Optimizers

Optimizers

Optimizers are the algorithms that update the weights during back propogation

Gradient Descent

The basic algorithm that is used to minimize the error with respect to the weights of the neural network. The learning rate determines the step size of the update used to reach the local minima of the loss function.

Reducing Error by Gradient Descent

We know that error is the difference between targeted value and predicted value.

$$E = y - y_{pred}$$

Gradient descent is an iterative optimization algorithm to find the global minimum of a function. For applying Gradient descent we will convert our error equation into mean squared error equation shown below.

$$E = \frac{1}{n}\sum_{i=1}^{n}(y - y_{pred})^2$$

$$E = \frac{1}{n}\sum_{i=1}^{n}(y - XW)^2$$

Above equation is a loss function for Regression problem and now we are applying Gradient descent function to find the minimum value of loss function. The first step is to take the partial derivative OR gradient with respect to W.

$$\frac{\partial E}{\partial W} = \frac{1}{n}\sum_{i=1}^{n} 2(y - XW)(-X)$$

Rearranging above equation

$$\frac{\partial E}{\partial W} = \frac{-2}{n}\sum_{i=1}^{n} X(y - XW)$$

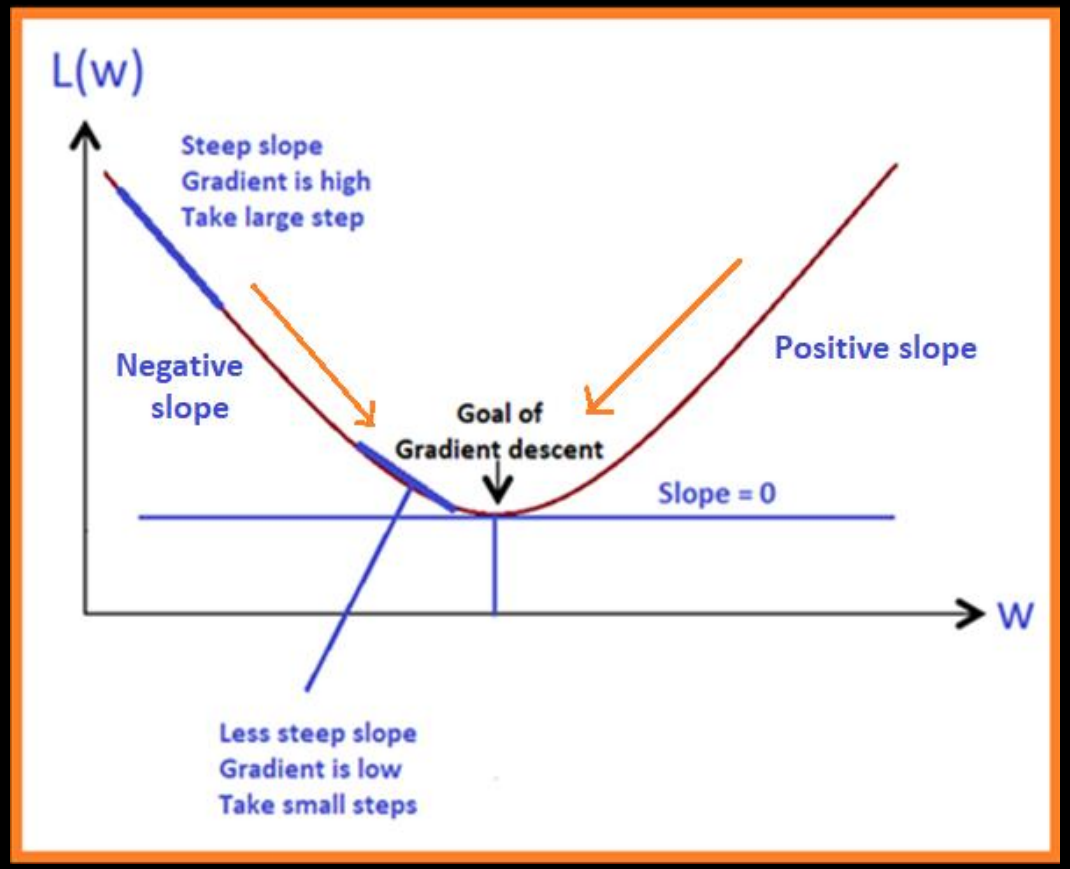
The term $\frac{\partial E}{\partial W}$ is called the gradient.

