Comparison of stochastic gradient descent based optimization techniques using Resnet50 on MNIST Dataset

Mert Burak Burabak (2020699)

June 13, 2021

Abstract

In this study, 5 different methods related to stochastic gradient descent optimization are explained and their applications are explained. For this, we trained the ResNet50 model on the well-known and used MNIST dataset with 5 different techniques. Considering the duration metric, while the RMSprop algorithm is the fastest working technique, it has been observed that Adam and Momentum techniques are more successful than other techniques according to the Loss and Accuracy metrics.

Keywords: Optimization Techniques, Deep Learning, ResNet50, MNIST, Momentum, SGD, Adam, RMSprop, Adadelta, Adagrad

Table of Contexts

1.Introduction

Gradient descent method is very important in machine learning. In neural networks, weight values at each step needs to be updated until the best learning process and this update process is specificly done within the different methods. One of the most commonly used methods for parameter updating is "Gradient Descent" method.

The gradient descent method is used in artificial neural networks to minimize the cost function (J(θ)) based on theta parameters. The learning of a network is performed by updating the randomly determined theta parameters at each step in the opposite direction of the gradient and repeating the process of training the network. There are three different types of gradient descent method, "Batch Gradient Descent", "Mini-Batch Gradient Descent" and "Stochastic Gradient Descent"[1].

$$\theta_t = \theta_{t-1} - \alpha . \nabla_\theta J(\theta) \qquad (1)$$

Here theta is the parameter vector to be updated, alpha is the learning coefficient and $\nabla_\theta J(\theta)$ is the gradient of the cost function. In order to make a single update according to the Batch

Gradient Descent method, it is necessary to calculate the gradient of the entire data set. Therefore, this method works very slowly and memory problems may be encountered in the application of large datasets [1].

In the Stochastic Gradient Descent (SGD) method, the update process is performed for each training set. Therefore, it runs faster and can reach the local minimum sooner.

$$\theta_t = \theta_{t-1} - \alpha.\nabla_\theta J(\theta; x^i, y^i) \qquad (2)$$

The Mini-Batch Gradient Descent method has been developed considering the advantages of both methods. The update process is performed for each n training set.

$$\theta_t = \theta_{t-1} - \alpha.\nabla_\theta J(\theta; x^{i+n}, y^{i+n}) \qquad (3)$$

In artificial neural networks, the learning success of the network is proportional to the convergence of the cost function to zero. In other words, obtaining theta parameters that will reduce the cost function to the minimum value shows the success rate of the network. In order to increase this success rate or to update the parameters faster, some studies have been developed on the optimization of the Gradient Descent method. In this study, it is aimed to better understand some of these methods by applying them. In the second part, 5 different methods related to stochastic gradient descent optimization are explained and their applications are explained. In the third part, the experimental results of the application were compared and the methods were compared.

2. Optimization Methods and Applications

One of the most important disadvantages of the gradient descent method is the determination of a fixed learning coefficient. When a small learning coefficient is chosen, the learning process will progress slowly and the convergence process to the optimum value will occur slowly. A large learning coefficient may prevent convergence to the optimum value or cause oscillations around the minimum value. Optimization methods are generally focused on updating parameters faster. For this purpose, different techniques have been presented, and while providing this acceleration in some studies, approaches to solving the above-mentioned problem related to learning coefficient have also been proposed [2 - 7]. These approaches are based on the principle that the learning coefficient is not used at all or that this parameter is updated at each step.

A. Momentum

Momentum [2] has been proposed as a method that accelerates the SGD method and acts to reduce oscillations in SGD. This effect is shown in Fig.1 [1].

