Optimizers in Deep Learning

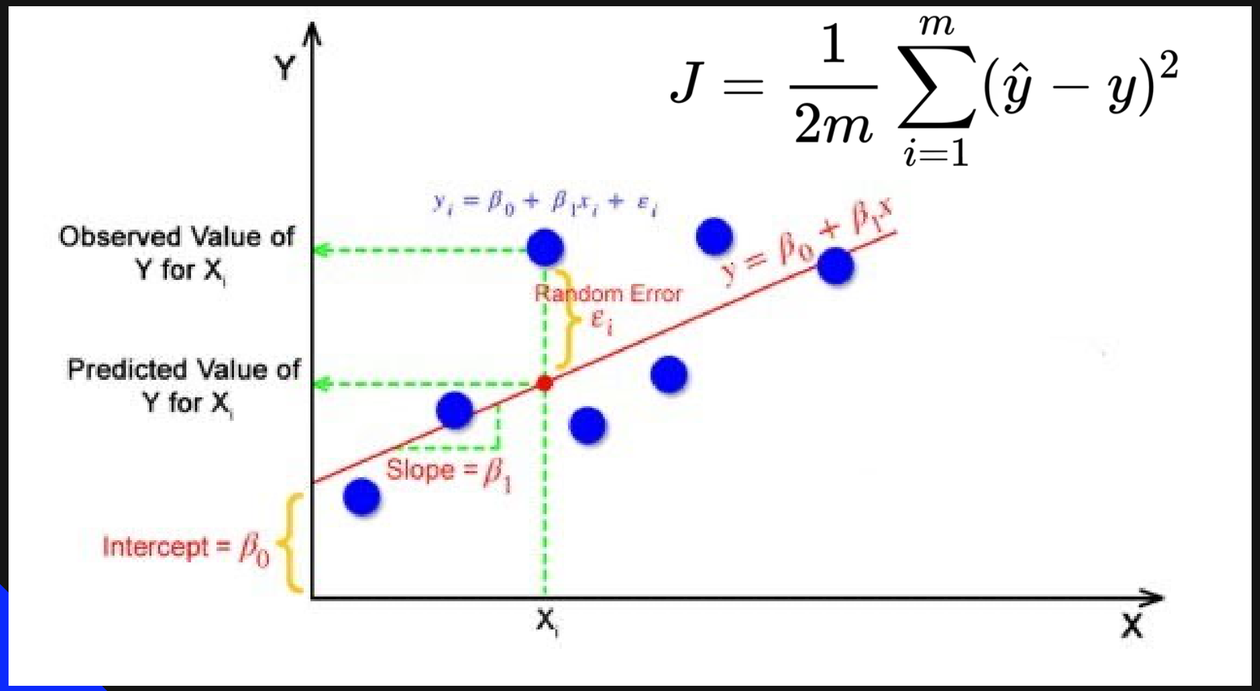
Lets's talk about the loss function

- It measures the error between predicted and actual values in a machine learning model. It is the key idea behind understanding optimizers.

The primary goal of optimizers to minimise this loss function because we want the error of the model to be minimal

For example, In simple linear regression, the loss function J is given by:

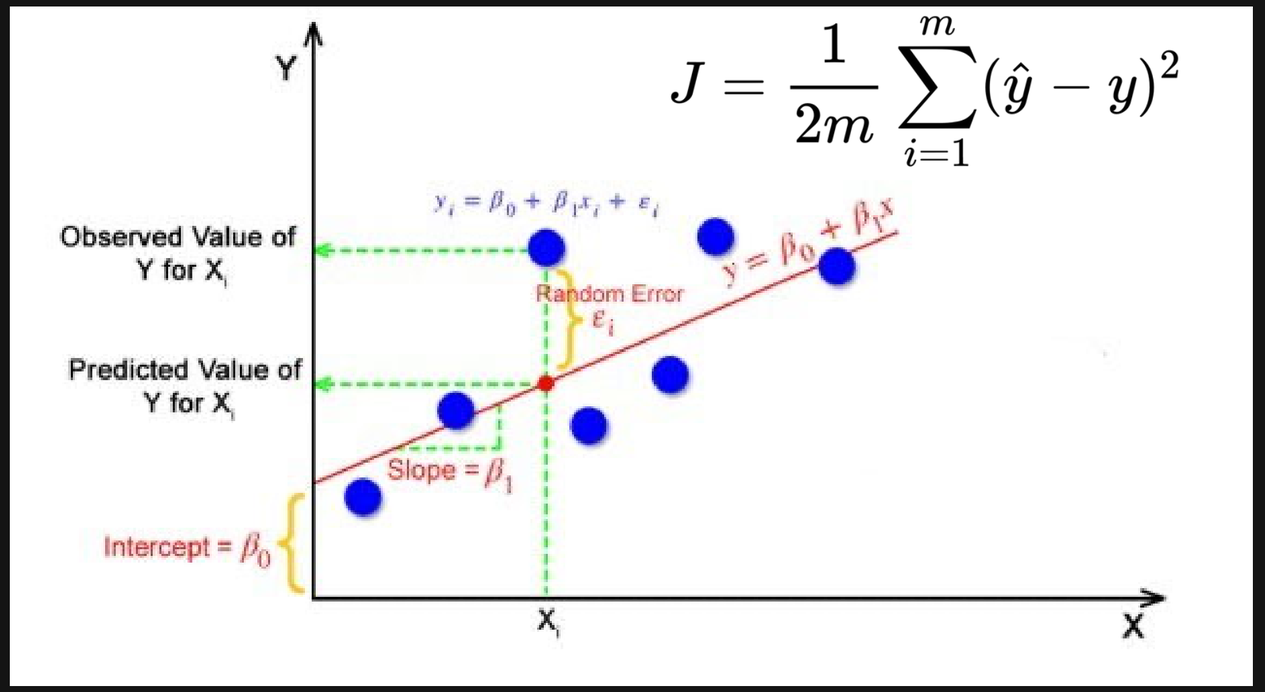
$$J = \frac{1}{2m}\sum_{i=1}^{m}(\hat{y} - y)^2$$



- Optimisation sits at the very core of machine learning models. It aims to lower the risk of errors or loss from these predictions, and improve the accuracy of the model.

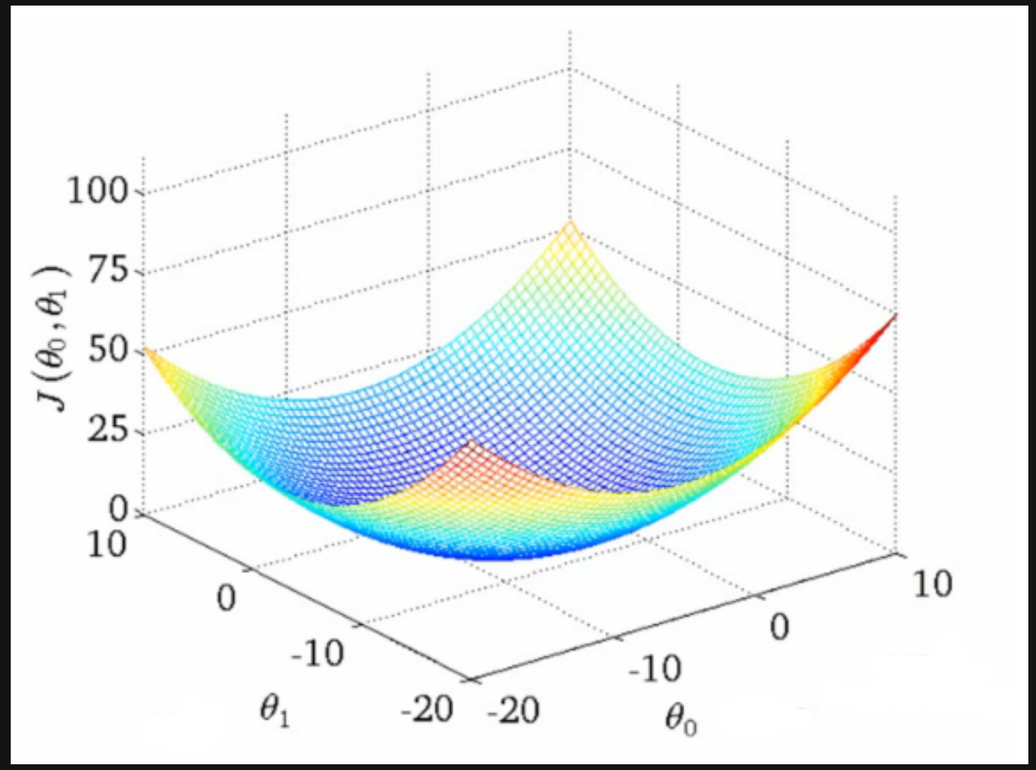
- An optimizer is a function or an algorithm that adjusts the attributes of the neural network, such as weights and learning rates.

Quick Question



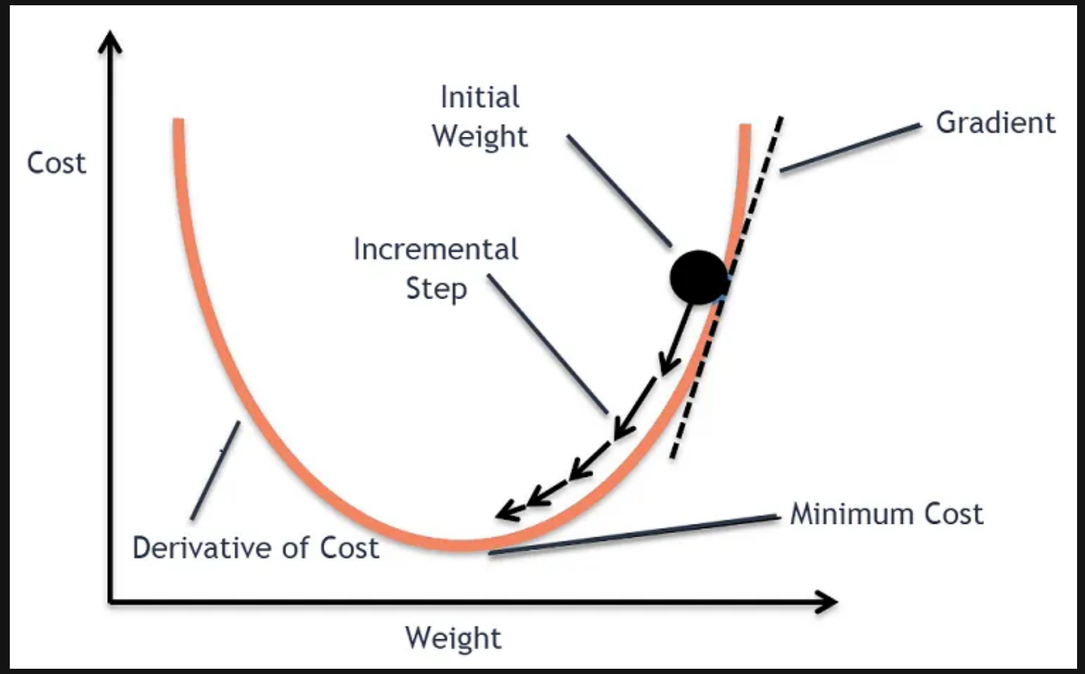
Why does the above expression for loss function **always** have a minima?

