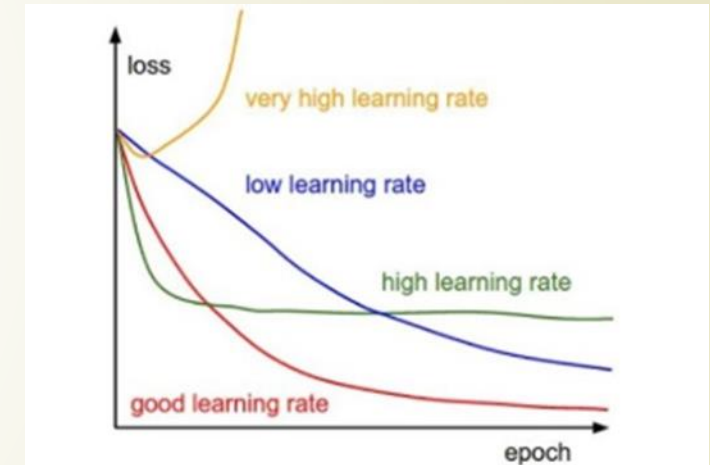


PROBLEMS

- Tuning hyperparameters like learning rate can be very difficult sometimes.
- The high-dimensional non-convex nature of neural networks optimization could lead to different sensitivity on each dimension.
- The learning rate could be too small in some dimension and could be too large in another dimension.
- Adagrad addresses this problem





ADAGRAD

It adapts the learning rate to the parameters, performing smaller updates (i.e. low learning rates) for parameters associated with frequently occurring features

Earlier an update was performed for all parameters θ at once as every parameter θ_i used the same learning rate η . But Adagrad uses a different learning rate for every parameter θ_i at every time step t .

THE MATHEMATICS BEHIND IT

$$g_{t,i} = \nabla_{\theta_t} J(\theta_{t,i})$$

$$\theta_{t+1,i} = \theta_{t,i} - \frac{\eta}{\sqrt{G_{t,ii} + \epsilon}} \cdot g_{t,i}$$

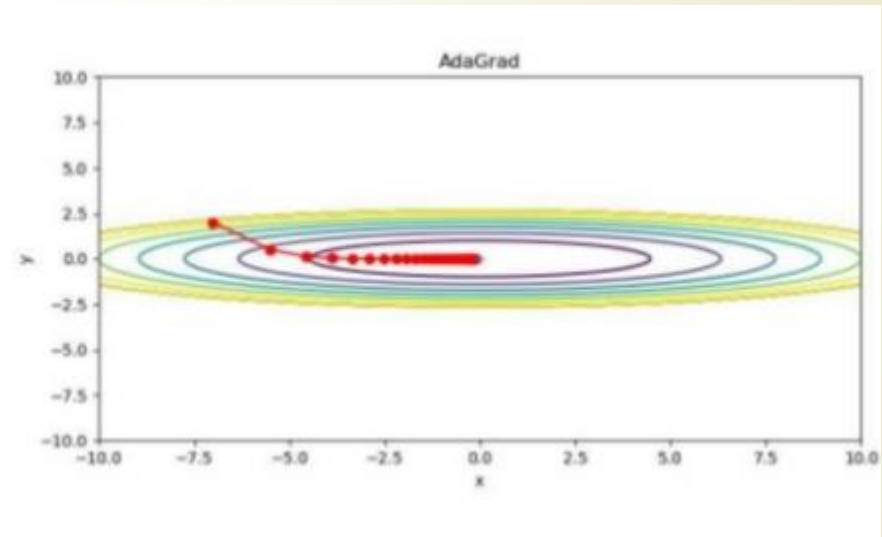
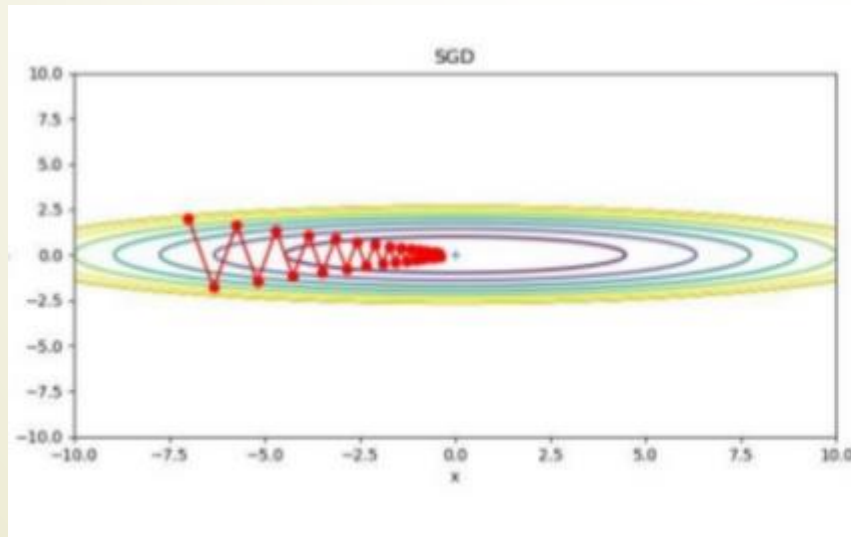
$G \in \mathbb{R}^{d \times d}$ here is a diagonal matrix where each diagonal element i , i is the sum of the squares of the gradients w.r.t. θ_i up to time step t . The matrix G is defined as $G = \sum g_i \cdot g_i^T$ where i varies from 1 to t . ϵ is a smoothing term

ADVANTAGE

Does away with the need to set the learning rate manually

DISADVANTAGE

The accumulation of positive terms in the denominator causes the update to become infinitesimally small throughout training, and model fails to learn more






ADADELTA

Adadelata is built on AdaGrad and attempts to address some of the above-mentioned disadvantages.

AdaDelta accumulates the gradients in the previous time steps using an *exponentially decaying average of the squared gradients*.

This prevents the denominator from becoming infinitesimally small, and ensures that the parameters continue to be updated even after a large number of iterations

This method has been shown to perform relatively well when used to train deep networks, and is much less sensitive to the choice of hyper-parameters.



The *running average* $E[g^2]$ at time step t then depends (as a fraction γ similarly to the Momentum term) only on the previous average and the current gradient

$$E[g^2]_t = \gamma E[g^2]_{t-1} + (1 - \gamma)g_t^2$$

$$\Delta\theta_t = -\frac{\eta}{\sqrt{E[g^2]_t + \epsilon}}g_t$$



REFERENCES

<https://medium.com/iitg-ai/into-the-depths-of-gradient-descent-52cf9ee92d36> (Into the depths of gradient descent, Medium blog, IITG.ai)

