**1.Explain in detail about Momentum based Gradient Descent?**

**What is Momentum and Gradient Descent?**

**Gradient descent** is an optimization algorithm that follows the negative gradient of an objective function in order to locate the minimum of the function.

A problem with gradient descent is that it can bounce around the search space on optimization problems that have large amounts of curvature or noisy gradients, and it can get stuck in flat spots in the search space that have no gradient.

**Momentum** is an extension to the gradient descent optimization algorithm that allows the search to build inertia in a direction in the search space and overcome the oscillations of noisy gradients and coast across flat spots of the search space.

**Update rule for momentum based gradient descent**

Update t=γ·updatet−1+η∇wt

wt+1=w t − update t

In addition to the current update, also look at the history of updates.

update t=γ·updatet−1+η∇wt

wt+1=w t − update t

update0=0

update1=γ·update0+η∇w1=η∇w1

update2=γ·update1+η∇w2=γ·η∇w1+η∇w2

update3=γ·update2+η∇w3=γ(γ·η∇w1+η∇w2) +η∇w3

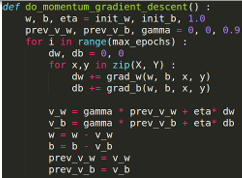
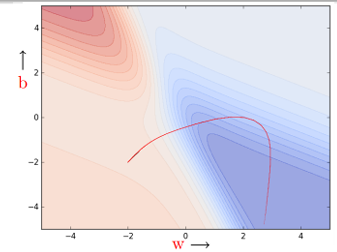
=γ·update2+η∇w3=γ2·η∇w1+γ·η∇w2+η∇w3 update4=γ·update3+η∇w4=γ3·η∇w1+γ2·η∇w2+γ·η∇w3+η∇w4

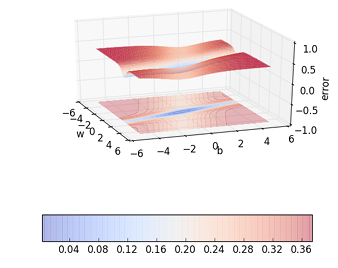
.

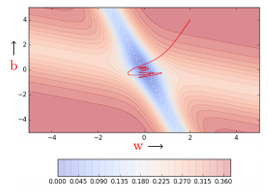
.

.

update t=γ·updatet−1+η∇wt=γt−1·η∇w1+γt−2·η∇w1+...+η∇w t

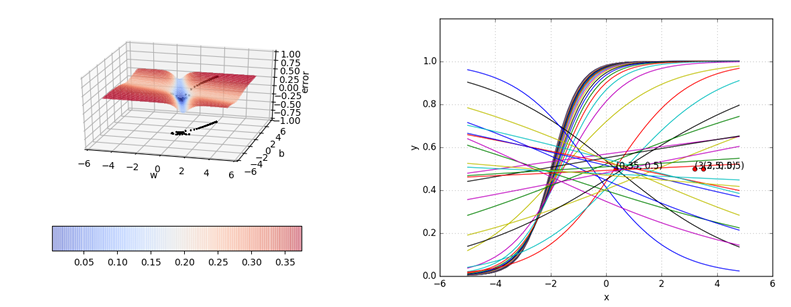
 

* In this case, the error is high on either side of the minima valley 
* Could momentum be detrimental in such cases... let’s see....
* Momentum based gradient descent oscillates in and out of the minima valley as the momentum carries it out of the valley.
* Takes a lot of u- turns before finally converging.



* Despite these u- turns it still converges faster than vanilla gradient descent
* After 100 iterations momentum based method has reached an error of 0.00001 whereas vanilla gradient descent is still stuck at an error of 0.36

Let’s look at a 3d visualization and a different geometric perspective of the same thing...



**Advantages of Momentum based Gradient Descent**

1. **Faster Convergence:**Momentum helps the optimization algorithm to accumulate speed in the direction of the gradient, enabling it to overcome small oscillations and converge faster to the minimum or optimal solution.
2. **Improved Stability:** By incorporating momentum, the optimization process becomes more stable. This is especially beneficial when dealing with noisy or ill-conditioned gradients, as it helps the algorithm continue in the right direction despite the noise.
3. **Escape from Local Minima:** Momentum helps the optimization algorithm to escape local minima or shallow regions in the loss landscape. The accumulated momentum allows the algorithm to traverse flatter regions and reach a more favorable area of the parameter space.
4. **Less Oscillation:** Traditional gradient descent methods might oscillate when approaching the minimum, especially in regions with high curvature. Momentum helps to smooth out these oscillations, leading to more consistent and predictable updates.

**Disadvantages of Momentum based Gradient Descent**

1. **Overshooting and Oscillations:** The momentum term can lead to overshooting the minimum, especially in cases where the optimization algorithm accumulates too much momentum. This may cause oscillations around the optimal solution, making it challenging to converge or converge slowly.
2. **Sensitivity to Hyperparameters:** The performance of momentum-based gradient descent is sensitive to the choice of hyperparameters, particularly the momentum coefficient. Selecting an inappropriate value for the momentum term may result in suboptimal convergence or even divergence.
3. **Dependency on Initialization:** Momentum-based methods can be sensitive to the choice of initial conditions. Poor initialization or a bad choice of starting point might result in the algorithm getting stuck in a suboptimal solution.
4. **Difficulty in tuning:** While the momentum term is a hyperparameter that can be tuned, finding the right value may require experimentation and tuning. It adds an additional dimension to hyperparameter tuning, which can increase the complexity of the optimization process.

