Evaluation of Various Gradient Descent Optimization Techniques for Neural Networks

Mit Patel, Dhaval Patel, Prajeet Bhavsar

***Abstract*— Ever since the inception of Gradient Descent algorithm, it is without a doubt the most popular optimization strategy used in machine learning and deep learning. In this paper we have used various techniques to accelerate the gradient vectors in the right direction. The main problem with gradient descent algorithm is the rate of convergence. So to speed up the process, we take into effect all the previous gradients and tune the current gradient accordingly. There are various techniques available to do so and we have explored 5 such techniques namely i) No Momentum ii) Polyak's Classical Momentum iii) Nesterov's Accelerated Gradient (iv) RmsProp and (v) ADAM . We will compare the accuracies, rate of convergence and the stability for all this techniques in this paper.**

***Index Terms*—Gradient Descent, Polyak's classical momentum, Nesterov's Accelerated Gradient, RmsProp, ADAM, MNIST fashion Data Set**

I. INTRODUCTION

THE most common method for neural network optimization is gradient descent and is one of the most favored algorithms. and almost every deep learning library is equipped with various implementations of numerous algorithms to optimize the gradient descent. in this paper, we have implemented some of those techniques.

The main objective of Gradient descent is to minimize an objective function $J(\theta)$ parameterized by a model's parameters $\theta \in \mathbb{R}^d$ by changing the parameters in the opposite direction of the gradient of the objective function $\nabla_\theta J(\theta)$ w.r.t. to the parameters. To control the size of leaps we take to converge (local minima), we use hyperparameter η – learning rate. Particularly, we go along the direction of the slope of the surface created by the objective function downhill until we reach a minima. Now this is easily achieved if the objective function is convex in nature. But that is not usually the case in real world problems. And if the objective function is not convex in nature, the probability of converging to a local minima is very high.

The nature of many real world machine learning problems have non convex formulation of the objective function. The problem of finding the global minimum in such cases is treated as NP hard problem[1]. Gradient Descent is naturally the first go to iterative algorithm to solve these problems. There are a number of variations of Gradient Descent methods and most of them can be collectively classified into these three: i) Momentum-based methods (e.g. Nesterov's Accelerated Gradient), ii) Variance Reduction Methods (e.g. Stochastic Variance Reduced Gradient) and iii) Adaptive Learning Methods (e.g. AdaGrad) [1].

In this work, we study momentum-based methods and implement the same on MNIST fashion dataset [2]. The paper is organized as follows: Section 2 discusses the various techniques and background, Section 3 discusses the methodology and architecture of our neural network, Section 4 showcases the comparisons between the performances of different techniques on the dataset, Section 5 contains the conclusions that we drew from the experiment, Section 6 shows the division of work for our project and finally Section 7 has the self-peer evaluation table followed by references.

II. RELATED WORK

As mentioned in the above section, the estimation of the network parameters can take a lot of time if we have a large dataset. The equation for the gradient descent parameter update is given by [3]:

$$\theta = \theta - \eta \nabla_\theta J(\theta) \tag{1}$$

So in order to speed up the convergence in gradient descent, some of the widely used optimization techniques are Polyak's classical momentum, Nesterov's Accelerated Gradient, RMSProp, ADAM, Adagrad and AdaDelta.

A. Polyak's classical momentum:

This method accelerates the stochastic gradient descent by taking the exponentially weighted average of the gradients into consideration. Here the parameters are updated by adding a fraction γ of the previous iteration's update vector to the current update vector.

$$v_t = \gamma v_{t-1} + \eta \nabla_\theta J(\theta) \tag{2}$$
$$\theta = \theta - v_t \tag{3}$$

The dimensions whose gradients are in same direction, the momentum term γ will increase and hence this will result in faster convergence [3].

B. Nesterov's Accelerated Gradient

In the momentum method, as we reach toward the minima, the momentum can be high which may lead to miss the lowest point. In Nesterov's Accelerated Gradient, we compute $\theta - \gamma v_{t-1}$ which gives us a rough estimate of the next position of the parameters.

$$v_t = \gamma v_{t-1} + \eta \nabla_\theta J(\theta - \gamma v_{t-1}) \tag{4}$$
$$\theta = \theta - v_t \tag{5}$$

Thus, NAG provides an ability to look ahead by calculating gradient with respect to the rough estimate of future position of the parameters as opposed to the current parameters [4].

C. RMSProp:

The main idea behind RMSProp is to update the learning rate at each step. Let g_t be the gradient at time step t. Here the sum of gradients is defined recursively as a decaying average of previously squared gradients. Hence the running

average $E[g^2]_t$ at time step t will depend only on the previous average and the current gradient [3]. RMSprop will then divide the learning rate by this exponentially decaying average of squared gradients.

$$E[g^2]t = \gamma E[g^2]_{t-1} + (1-\gamma)g_t^2 \quad (6)$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} g_t \quad (7)$$

In practice for good results, γ is set to 0.9 and the learning rate η to 0.001.

D. ADAM

In ADAM, we store exponentially decaying average of past squared gradients v_t as well as exponentially decaying average of past gradients m_t, and use it to update the weights θ_{t+1}.

$$m_t = \beta_1 m_{t-1} + (1-\beta_1)g_t \quad (8)$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2)g_t^2 \quad (9)$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{v_t} + \epsilon} m_t \quad (10)$$

The default values of 0.9 for β_1 , 0.999 for β_2, and 10^{-8} for ϵ are favorable. They show empirically that Adam works well in practice and compares favorably to other adaptive learning-method algorithms.

III. METHODOLOGY

IV. EXPERIMENT

V. CONCLUSION

VI. DIVISION OF WORK

VII. SELF-PEER EVALUATION TABLE

Prajeet Bhavsar:	Dhaval Patel:	Myself:

Table 1: Self-Peer Evaluation Table

VIII. REFERENCES

[1] Vishwak Srinivasan, Adepu Ravi Sankar and Vineeth N Balasubramanian,"ADINE: An Adaptive Momentum Method for Stochastic Gradient Descent ",Indian Institute of Technology Hyderabad

[2] H. Xiao, K. Rasul, and R. Vollgraf. (2017) Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms.

[3] S. Ruder, An overview of gradient descent optimization algorithms, 2018 [Online]. Available: http://ruder.io/optimizing-gradient-descent/

[4] Anish Singh Walia, Types of Optimization Algorithms used in Neural Networks and Ways to Optimize Gradient Descent. [Online]. Available: https://towardsdatascience.com/types-of-optimization-algorithms-used-in-neural-networks-and-ways-to-optimize-gradient-95ae5d39529f