Loss Function Plotted (MSE).

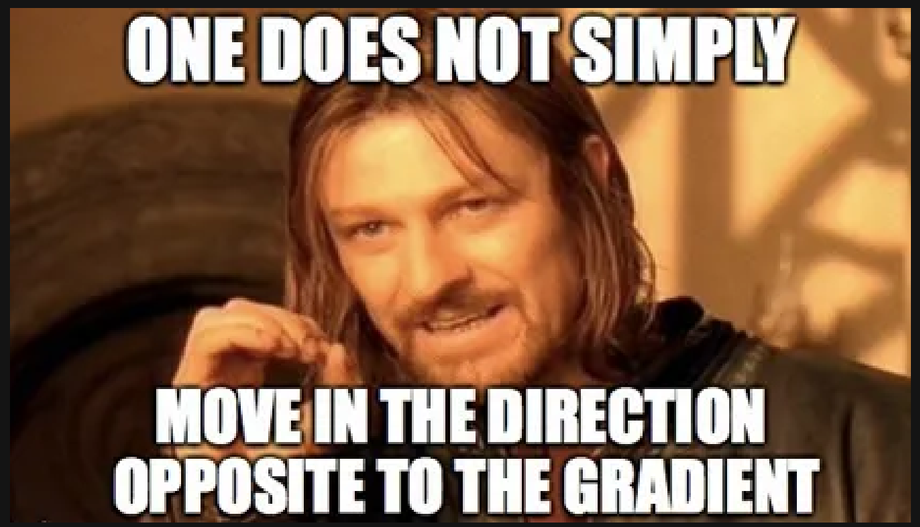


This graph is used to determine θ_0 and θ_1. which minimize the Cost function thereby giving the best fit line .

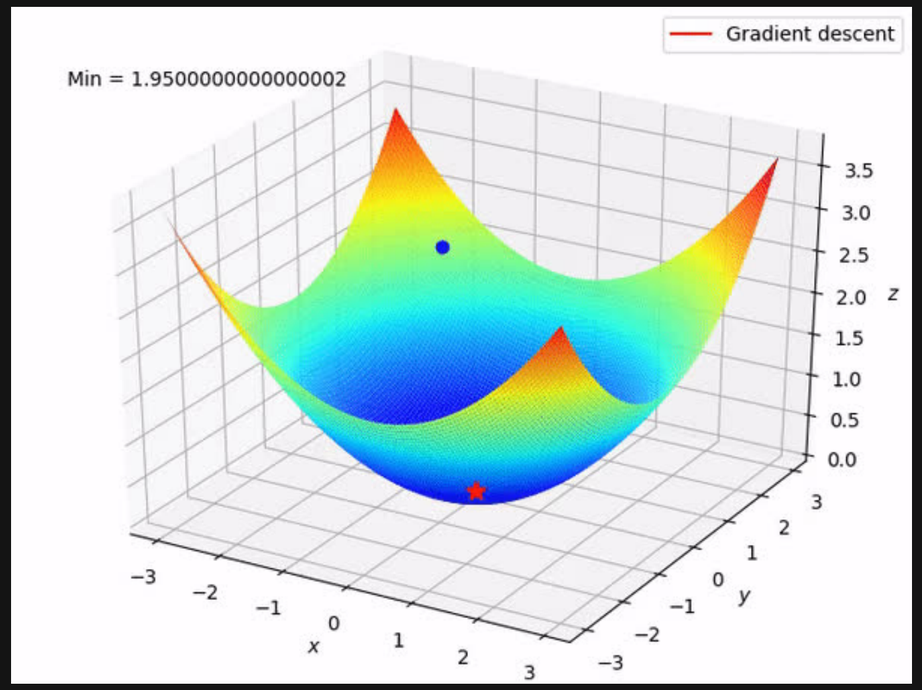
Gradient Descent

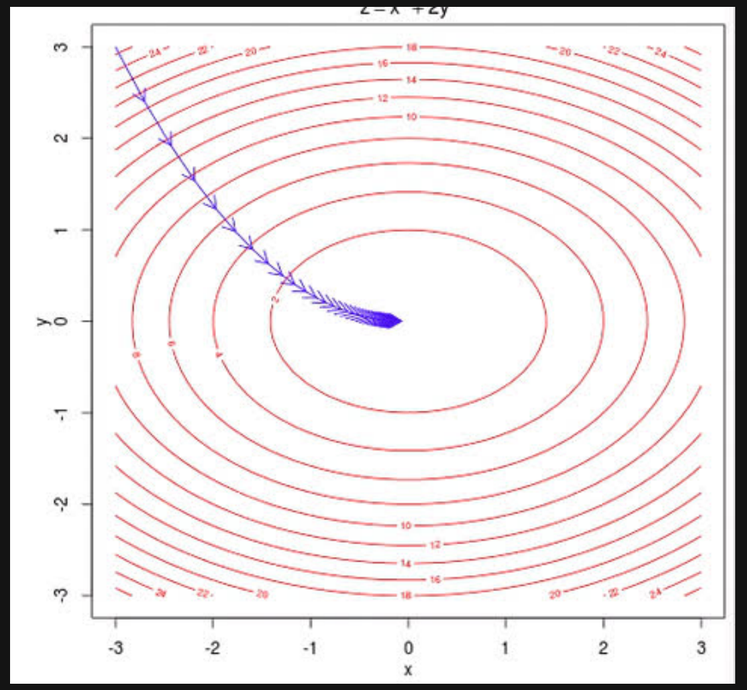


It simply moves in the opposite direction of the slope increase from the current point by the computed amount



The slope of the tangent is the derivative at that point and it will give us a direction to move towards. We make steps down the cost function in the direction with the steepest descent.



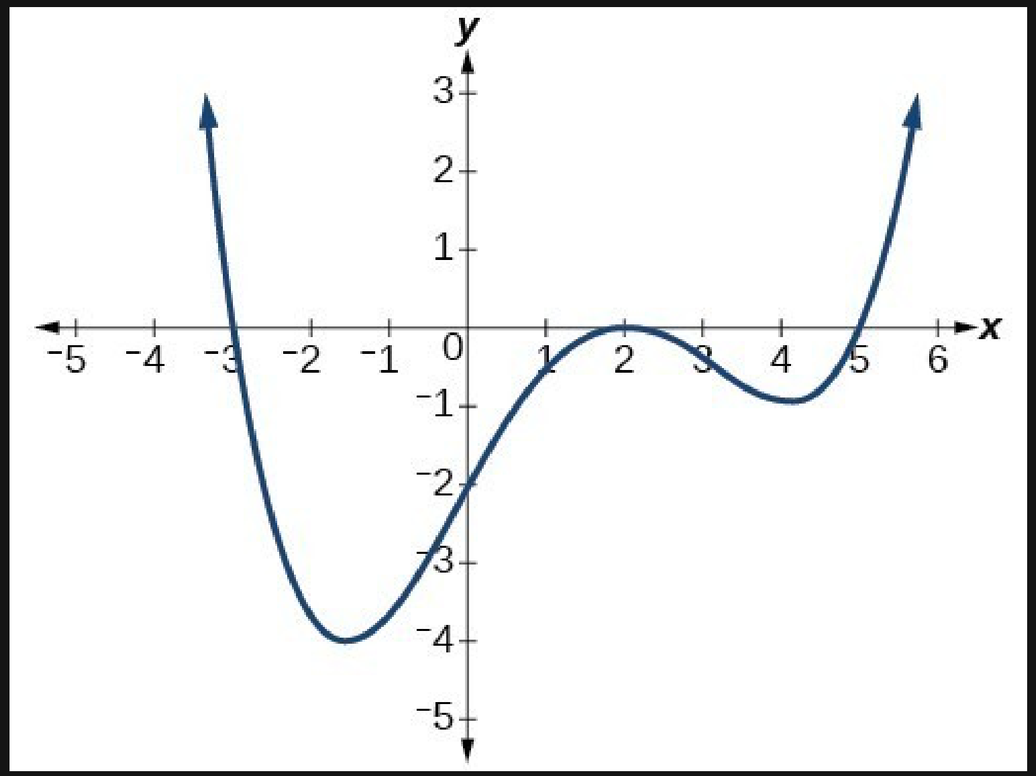


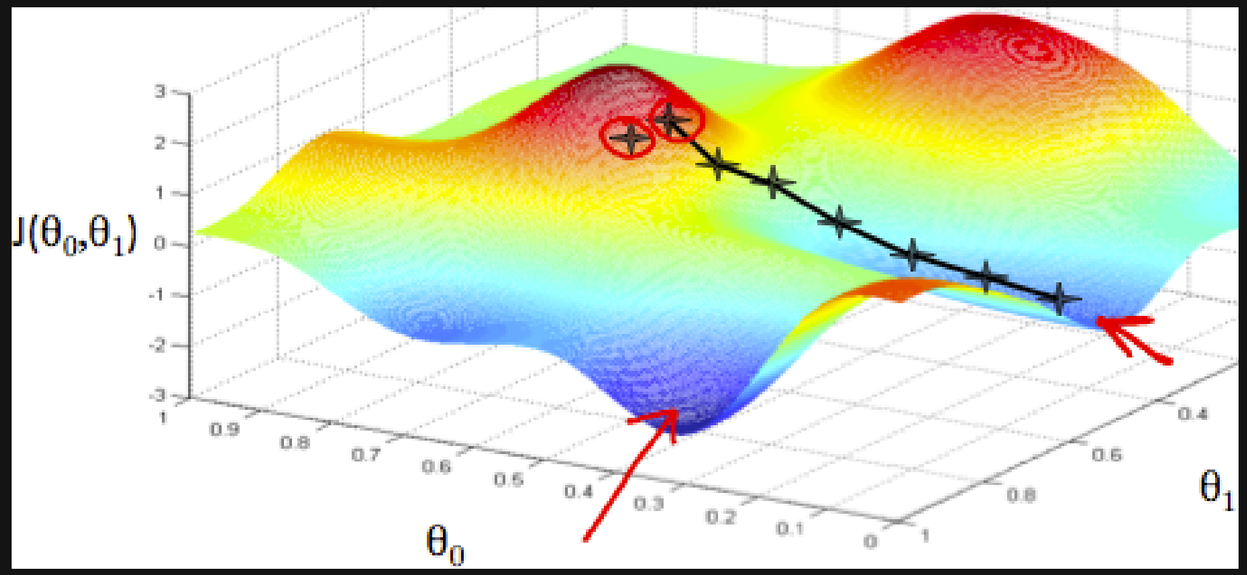
Quick Question

Does the starting point affect the convergence of gradient descent?