Now the third important step is to update the parameters of regression equation i.e weights by using the following equation.

$$W = W - \alpha * gradient$$

In above equation α is called the learning rate. Learning rate controls how much the value of W changes with each step. The value of α should be small such as 0.01 for good training of the model.

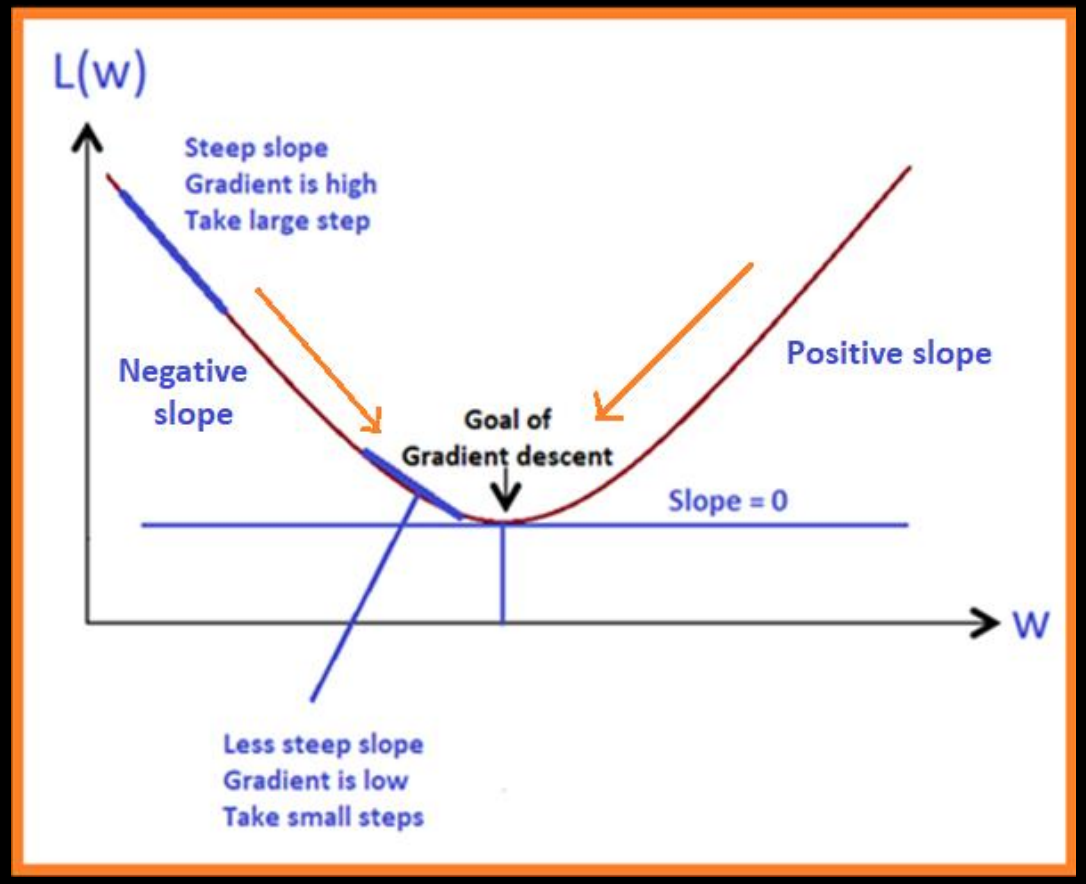
Pictorial Explanation Of Gradient Descent



$$-\alpha \frac{\partial E}{\partial w}$$

Negative sign in the updating term indicates that we move in the opposite direction to gradient (slope)

Pictorial Explanation Of Gradient Descent



Positive slope means that when we increase w, loss increases, therefore to decrease loss we have to move left to decrease weight.

