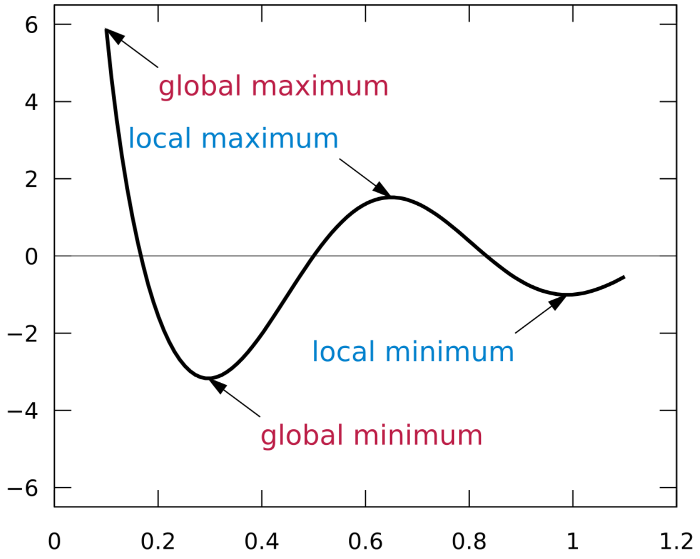
Optimization Techniques

- Optimization algorithms are responsible for reducing losses and provide most accurate results possible.
- The weight is initialized using some initialization strategies and is updated with each epoch according to the equation. The best results are achieved using some optimization strategies or algorithms called Optimizer.

- Some of the techniques that we will be discussing in this article is-

```
* Gradient Descent
* Stochastic Gradient Descent (SGD)
* Mini-Batch Stochastic Gradient Descent (MB – SGD)
* SGD with Momentum
* Nesterov Accelerated Gradient (NAG)
* Adaptive Gradient (AdaGrad)
* AdaDelta
* RMSProp
* Adam
```



1. Gradient Descent or Batch Gradient Descent

- A Gradient Descent is an iterative algorithm, that starts from a random point on the function and traverses down its slope in steps until it reaches lowest point (global minima) of that function.
- This algorithm is apt for cases where optimal points cannot be found by equating the slope of the function to 0. For the function to reach minimum value, the weights should be altered.
- With the help of back propagation, loss is transferred from one layer to another and “weights” parameter are also modified depending on loss so that loss can be minimized.

Point :

1. Use all training Sample for a forward pass and adjust the weights.
2. This makes it computationally intensive. 3. Another drawback is there are chances the iteration values may get stuck at local minima or saddle point and never converge to minima. To obtain the best solution, the must reach global minima. 4. Good For Small training data.

Cost function: $\theta=\theta-\alpha\cdot\nabla J(\theta)$

Advantages:

- Easy computation.
- Easy to implement.
- Easy to understand.

Disadvantages:

- May trap at local minima.
- Weights are changed after calculating gradient on the whole dataset. So, if the dataset is too large than this may take years to converge to the minima.
- Requires large memory to calculate gradient on the whole dataset.

2. Stochastic Gradient Descent

- Stochastic Gradient Descent is an extension of Gradient Descent, where it overcomes some of the disadvantages of Gradient Descent algorithm.
- SGD tries to overcome the disadvantage of computationally intensive by computing the derivative of one point at a time.
- Due to this fact, SGD takes more number of iterations compared to GD to reach minimum and also contains some noise when compared to Gradient Descent.
- As SGD computes derivatives of only 1 point at a time, the time taken to complete one epoch is large compared to Gradient Descent algorithm.

Point :

1. Use One (Randomly Picked) Sample for a forward pass and adjust the weights.
2. Good when training set is very big and we dont want too much computation.

cost function $\theta=\theta-\alpha\cdot\nabla J(\theta;x(i);y(i))$, where $\{x(i) ,y(i)\}$ are the training examples.

Advantages:

- Frequent updates of model parameters hence, converges in less time.
- Requires less memory as no need to store values of loss functions.
- May get new minima’s.

Disadvantages:

- High variance(noisey) in model parameters.

- May shoot even after achieving global minima.
- To get the same convergence as gradient descent needs to slowly reduce the value of learning rate.