$$J = \frac{1}{2m} \sum_{i=1}^{m} (\hat{y} - y)^2$$

HINT: There can be more than one minima





As we move closer and closer to the minima the slope would decrease and the person would take smaller and smaller steps.

$$X = X - lr * \frac{d}{dX} f(X)$$

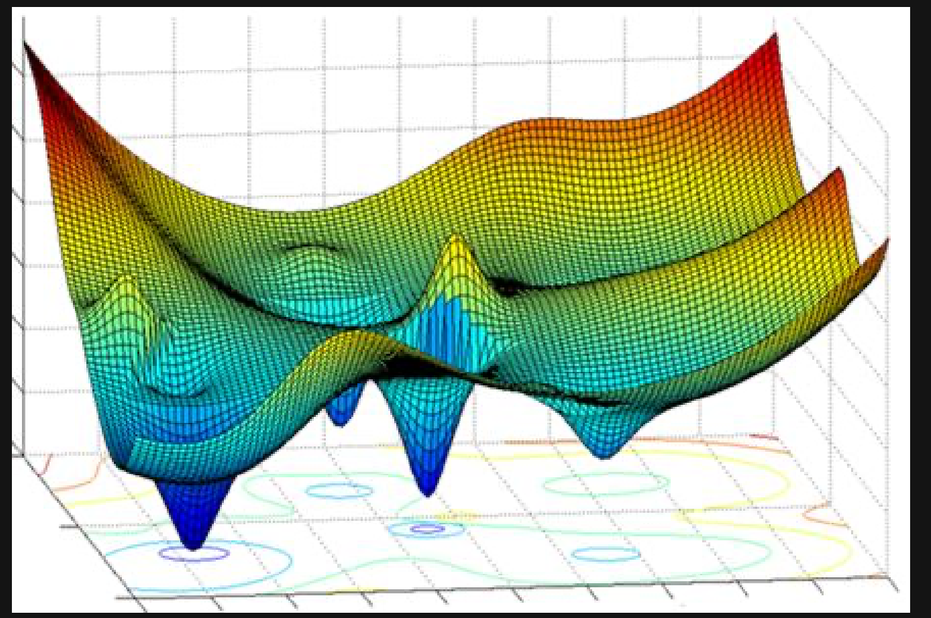
$Where,$

$X \quad = \quad input$

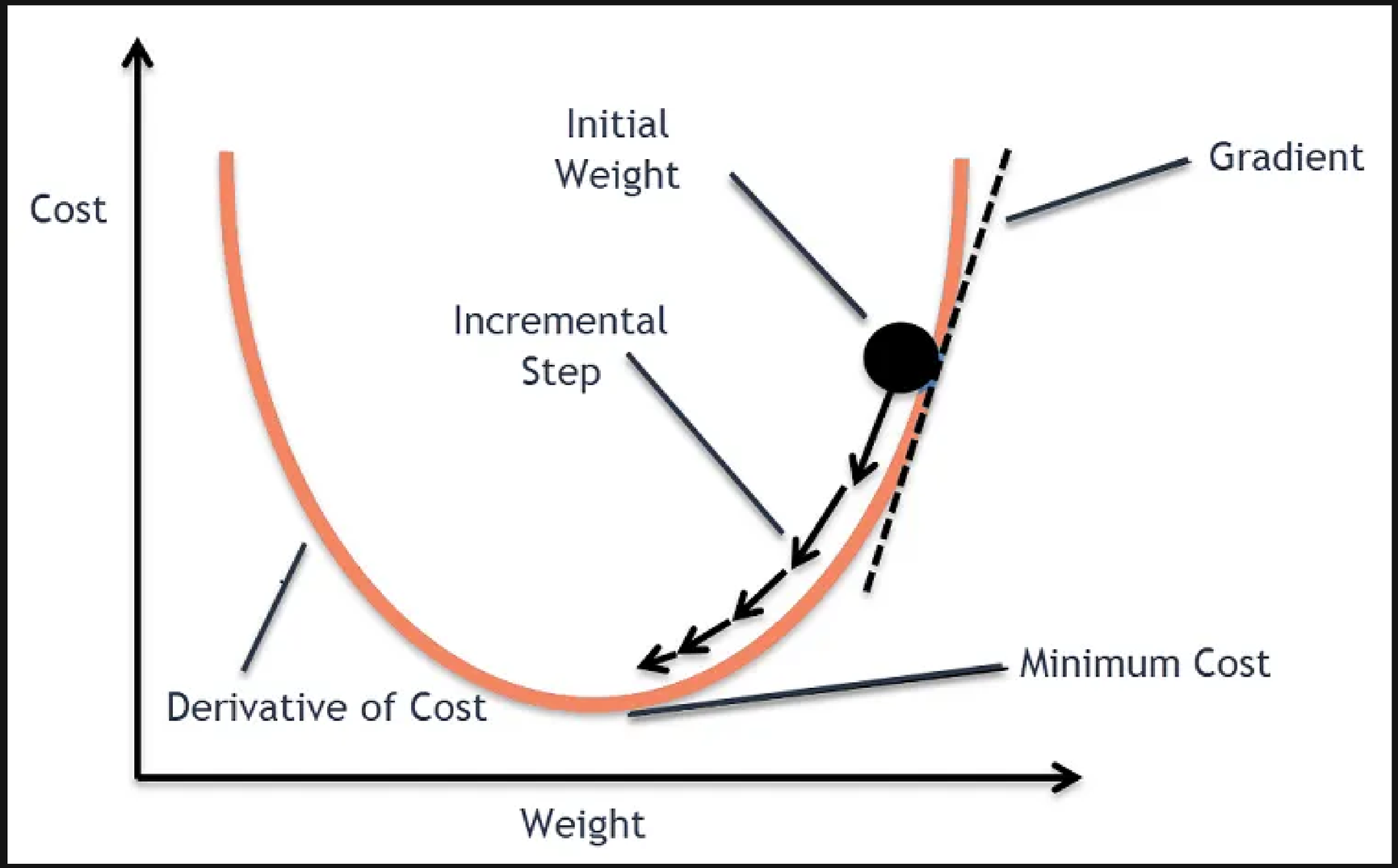
$F(X) \quad = \quad output \ \ based \ on \ X$

$lr \quad = \quad learning \ rate$

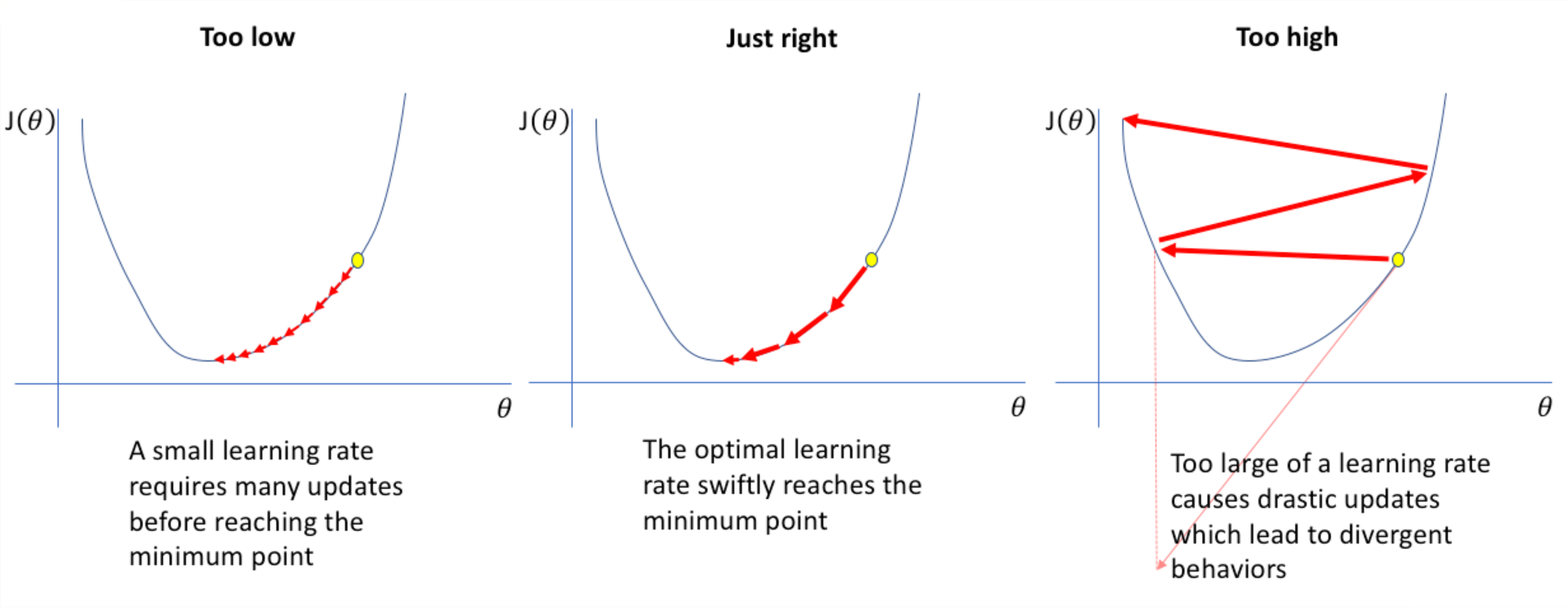
Here X can be any of the parameters of the neural network, i.e., any of the weights of biases.