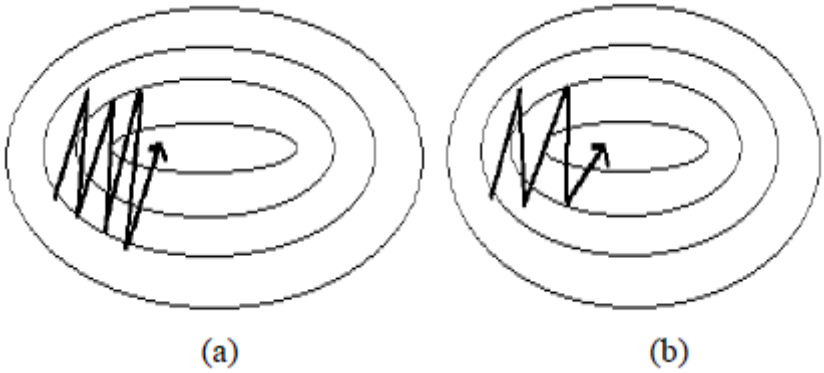

Figure 1 a) SGD without Momentum b)SGD with Momentum

As seen in figure.1(b), when momentum is applied to the SGD method, the oscillations seen while converging to the minimum value decrease. While the momentum method updates the past parameters, it takes into account not only the gradient in that iteration but also the past gradients.

$$v_t = \rho . v_{t-1} + \alpha . \nabla_\theta J(\theta; x^i, y^i) \quad (4)$$

$$\theta_t = \theta_{t-1} - v_t \quad (5)$$

The p parameter seen in Equation.4 is a value between 0 and 1 and determines how much of the previous gradients are taken into account and is usually taken as a value around 0.9.

B. Adagrad

The Adagrad method [3] is a method that eliminates the problem arising from the constant learning coefficient in the gradient descent method. According to this method, the learning coefficient is also updated at each step.

$$G_t = G_{t-1} + \nabla J(\theta_{t-1})^2 \quad (6)$$

$$\theta_t = \theta_{t-1} - \frac{\alpha}{\sqrt{G_{t-1}}} . \nabla J(\theta_{t-1}) \quad (7)$$

G_t is a matrix of the same size as the number of parameters, holding historical gradient information. Each element of this matrix represents the sum of the squares of the previous gradients of the corresponding parameter. This matrix is used to scale the learning coefficient. Thus, the degradation coefficient is updated at each step depending on the parameters. However, a learning coefficient needs to be determined initially. Another point to note here is the term ε in the denominator. This term is a very small value used to avoid division by zero and is usually taken as 10^{-8}[8].

C. Adadelta

A weakness of the Adagrad method is that the learning coefficient converges to zero over time. As can be seen in Equation.5, the growth of the G matrix over time causes the learning coefficient to decay. This is one of the important roles in the development of the Adadelta [4] method. Another is the need for the learning coefficient parameter at the beginning. The Adadelta method has eliminated these two disadvantages of the Adagrad method.

$$E[g^2]_t = \rho E[g^2]_{t-1} + (1-\rho) g_t^2 \qquad (8)$$

$$\Delta\theta_t = -\frac{RMS[\Delta\theta]_{t-1}}{RMS[g]_t} \cdot g_t \qquad (9)$$

$$RMS[g]_t = \sqrt{E[g^2]_t + \epsilon} \qquad (10)$$

$$E[\theta^2]_t = \rho E[\theta^2]_{t-1} + \Delta\theta_t^2 \qquad (11)$$

$$\theta_{t+1} = \theta_t - \Delta\theta_t \qquad (12)$$

As seen in Equation.8, Adadelta method takes the sum of the gradient squares at a certain rate, not all the squares of the previous gradients, as in the Adagrad method. It adjusts this with the constant term p, as in the momentum method.

D. RMSprop

RMSProp [5] was proposed by Geoff Hinton and is included in his lecture notes. The Adadelta method was recently introduced independently of each other [1]. This method is similar to the Adadelta method but uses the learning coefficient parameter.

$$E[g^2]_t = \rho E[g^2]_{t-1} + (1-\rho) g_t^2 \qquad (13)$$

$$\theta_{t+1} = \theta_t - \frac{\alpha}{\sqrt{E[g^2]_t + \epsilon}} \cdot g_t \qquad (14)$$

While presenting this method in his notes, Hinton suggested that taking the value of the p parameter as 0.9 and the value of the kissing coefficient as 0.001 would yield good results.

E. Adam

The Adaptive Moment Estimation (Adam) method [6] is one of the methods that updates the learning coefficient in each iteration. As with the Adadelta and RMSProp methods, it not only takes into account the squares (vt) of previous gradients, but also takes into account past gradients without being quadratic (mt).