3. Mini Batch — Stochastic Gradient Descent

- MB-SGD is an extension of SGD algorithm. It overcomes the time-consuming complexity of SGD by taking a batch of points / subset of points from dataset to compute derivative.
- It's best among all the variations of gradient descent algorithms. It is an improvement on both SGD and standard gradient descent. It updates the model parameters after every batch. So, the dataset is divided into various batches and after every batch, the parameters are updated.
- This is a mixture of both stochastic and batch gradient descent.
- The training set is divided into multiple groups called batches. Each batch has a number of training samples in it.
- At a time a single batch is passed through the network which computes the loss of every sample in the batch and uses their average to update the parameters of the neural network.
- For example, say the training set has 100 training examples which is divided into 5 batches with each batch containing 20 training examples. This means that the equation in figure2 will be iterated over 5 times (number of batches).

Point:

1. Use a Batch Of (Randomly Picked) Sample for a forward pass and adjust the weights.
2. It is observed that the derivative of loss function of MB-SGD is similar to the loss function of GD after some iterations. But the number iterations to achieve minima in MB-SGD is large compared to GD and is computationally expensive. The update of weights in much noisier because the derivative is not always towards minima.

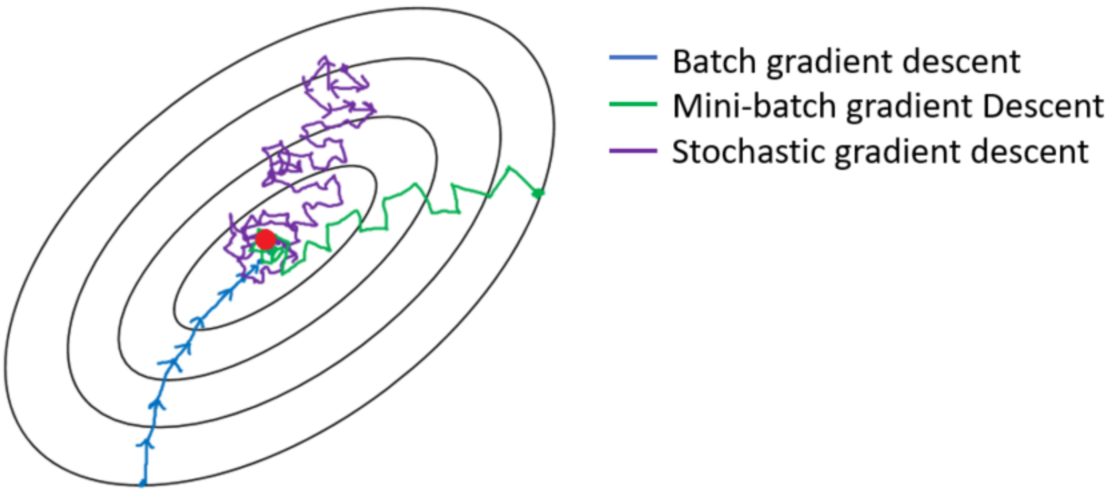
$\theta=\theta-\alpha\cdot\nabla J(\theta; B(i))$, where $\{B(i)\}$ are the batches of training examples.

Advantages:

- Frequently updates the model parameters and also has less variance.
- Requires medium amount of memory.
- Easily fits in the memory
- It is computationally efficient
- Benefit from vectorization
- If stuck in local minimums, some noisy steps can lead the way out of them
- Average of the training samples produces stable error gradients and convergence

!!!! This ensures the following advantages of both stochastic and batch gradient descent are used due to which Mini Batch Gradient Descent is most commonly used in practice.

See How in this above three convergence Occure towards minima point



Here We see in SGD Due to frequent updates the steps taken towards the minima are very noisy. This can often lead the gradient descent into other directions. Also, due to noisy steps it may take longer to achieve convergence to the minima of the loss function. to reduce this we can use SGD with Momentum.