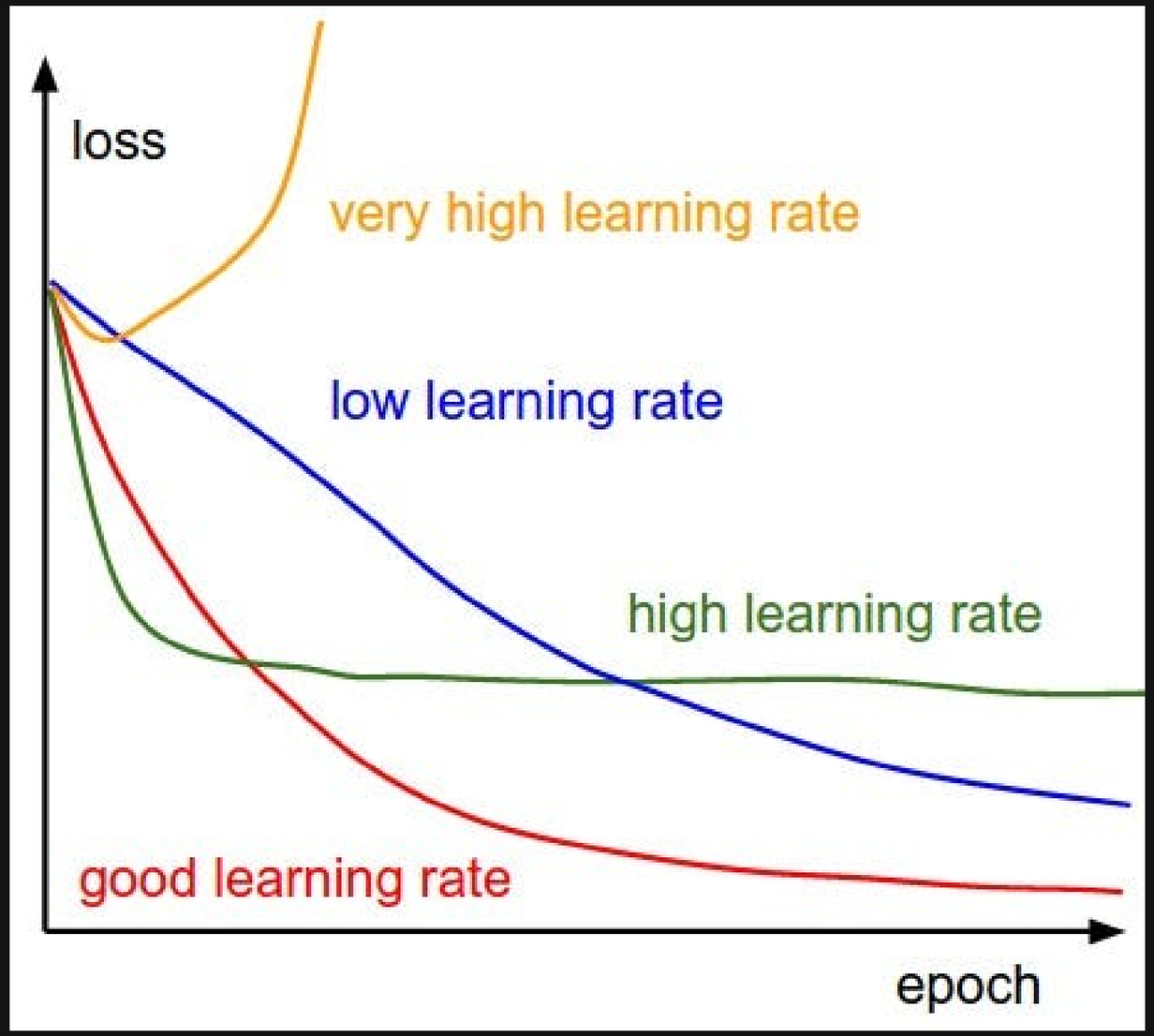
This minimisation is done for every weight and bias to ensure optimal output with minimal loss.


Cost
Initial
Weight
Gradient
Incremental
Step
Derivative of Cost
Minimum Cost
Weight

The Learning Rate α



The Learning Rate α



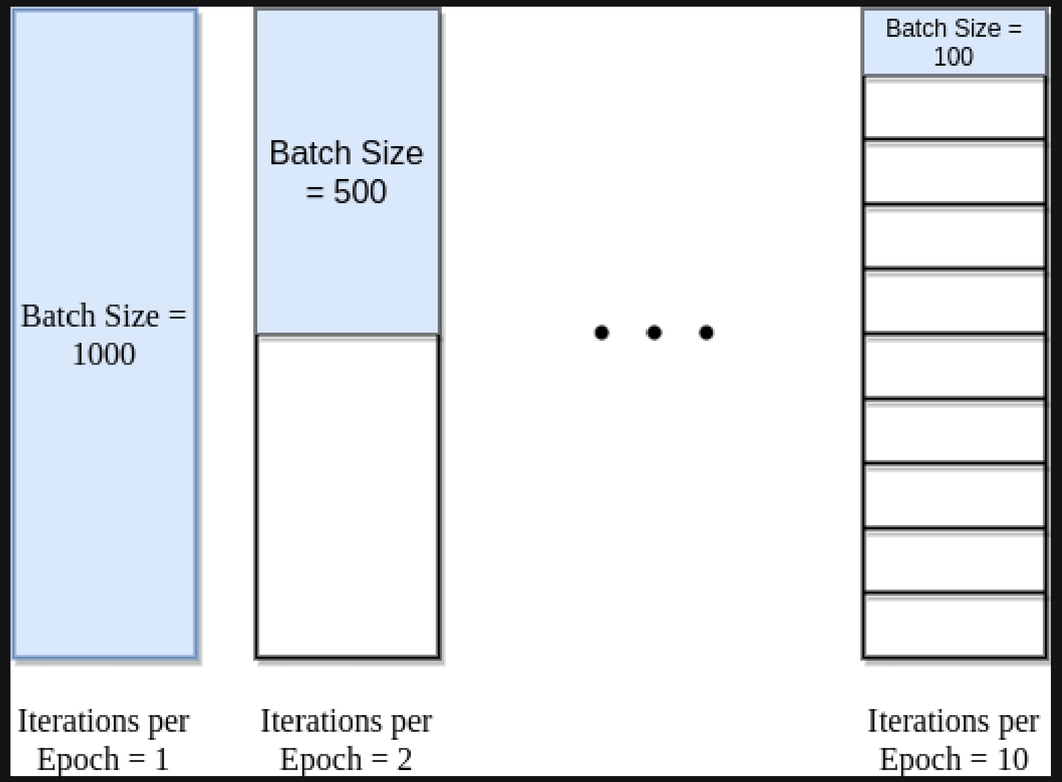
- A good way to determine an optimum α is to plot the cost function with time.
- If it is increasing, learning rate should be reduced and if the convergence is very slow, then it can be increased to achieve quicker convergence

Types of Gradient Descent

What if we have many samples? Will it not be time consuming to apply the gradient formula for each and every sample's weights and bias?

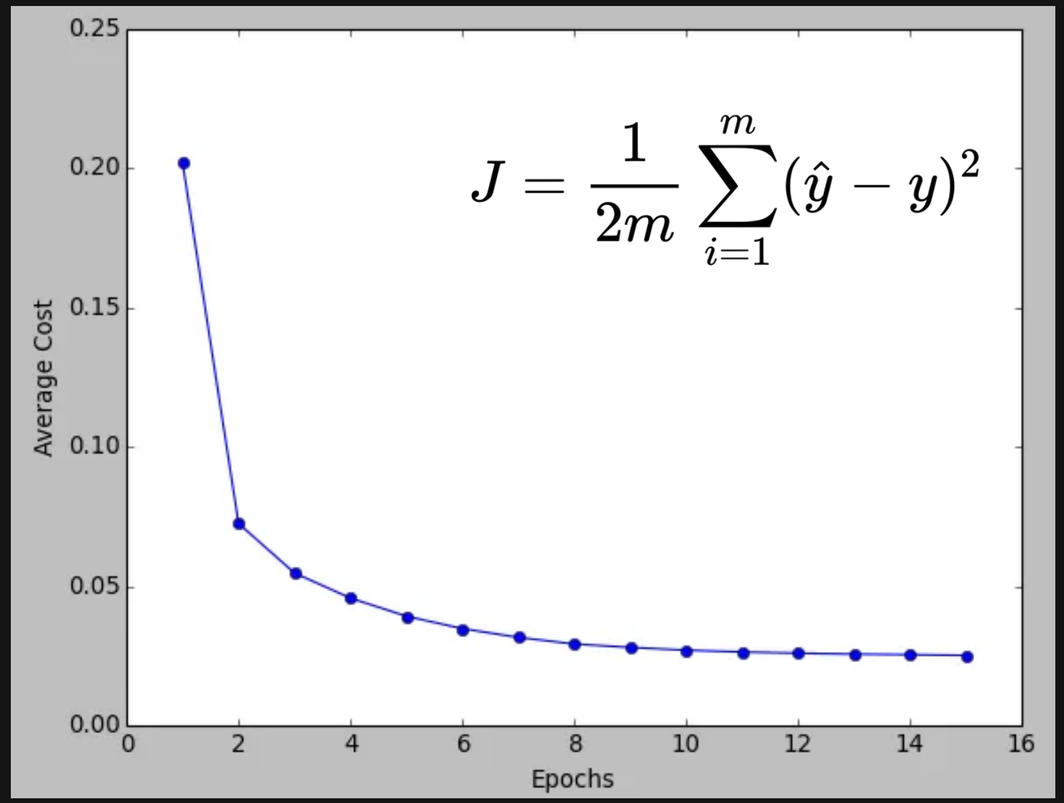
What is an epoch?

One epoch means that each sample in the training dataset has had an opportunity to update the internal model parameters.