Negative slope means that when we increase w, loss decreases, therefore to decrease loss we have to move right to increase weight.

Epoch, Batch and Iteration

Epoch

An epoch is one complete pass of training data through the training model. If we set epoch = 10, it means we pass the whole training samples 10 times through the training model.

Batch

In deep learning, the data is large, therefore, We use batches to pass the whole training data through the model. Batch size is the number of training samples we pass at particular time for training the model. In stochastic gradient descent algorithm, batch size is fixed to 1.

Iteration

Iterations are the number of batches we need to complete one epoch. Suppose we have 1000 training samples in our dataset, and we choose a batch size of 100. It menas the number of iterations will be 10.

$$Iterations = \frac{Samples\ in\ Epoch}{Batch\ Size}$$

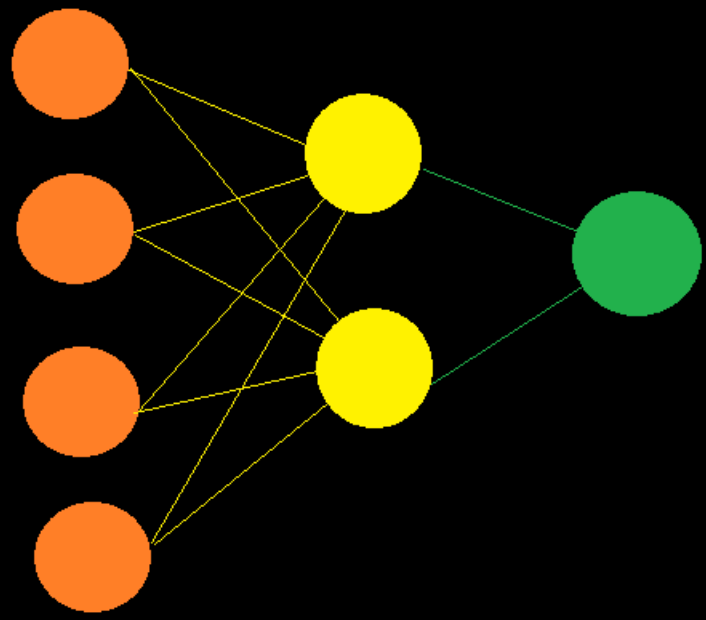
Stochastic Gradient Descent

- Batch Size is one. Take one sample in each iteration and update the weights.
- It works better when all samples are really similar to each other
- The real world data is versatile and samples are not similar.

Stochastic Gradient Descent

Samples	A	B	C	D
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2
5	5.4	3.9	1.7	0.4
6	4.6	3.4	1.4	0.3
7	5.0	3.4	1.5	0.2
8	4.4	2.9	1.4	0.2
9	4.9	3.1	1.5	0.1
10	5.4	3.7	1.5	0.2
11	4.8	3.4	1.6	0.2
12	4.8	3.0	1.4	0.1