4. SGD with Momentum

- Momentum was invented for reducing high variance in SGD and softens the convergence.
- It accelerates the convergence towards the relevant direction and reduces the fluctuation to the irrelevant direction. One more hyperparameter is used in this method known as momentum symbolized by ‘γ’(gamma).
- It is an adaptive optimization algorithm which exponentially uses weighted average gradients over previous iterations to stabilize the convergence, resulting in quicker optimization.
- This is done by adding a fraction (gamma) to the previous iteration values.
- Essentially the momentum term increase when the gradient points are in the same directions and reduce when gradients fluctuate. As a result, the value of loss function converges faster than expected.

$$v_t=\gamma v_{t-1}+\eta\nabla J(w_t)$$

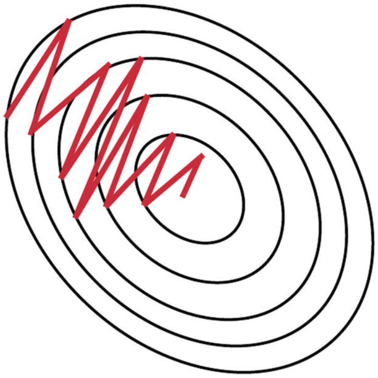
$$w_t=w_{t-1}-v_t$$

Advantages:

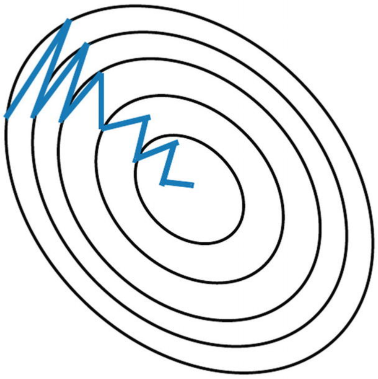
- Reduces the oscillations and high variance of the parameters.
- Converges faster than gradient descent.

Disadvantages:

- One more hyper-parameter is added which needs to be selected manually and accurately.