Batch Gradient Descent

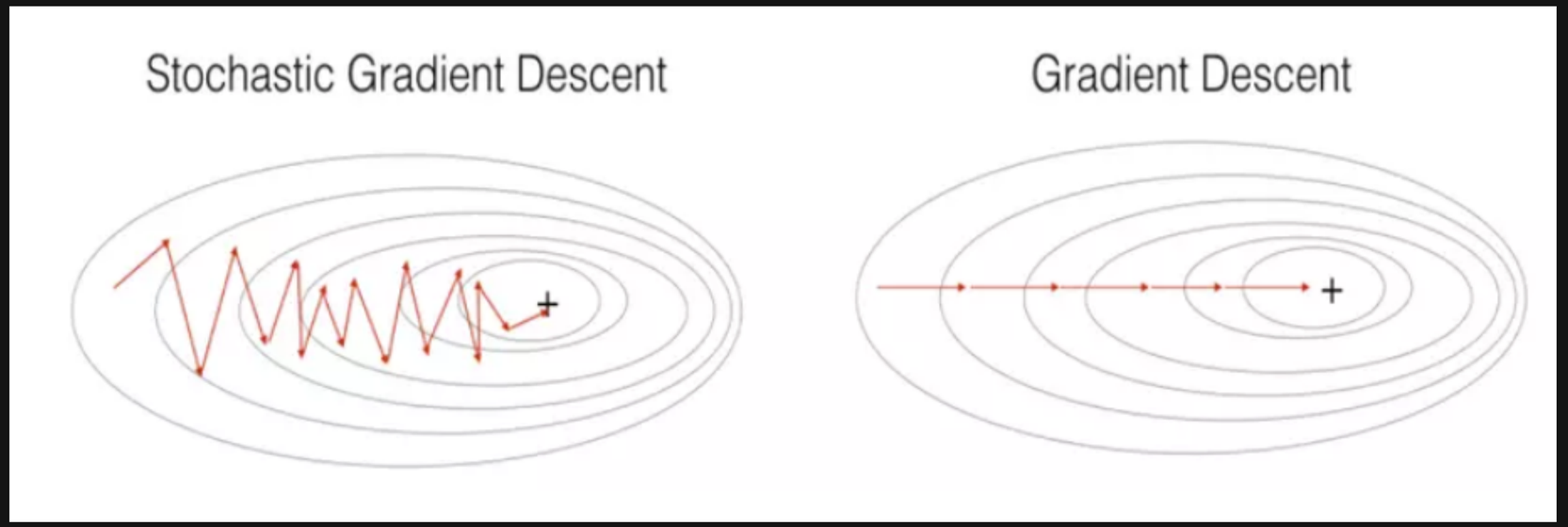
- In Batch Gradient Descent, all the training data is taken into consideration to take a single step. We take the average of the gradients of all the training examples and then use that mean gradient to update our parameters. So that's just one step of gradient descent in one epoch.



Batch Gradient Descent is great for convex or relatively smooth error manifolds.

Stochastic gradient descent

In Stochastic Gradient Descent (SGD), we consider just one sample at a time and update 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.

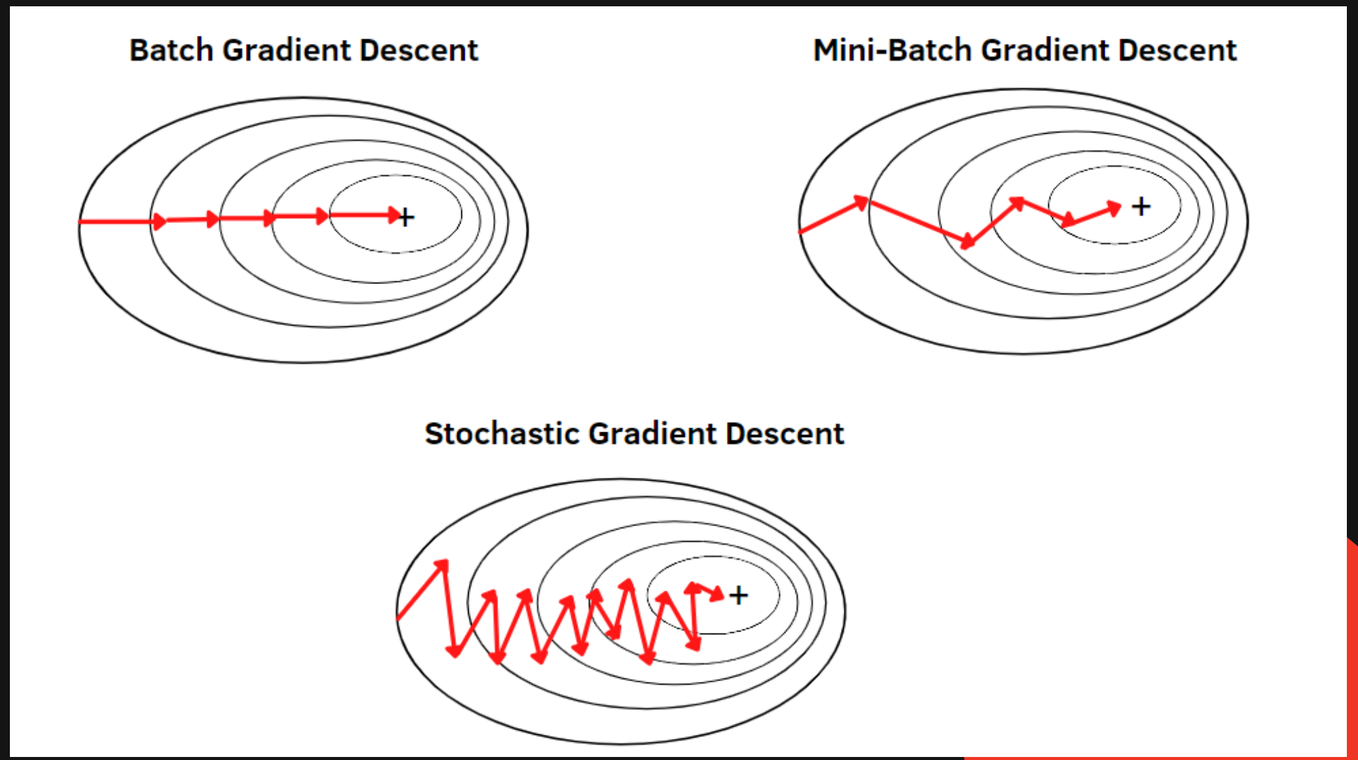


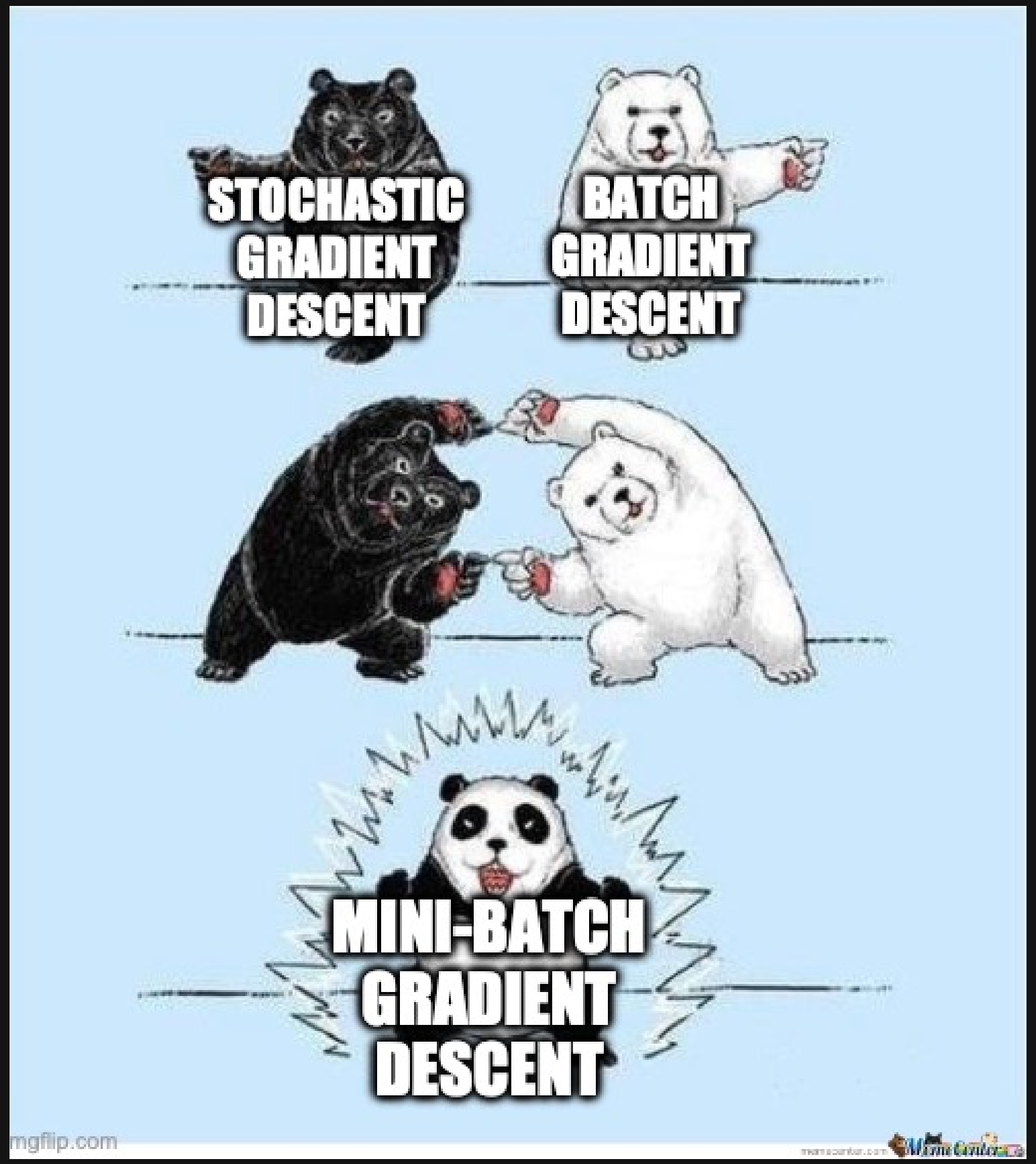
Mini-Batch gradient descent

It is a mixture of Batch Gradient Descent and Stochastic Gradient Descent.
A fixed number of training examples which is less than the actual dataset is called a mini-batch.

But why though?

- SGD's frequent updates can result in noisy gradients, but this can also be helpful in escaping the local minimum and finding the global one.
- Batch Gradient descent can still have a long processing time for large training datasets as it still needs to store all of the data into memory.




STOCHASTIC GRADIENT DESCENT
BATCH GRADIENT DESCENT
MINI-BATCH GRADIENT DESCENT
imgflip.com

Some more optimizers!

The problem with Stochastic gradient Descent is that is helps reach the **local minima of the loss function and not the global one**. There are optimizers which provide more robustness in finding the **global minima.**

SGD with Momentum

$$W_{t+1} = W_t - V_t$$

'V_t' is a term providing velocity to the change in weights.