4.9	3.0	1.4	0.2



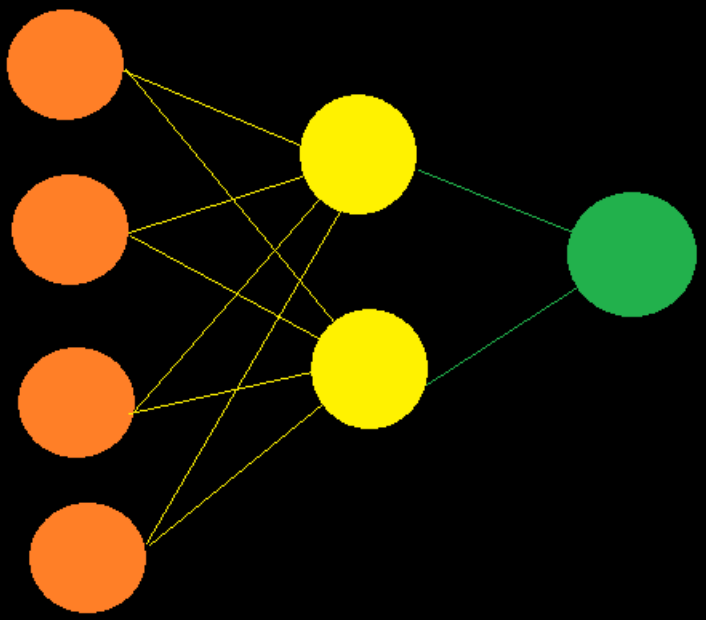
Mini-batch Gradient Descent

- Take n samples (Batch size) in each iteration and update the weights.
- Number of $Iterations\ per\ epoch = \frac{Samples}{Batch\ Size}$