**2.Explain in detail about Nesterov Accelerated Gradient Descent?**

Nesterov Accelerated Gradient Descent (NAG), often simply referred to as Nesterov Momentum, is a variant of the traditional momentum-based gradient descent optimization algorithm. It was introduced by Yurii Nesterov in 1983 and has become a popular choice in deep learning applications due to its improved convergence properties compared to standard momentum-based methods.

In Nesterov Accelerated Gradient Descent, the key idea is to modify the momentum-based update by incorporating information from the future estimated position of the parameters.

Instead of evaluating the gradient at the current position of the parameters, the algorithm evaluates the gradient at a point slightly ahead in the direction of the momentum. This helps to anticipate the upcoming movement and adjust the update accordingly.

The update rule for Nesterov Accelerated Gradient Descent is given by:

*vt*+1​=*γ* v t​+*η* ∇*f* (θt​− *γ* v t​)

*θt*+1​= θt − *vt*+1

where:

θtis the current parameter vector,

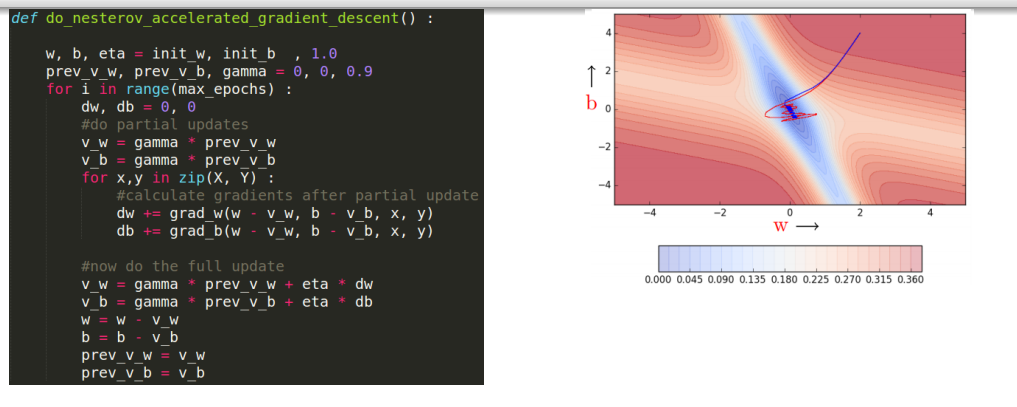
∇*f*(θt) is the gradient of the loss function with respect to the parameters at the current position,

v t is the momentum term at time step t

*γ* is the momentum coefficient,

*η* is the learning rate.

The key modification in Nesterov Accelerated Gradient Descent is that the gradient is evaluated at θt​− *γ* v t, the anticipated future position of the parameters. This "look-ahead" correction helps in reducing the oscillations and provides better convergence properties compared to traditional momentum-based methods.

****

In summary, Nesterov Accelerated Gradient Descent is an enhancement of momentum-based optimization algorithms, introducing a correction term to the gradient evaluation that improves convergence and stability during the optimization process.

**Advantages of Nesterov Accelerated Gradient Descent**

1. **Faster Convergence:**Nesterov Accelerated Gradient Descent often converges faster than standard momentum-based methods. The "look-ahead" correction helps the algorithm anticipate the direction of the gradient, allowing it to converge more efficiently towards the optimum.
2. **Reduced Oscillation:** By incorporating the information from the future estimated position of the parameters, NAG tends to exhibit fewer oscillations during the optimization process. This results in a smoother trajectory towards the optimal solution, contributing to improved stability.
3. **Improved Accuracy:** The anticipatory correction in Nesterov Accelerated Gradient Descent helps reduce the overshooting problem associated with traditional momentum methods. This can lead to more accurate updates, allowing the algorithm to approach the optimal solution more precisely.
4. **Better Handling of High Curvature:** Nesterov Accelerated Gradient Descent is known to perform well in scenarios with high curvature in the optimization landscape. The look-ahead correction helps the algorithm navigate through such regions more effectively, preventing it from overshooting and oscillating excessively.
5. **Consistent Performance Acros Learning Rates:** NAG is often more robust to the choice of learning rates compared to standard momentum-based methods. This can simplify the hyperparameter tuning process and make the algorithm more user-friendly.

**Disadvantages of Nesterov Accelerated Gradient Descent**

1. **Sensitivity to Hyperparameters:** NAG involves tuning hyperparameters such as the learning rate and the momentum coefficient. Selecting inappropriate values for these hyperparameters can impact the algorithm's performance, and finding the right combination may require experimentation.
2. **Complexity in implementation:** Compared to standard gradient descent or traditional momentum methods, Nesterov Accelerated Gradient Descent has a more complex update rule due to the lookahead correction. This complexity may make the implementation less straightforward, and incorrect implementations can lead to unexpected behavior.
3. **Limited Performance Improvement in Some Cases:** While NAG often converges faster in practice, the improvement may not be significant in all scenarios. In certain optimization landscapes, especially those with simple structures, the benefits of the lookahead correction may not be as pronounced, and the additional complexity might not be justified.
4. **Computational Cost:** The lookahead correction involves an additional computation step, making the algorithm computationally more expensive compared to traditional momentum-based methods. This can be a consideration in resource-constrained environments or when training large-scale models.
5. **Dependency on Smoothness of the Objective Function:** NAG assumes a certain level of smoothness in the objective function. In cases where the objective function is highly non-smooth or has discontinuities, the lookahead correction might not provide as much benefit, and simpler optimization methods may be more appropriate.