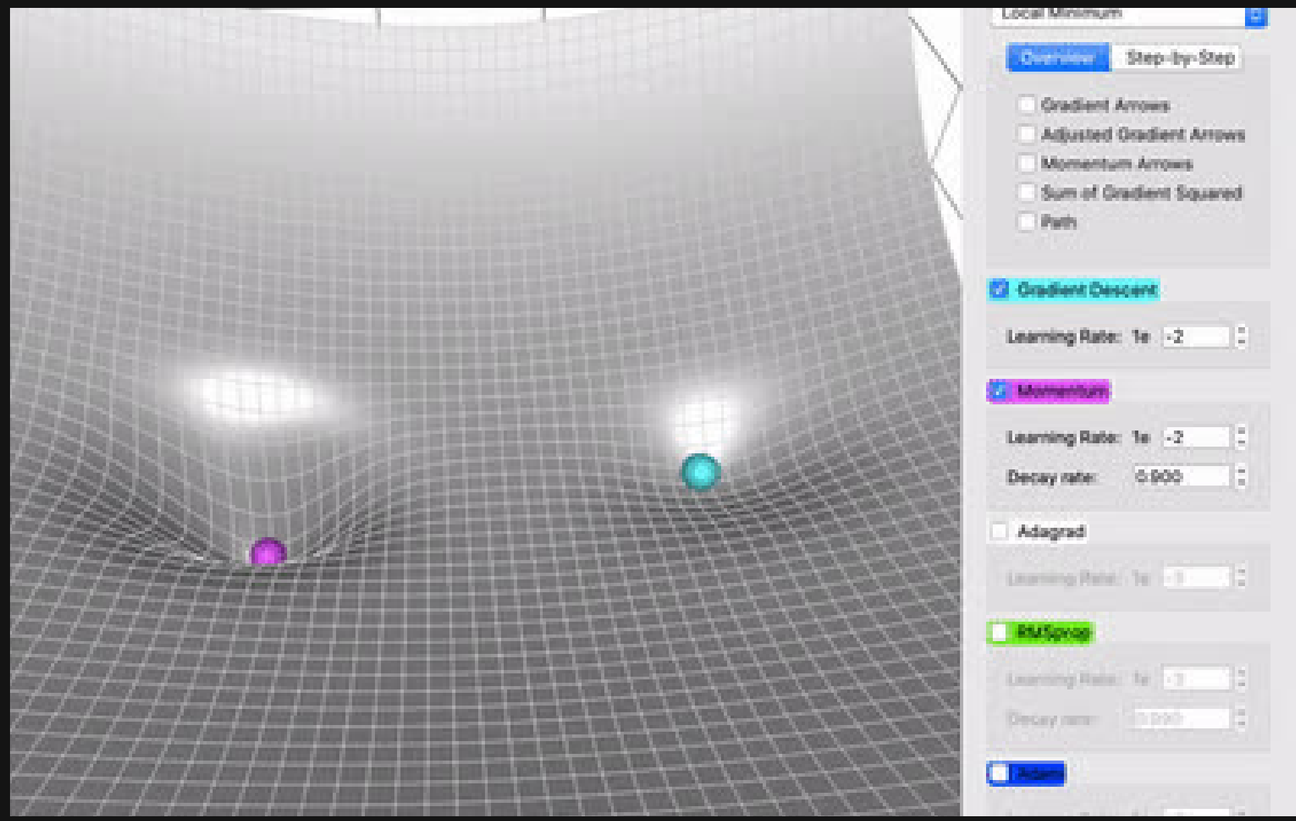
$$V_t = \begin{cases} 0, & t = 0 \\ \beta V_{t-1} + \alpha \frac{\partial L}{\partial W_t}, & t > 0 \end{cases}$$

where $\beta \in (0, 1]$ (parameter);

α is learning rate

$\frac{\partial L}{\partial W_t}$ is gradient of loss wrt W. at time t.

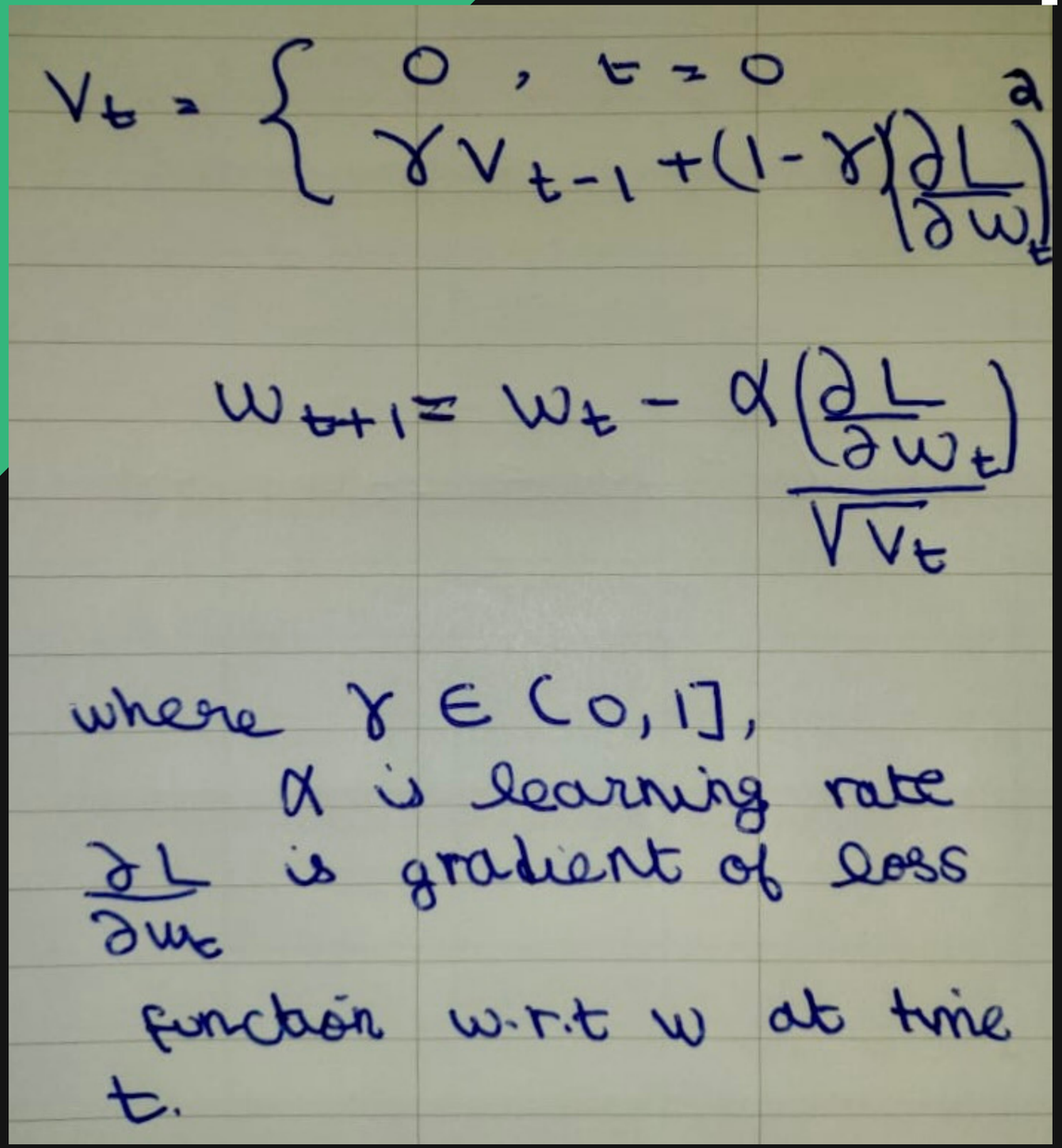
- The higher the value of β the more we get to remember past data, **and the faster we get to the minima.**



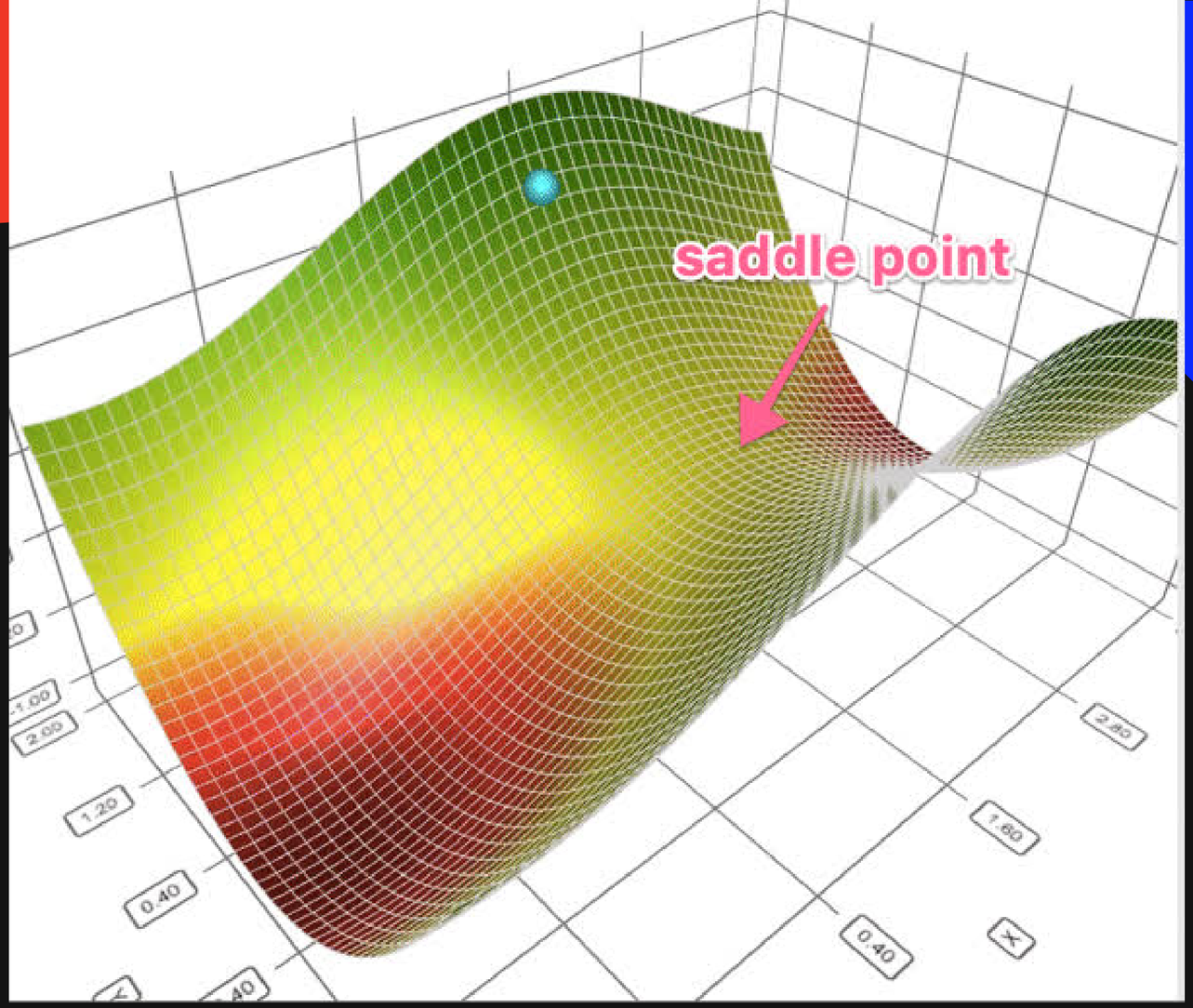
But here's a catch...