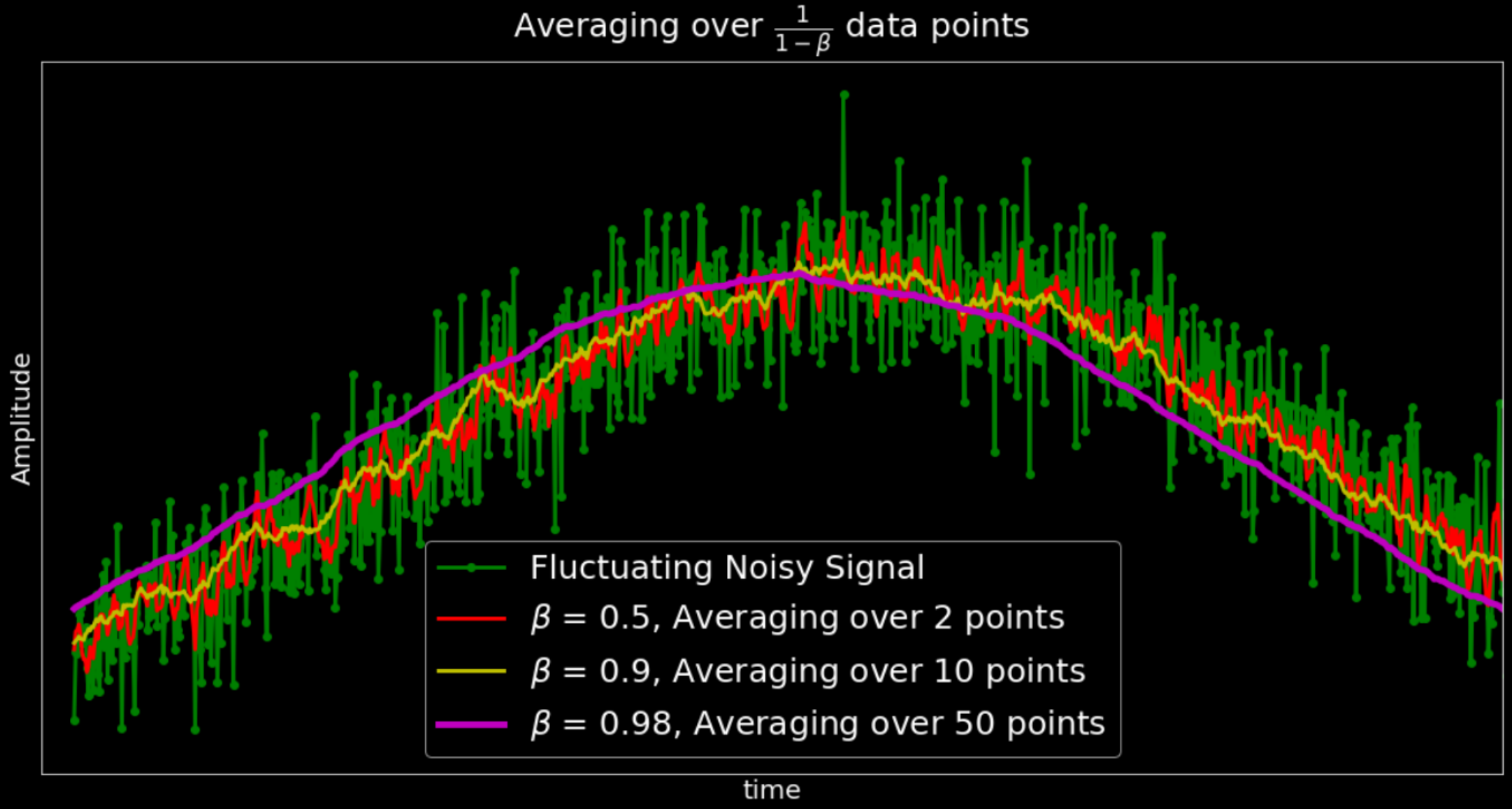
Mini-batch Gradient Descent

Samples	A	B	C	D
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2
5	5.4	3.9	1.7	0.4
6	4.6	3.4	1.4	0.3
7	5.0	3.4	1.5	0.2
8	4.4	2.9	1.4	0.2
9	4.9	3.1	1.5	0.1
10	5.4	3.7	1.5	0.2
11	4.8	3.4	1.6	0.2
12	4.8	3.0	1.4	0.1

