LASSO selects most n variables before it saturates. Also, if there is a group of highly correlated variables, then the LASSO tends to favor one variable from a group and ignore the others. To overcome these limitations, the elastic net adds a quadratic part to the penalty, which, when used alone, is ridge regression (also known as Tikhnov regularization). The estimates from the fexible net method are defined by,

$$\beta^{\wedge} = argmin_{\beta}(||y - X\beta\,||^2 \,+\, \gamma_2||\,\beta||^2 + \gamma_1\,||\,\beta\,||_1\,)$$