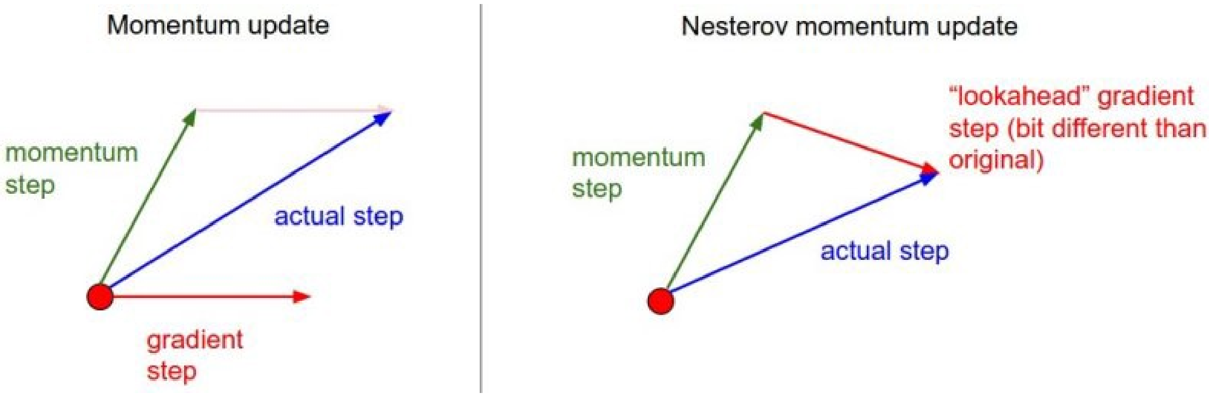


Stochastic Gradient Descent **withhout** Momentum



Stochastic Gradient Descent **with** Momentum

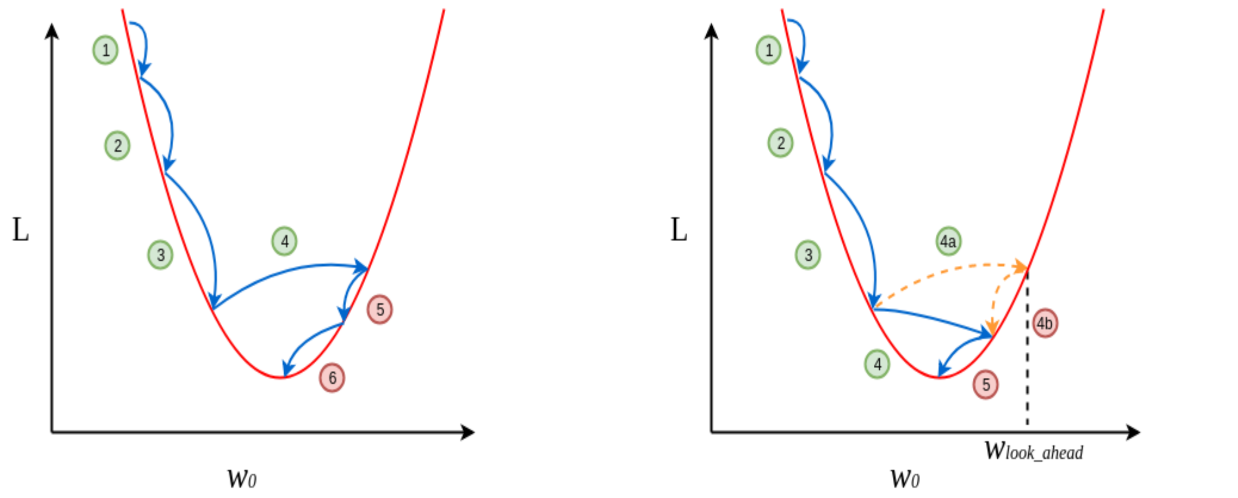
5. Nesterov accelerated gradient(NAG)



- Momentum may be a good method but if the momentum is too high the algorithm may miss the local minima and may continue to rise up. So, to resolve this issue the NAG algorithm was developed.
- Nesterov accelerated gradient (NAG) is a way to give momentum more precision.
- The idea of the NAG algorithm is very similar to SGD with momentum with a slight variant. In the case of SGD with momentum algorithm, the momentum and gradient are computed on previous updated weight.
- Both NAG and SGD with momentum algorithms work equally well and share the same advantages and disadvantages.

$$v_t=\gamma v_{t-1}+\eta\nabla J(w_t - \gamma v_{t-1})$$

$$w_t=w_{t-1}-v_t$$



(a) Momentum-Based Gradient Descent

(b) Nesterov Accelerated Gradient Descent

🟢 $\Longrightarrow \frac{\partial L}{\partial w_0} = \frac{Negative(-)}{Positive(+)}$ 🔴 $\Longrightarrow \frac{\partial L}{\partial w_0} = \frac{Negative(-)}{Negative(-)}$

figure (a) :

- In figure (a), update 1 is positive i.e., the gradient is negative because as w_0 increases L decreases. Even update 2 is positive as well and you can see that the update is slightly larger than update 1 because of momentum.
- By now, you should be convinced that update 3 will be bigger than both update 1 and 2 simply because of momentum and the positive update history.
- Update 4 is where things get interesting. In SGD with Momentum case, due to the positive history, the update overshoots and the descent recovers by doing negative updates.

figure (b) :

- But in NAG's case, every update happens in two steps — first, a partial update, where we get to the look_ahead point and then the final update (see the NAG update rule), see figure (b).
- First 3 updates of NAG are pretty similar to the momentum-based method as both the updates (partial and final) are positive in those cases. But the real difference becomes apparent during update 4.
- As usual, each update happens in two stages, the partial update (4a) is positive, but the final update (4b) would be negative as the calculated gradient at w_lookahead would be negative (convince yourself by observing the graph).
- This negative final update slightly reduces the overall magnitude of the update, still resulting in an overshoot but a smaller one when compared to the vanilla momentum-based gradient descent. And that my friend, is how NAG helps us in reducing the overshoots, i.e. making us take shorter U-turns.

Advantages:

- Does not miss the local minima.
- Slows if minima's are occurring.

Disadvantages:

- Still, the hyperparameter needs to be selected manually.

Point :

- By using NAG technique, we are now able to adapt error function with the help of previous and future values and thus eventually speed up the convergence. Now, in the next techniques we will try to adapt alter or vary the individual parameters depending on the importance factor it plays in each case.