4.9	3.0	1.4	0.2
4.7	3.2	1.3	0.2
4.6	3.1	1.5	0.2
5.0	3.6	1.4	0.2

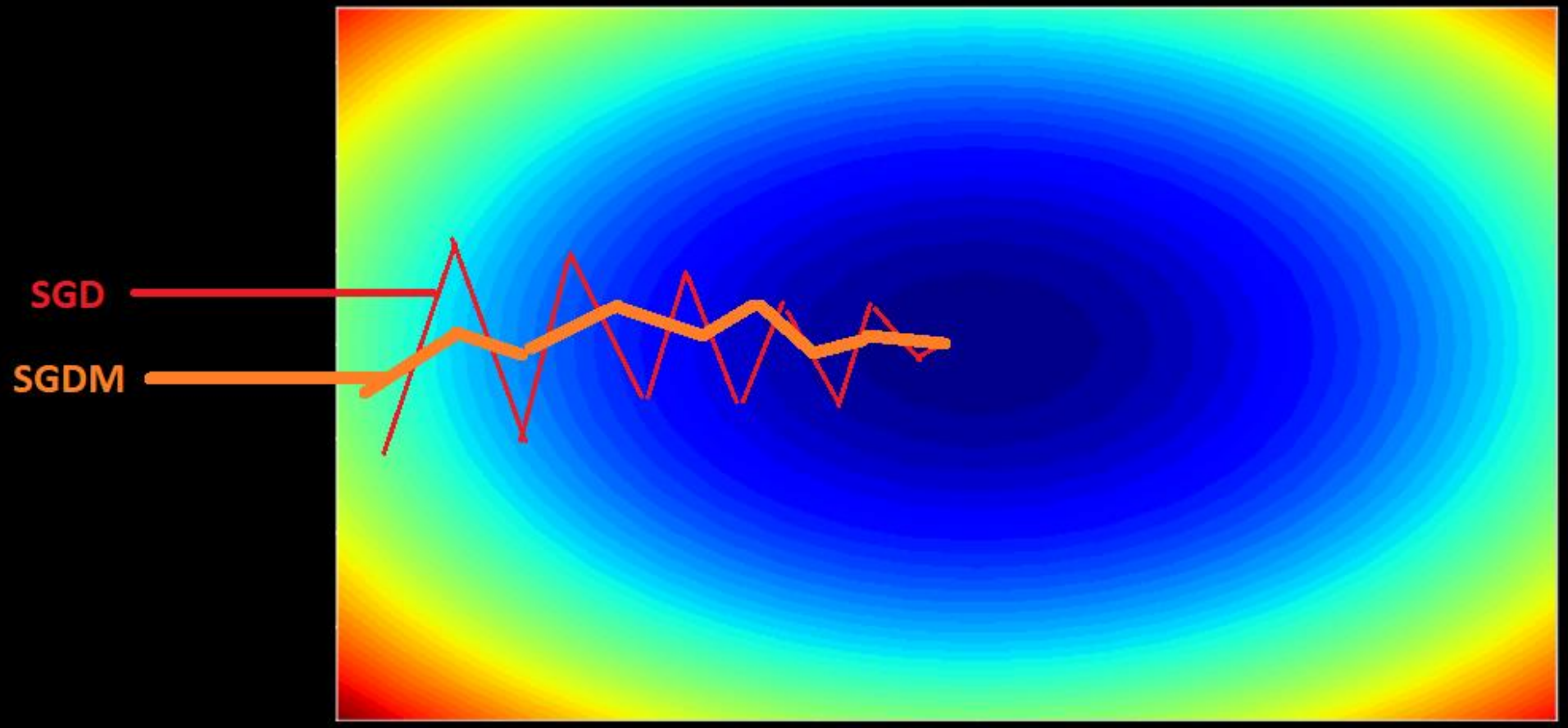


Exponentially Weighted Averages



Momentum

The concept of exponentially moving average is used to reduce the oscillations in the vertical direction and speed up in the horizontal direction. So, momentum means smoothing via weighted average.



Momentum

The concept of exponentially moving average is used to reduce the oscillations in the vertical direction and speed up in the horizontal direction. So, momentum means smoothing via weighted average.

$$v_t = \beta v_{t-1} + (1-\beta)\frac{dE}{d\theta} \qquad \theta = (weights, bias)$$

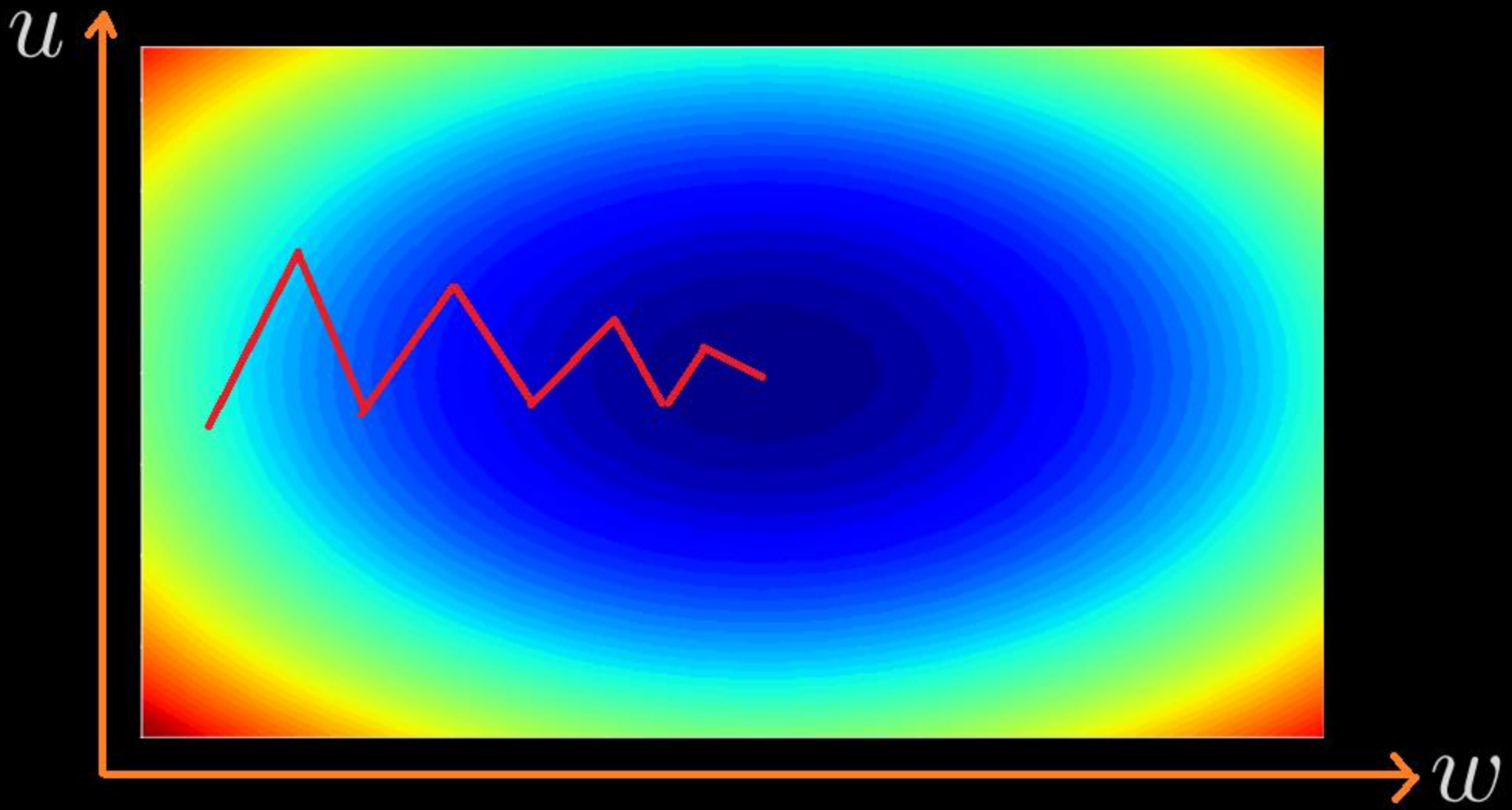
Adding a fraction β of the past update vector, to the new update vector.