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t \quad (15)$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \quad (16)$$

$$\acute{m}_t = \frac{m_t}{1-\beta_1^t} \quad (17)$$

$$\tilde{v}_t = \frac{v_t}{1-\beta_2^t} \quad (18)$$

$$\theta_{t+1} = \theta_t - \frac{\alpha}{\sqrt{\tilde{v}_t + \epsilon}} . \acute{m}_t \quad (19)$$

This method is designed to combine the advantages of Adagrad and RMSProp methods [7]. In the study, it was stated that this method corresponds to the Adagrad method when β_1 =0, β_2 is chosen as a very small and the learning coefficient is chosen as ά=ά.$t^{-1/2}$. The difference with RMSProp is that while RMSProp performs the parameter update using a momentum on the scaled gradient, in this method, the updates are expressed by using the average of the first and second moments of the gradient.

3.Experimental Results

In this study, the Stochastic Gradient Descent method and the optimization methods explained above (Momentum, Adagrad, Adadelta, Rmsprop and Adam) were applied and the results were compared. The application for this is coded in Python language, and the parameters updated collectively for the same number of iterations, and the results of the cost function are shown. All models have same architecture and all models trained on MNIST dataset[10].

In practice, the fixed parameters used by the methods are set to ά = 10^{-5}, p = 0.9 and ε = 10^{-7}, and β_1 = 0.9 and β_2 = 0.999 used in the Adam method. Other parameters are:

- Validation size = 0.2
- Batch size = 64
- Epochs=5

The experiment results are shown in Figure 2.

Optimization Technique	Convergence Time(s)	Loss	Accuracy
Momentum	432.758	0.0738	**0.9791**
RMSprop	**345.282**	0.0792	0.9767
Adam	391.794	**0.0743**	0.9787
Adadelta	391.229	0.0827	0.9767
Adagrad	391.347	0.0817	0.9766

Figure 2Experiment Results

In Figure 2, we trained our models on the MNIST dataset using the ResNet50 model. We compared all our training results. From the results, we see that the technique that gives the fastest results among the optimization techniques is the RMSprop technique. In the loss

function, there are 2 techniques that give the least loss. These are the Momentum and Adam techniques and are very close to each other. In the Accuracy metric, all techniques are very close to each other and show only minor differences. Momentum and Adam techniques give the best results.

4.Conclusion

In this study, 5 different methods related to stochastic gradient descent optimization are explained and their applications are explained. For this, we trained the ResNet50 model on the well-known and used MNIST dataset with 5 different techniques. Considering the duration metric, while the RMSprop algorithm is the fastest working technique, it has been observed that Adam and Momentum techniques are more successful than other techniques according to the Loss and Accuracy metrics.

5.References

1- S. Ruder, "An overview of gradient descent optimization algorithms." *arXiv preprint arXiv:1609.04747*

[2] N. Qian, "On the momentum term in gradient descent learning algorithms. Neural Networks", The Official Journal of the International Neural Network Society, 12(1), 145–151.

[3] J. Duchi, E. Hazan and Y. Singer, "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization." Journal of Machine Learning Research, 12, 2121–2159, 2011.

[4] http://int8.io/comparison-of-optimization-techniques-stochasticgradient-descent-momentum-adagrad andadadelta/#AdaGrad_8211_description. Nisan, 2017

[5] M.D. Zeiler, "ADADELTA: an adaptive learning rate method." arXiv preprint arXiv:1212.5701, 2012.

[6] T. Tijmen, G. Hinton. "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude." COURSERA: Neural networks for machine learning, 4.2 2012.

[7] D. P. Kingma, J. L. Ba, "Adam: a Method for Stochastic Optimization." International Conference on Learning Representations,1–13, 2015.

[8] https://www.sfu.ca/~ssurjano/griewank.html, May□s 2017

[9] C. Gulcehre, M. Moczulski, and Y. Bengio. "ADASECANT: Robust Adaptive Secant Method for Stochastic Gradient." arXivpreprint arXiv:1412.7419, 2014.

[10]- http://yann.lecun.com/exdb/mnist/index.html