That's right, SGD with momentum can be so fast that it can go right past the global minima.

Root Mean Square Propagation



- The higher the value of **Γ**, the more we get to remember past data, **and the slower we get to the minima.**
- Thanks to the accumulated-gradient-squared term in the denominator, **the more you move down along a slope, the less you will have to move along it in the future.**
- aka RMSProp



The white ball and green balls are the loss functions of RMSProp and SGD respectively

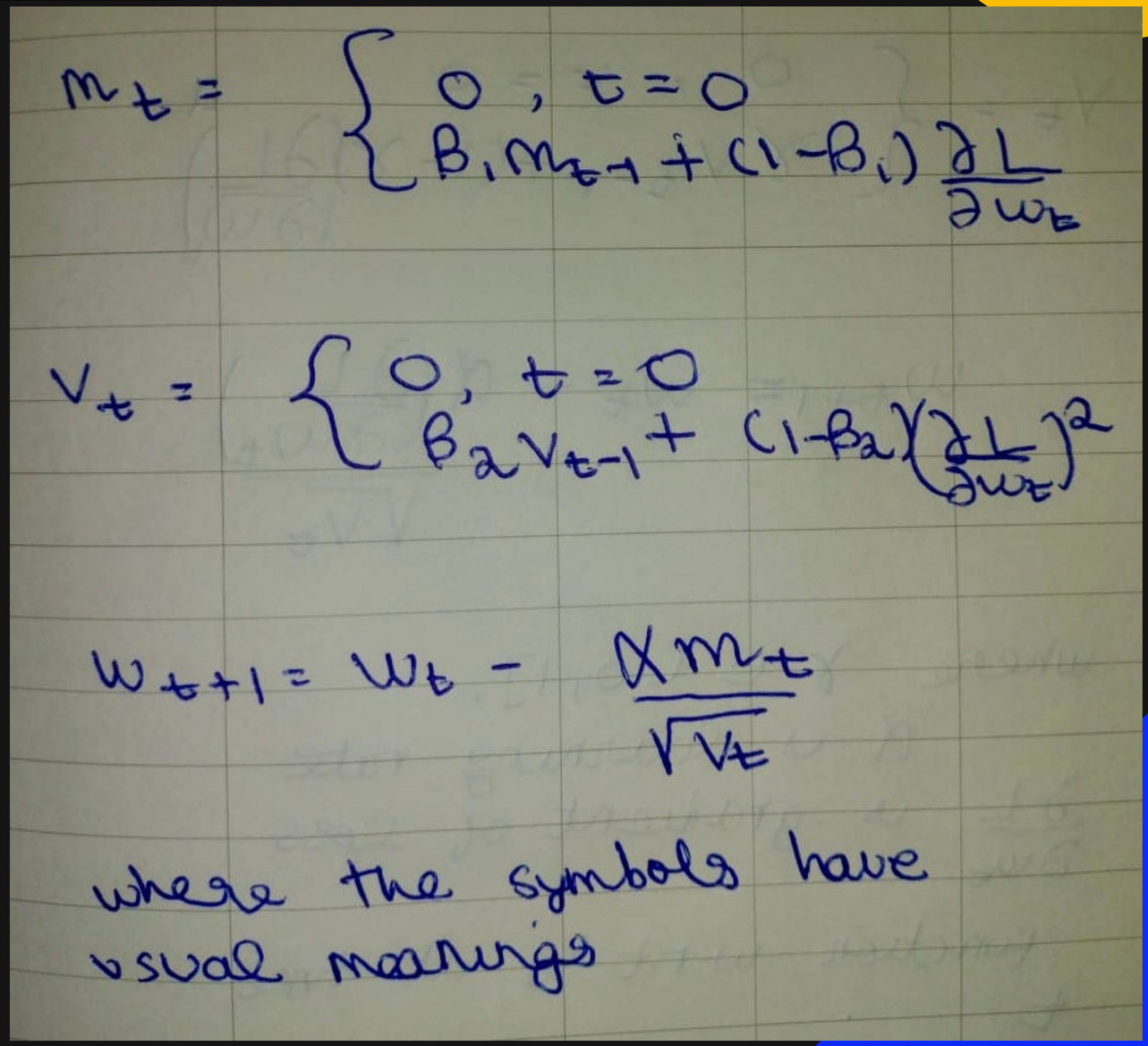
---

Thanks to the Γ term, RMSProp is not THAT slow. **Escaping saddle points is a great plus-point of RMSProp.** However, we can do much better in terms of speed.