$$\theta(new) = \theta(old) - \eta v_t$$

Updated the parameters according to the weighted average of current and previous loss.

RMS Prop

Requirement from RMS Prop is to minimize the oscillation in the vertical direction and increase the speed in the horizontal direction



RMS Prop

$$v_t = \beta v_{t-1} + (1-\beta)(\frac{dE_v}{d\theta})^2$$

$$u_t = \beta u_{t-1} + (1-\beta)(\frac{dE_u}{d\theta})^2$$

$(\frac{dE_v}{d\theta})^2$ and $(\frac{dE_u}{d\theta})^2$ in the above equations are element wise squared.

$(\frac{dE_v}{d\theta})$ is small, therefore $(\frac{dE_v}{d\theta})^2$ is smaller.

$(\frac{dE_u}{d\theta})$ is large, therefore $(\frac{dE_u}{d\theta})^2$ is larger.

Adaptive Moment Estimation (ADAM) Optimization

Adaptive Moment Estimation (ADAM) Optimization

From stochastic gradient descent with momentum, we have

$$v_t = \beta_1 v_{t-1} + (1 - \beta_1)\frac{dE}{d\theta}$$

From RMS Prop, we have

$$s_t = \beta_2 s_{t-1} + (1 - \beta_2)(\frac{dE}{d\theta})^2$$

Adam combines the average $\frac{dE}{d\theta}$ of the history of the gradients and the variance or the energy $(\frac{dE}{d\theta})^2$ of the gradients

$$\theta(new) = \theta(old) - \frac{\eta}{\sqrt{\tilde{s} + \epsilon}}\tilde{v}$$

Adaptive Moment Estimation (ADAM) Optimization

$$\tilde{v} = \frac{v}{1 - \beta_1^t}$$

This is called the bias correction

$$\tilde{s} = \frac{s}{1 - \beta_2^t}$$

Recommended values are

$$\eta = 0.01$$
$$\beta_1 = 0.9$$
$$\beta_2 = 0.99$$
$$\epsilon = 10^{-8}$$

Thank you!

Thank you!