Optimization techniques for Gradient Descent

Gradient Descent is a widely used optimization algorithm for machine learning models. However, there are several optimization techniques that can be used to improve the performance of Gradient Descent. Here are some of the most popular optimization techniques for Gradient Descent:

Learning Rate Scheduling: The learning rate determines the step size of the Gradient Descent algorithm. Learning Rate Scheduling involves changing the learning rate during the training process, such as decreasing the learning rate as the number of iterations increases. This technique helps the algorithm to converge faster and avoid overshooting the minimum.

Momentum-based Updates: The Momentum-based Gradient Descent technique involves adding a fraction of the previous update to the current update. This technique helps the algorithm to overcome local minima and accelerates convergence.

Batch Normalization: Batch Normalization is a technique used to normalize the inputs to each layer of the neural network. This helps the Gradient Descent algorithm to converge faster and avoid vanishing or exploding gradients.

Weight Decay: Weight Decay is a regularization technique that involves adding a penalty term to the cost function proportional to the magnitude of the weights. This helps to prevent overfitting and improve the generalization of the model.

Adaptive Learning Rates: Adaptive Learning Rate techniques involve adjusting the learning rate adaptively during the training process. Examples include Adagrad, RMSprop, and Adam. These techniques adjust the learning rate based on the historical gradient information, which can improve the convergence speed and accuracy of the algorithm.

Second-Order Methods: Second-Order Methods use the second-order derivatives of the cost function to update the parameters. Examples include Newton's Method and Quasi-Newton Methods. These methods can converge faster than Gradient Descent, but require more computation and may be less stable.

Gradient Descent is an iterative optimization algorithm, used to find the minimum value for a function. The general idea is to initialize the parameters to random values, and then take small steps in the direction of the "slope" at each iteration. Gradient descent is highly used in supervised learning to minimize the error function and find the optimal values for the parameters. Various extensions have been designed for the gradient descent algorithms. Some of them are discussed below:

Momentum method: This method is used to accelerate the gradient descent algorithm by taking into consideration the exponentially weighted average of the gradients. Using averages makes the

algorithm converge towards the minima in a faster way, as the gradients towards the uncommon directions are canceled out. The pseudocode for the momentum method is given below.

```
V = 0
for each iteration i:
    compute dW
    V = β V + (1 - β) dW
    W = W - α V
```

V and **dW** are analogous to velocity and acceleration respectively. α is the learning rate, and β is analogous to momentum normally kept at 0.9. Physics interpretation is that the velocity of a ball rolling downhill builds up momentum according to the direction of slope(gradient) of the hill and therefore helps in better arrival of the ball at a minimum value (in our case – at a minimum loss).

RMSprop: RMSprop was proposed by the University of Toronto's Geoffrey Hinton. The intuition is to apply an exponentially weighted average method to the second moment of the gradients (dW2). The pseudocode for this is as follows:

```
S = 0
for each iteration i
    compute dW
    S = β S + (1 - β) dW2
    W = W - α dW/√S + ε
```

Adam Optimization: Adam optimization algorithm incorporates the momentum method and RMSprop, along with bias correction. The pseudocode for this approach is as follows:

```
V = 0
S = 0
for each iteration i
    compute dW
    V = β1 S + (1 - β1) dW
    S = β2 S + (1 - β2) dW2
    V = V/{1 - β1i}
    S = S/{1 - β2i}
    W = W - α V/√S + ε
```

Kingma and Ba, the proposers of Adam, recommended the following values for the hyperparameters.

```
α = 0.001
β1 = 0.9
```

$\beta 2 = 0.999$

$\varepsilon = 10\text{-}8$