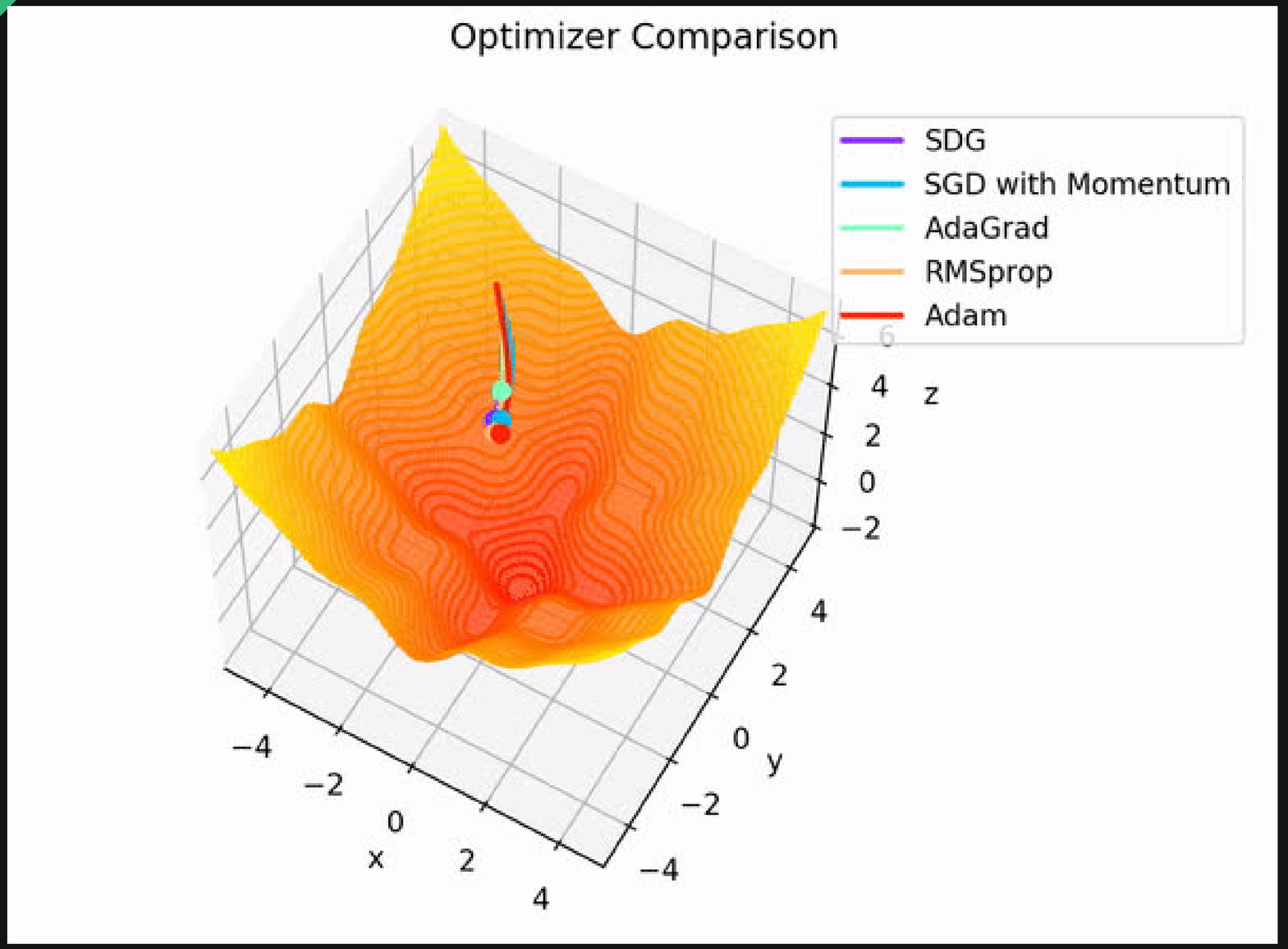
Adam

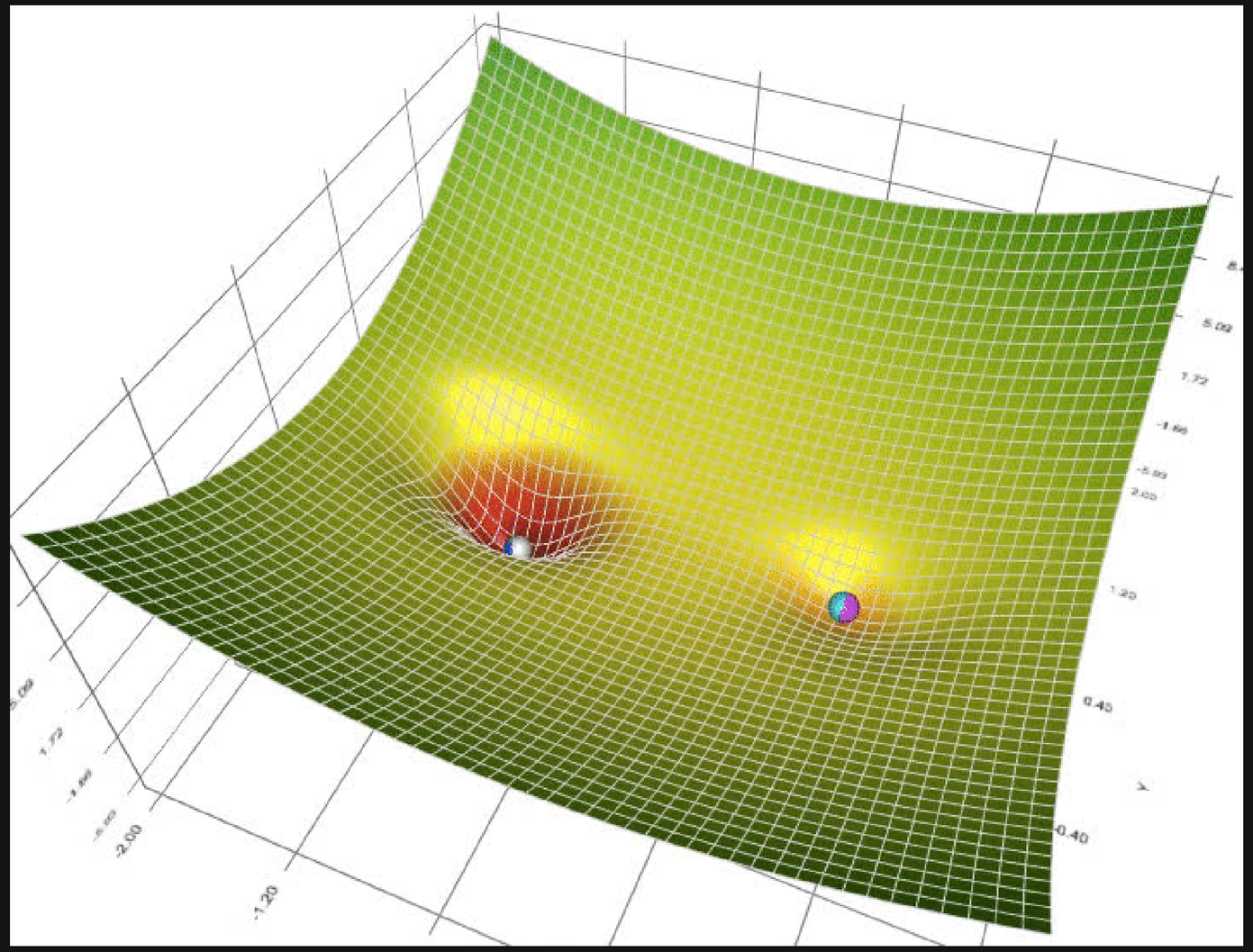
- Adam (short for Adaptive Moment Estimation) takes the best of both worlds of Momentum and RMSProp.

- **The intuition behind the Adam is that we don't want to roll so fast just because we can jump over the minimum, we want to decrease the velocity a little bit for a careful search.**

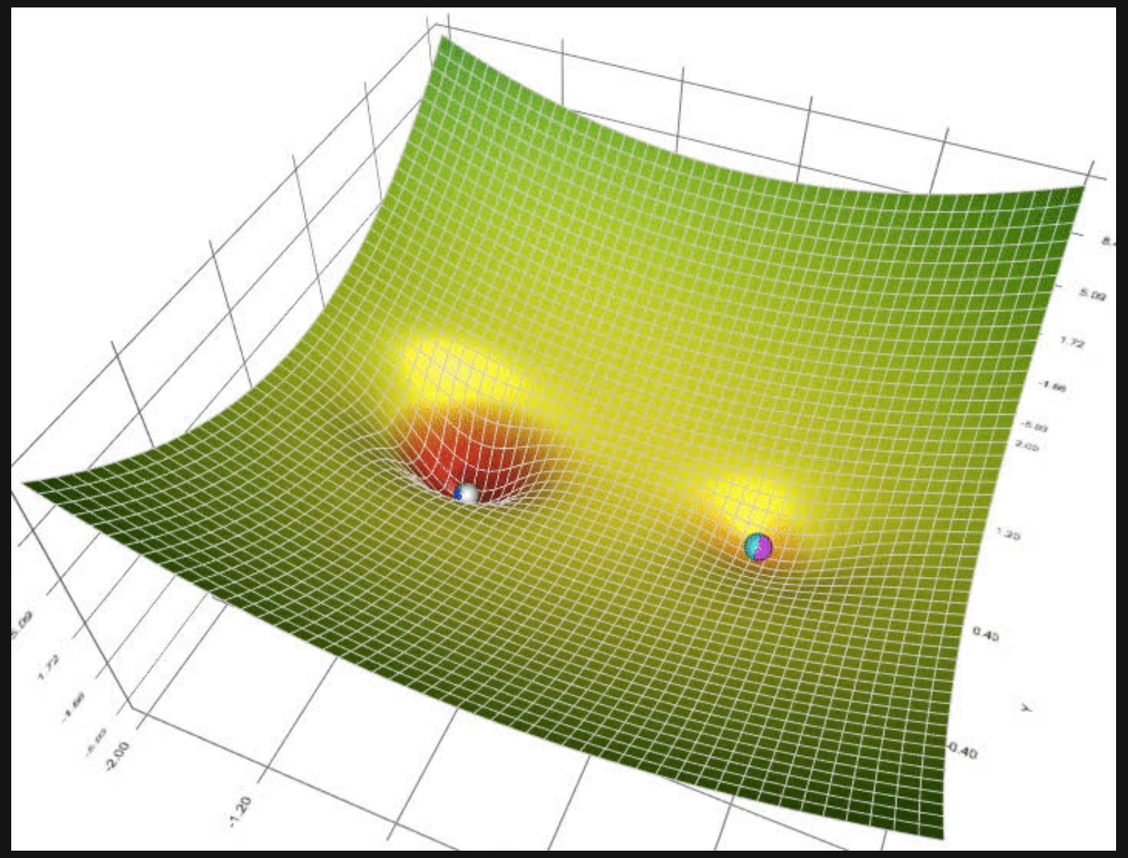


Optimizers: A Summary



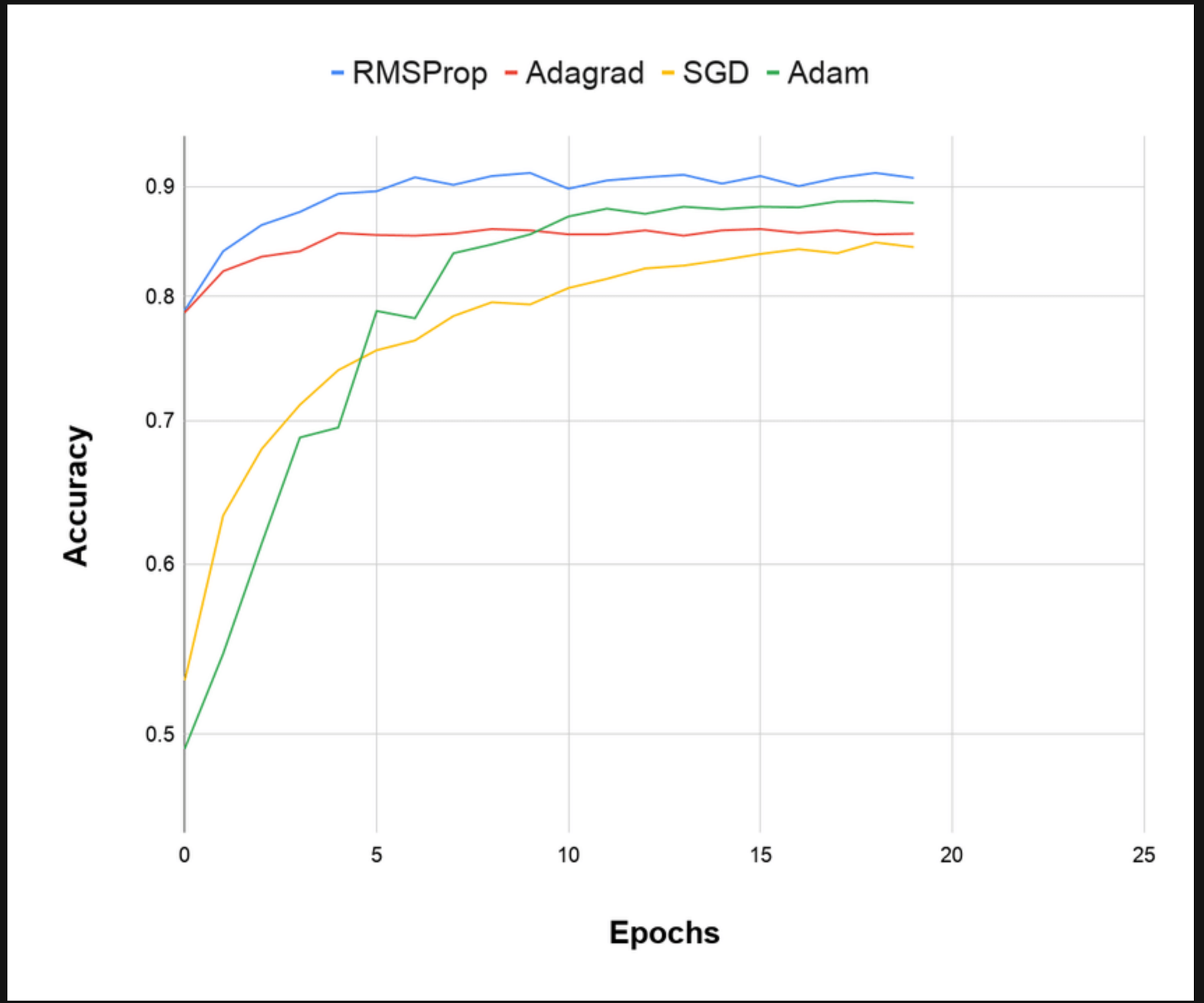


Which is which?



- Gradient descent (cyan),
- Momentum (magenta),
- AdaGrad (white),
- RMSProp (green),
- Adam (blue).

Left well is the global minimum; right well is a local minimum.


RMSProp
Adagrad
SGD
Adam
Accuracy
0.9
0.8
0.7
0.6
0.5
0
5
10
15
20
25
Epochs