ISSN No:-2456-2165

Comparative Analysis of Optimizers in Deep Neural Networks

Chitra Desai
Professor
Department of Computer Science
National Defence Academy, Pune, India

Abstract:- The role of optimizer in deep neural networks model impacts the accuracy of the model. Deep learning comes under the umbrella of parametric approaches; however, it tries to relax as many as assumptions as possible. The process of obtaining parameters from the data is gradient descent. Gradient descent is the chosen optimizer in neural network and many of the machine learning algorithms. The classical stochastic gradient descent (SGD) and SGD with momentum which were used in deep neural networks had several challenges which were attempted to resolve using adaptive learning optimizers. Adaptive learning algorithms like- RMSprop, Adagrad, Adam wherein learning rate for each parameter is computed were further developments for better optimizer. Adam optimizer in Deep Neural Networks is often a default choice observed recently. Adam optimizer is a combination of RMSprop and momentum. Though, Adam since its introduction has gained popularity, there are claims that report convergence problem with Adam optimizer. Also, it is advocated that SGD with momentum gives better performance compared to Adam. This paper presents comparative analysis of SGD, SGD with momentum, RMSprop, Adagrad and Adam optimizer on Seattle weather dataset.The Seattle weather dataset, was processed assuming Adam optimizer will prove to be the better optimizer choice as preferred a default choice by many, however, SGD with momentum proved to be a unsurpassed optimizer for this particular dataset.

Keywords:- *Gradient Descent, SGD with momentum RMSprop, Adagrad and Adam.*

I. INTRODUCTION

Deep learning algorithms involve optimizations. Optimization refers to minimizing or maximizing an objective function, which, is also called cost function or loss function. Given a training dataset for deep neural network, there are attempts to find optimal parameters (θ) that significantly reduce the cost function J (θ). Gradient descent can be used in deep neural network to find the optimal parameters [1].

Training deep learning models are iterative and requires initial point to be specified to start with and it is this initialization that strongly affects most algorithm [2]. The classical Stochastic Gradient Descent (SGD) [3] and SGD with momentum have proven track of their suitability for learning deep neural network. Enhancement to existing techniques is inevitable and so came set of adaptive learning methods.

Adaptive learning methods were developed over a period of time to claim their supremacy over classical SGD and SGD with momentum. However, several studies [4][5][6]show that SGD with momentum proved comparatively better than the adaptive learning methods in particular Adam optimizer which tends to be a default choice.

The paper aims at analyzing the performance of deep neural network by applying different optimizer to the chosen dataset.

The dataset is divided into training set and test set. The deep neural network is trained on the training data and tested on the test data.

The paper does not cover the underlying data preprocessing and deep neural network, the focus here is on minimizing the training and validation loss and observing the testing loss by changing optimizers. The optimizer used for comparative study in this paper are SGD, RMSprop, Adagrad, SGD with momentum and Adam.

II. DATA AND DATA PRE-PROCESSING

The dataset for study used is Seattle, US weather dataset [7]. It is labelled dataset which consists of 4 feature variables – DATE, PRCP, TMAX and TMIN and one target variable RAIN which is categorical having value {0,1}. The dataset contains 25552 records of daily rainfall patterns from 1st Jan 1948 to 12th Dec 2017. The data is preprocessed to provide input to the deep neural network by checking for duplicates, removing null values and splitting of DATE column DAY, MON and YEAR. Table 1 shows the sample data after splitting the column. Scaling applied to the data is standardization. Data is split into train and test with a ratio of 80:20.

TABLE 1. SAMPLE DATA AFTER SPLITTING DATE COLUMN

PRCP	TMAX	TMIN	RAIN	YEAR	MON	DAY
0.47	51	42	1	1948	1	1
0.59	45	36	1	1948	1	2
0.42	45	35	1	1948	1	3
0.31	45	34	1	1948	1	4
0.17	45	32	1	1948	1	5

The elaborated details of data pre-processing for Seattle weather data set and the Deep neural architecture as presented in next session can be referred at [8].

III. DNN ARCHITECTURE DESIGN

The overall structure of the deep neural network organized into layers to study the impact of different optimizers is presented here. Deep sequential model is used, the summary of it is shown in table 1. There are six input features for the Seattle weather dataset. The shape of the weights depends upon the shape of the input. The target variable is binary with output either 0 or 1. At hidden layers ReLu [9] activation function is used at hidden layers and sigmoid function is used that output layer. Weights are initialized using uniform optimizers.

Model is compiled by setting the learning rate to 0.001, which is chosen by observing the learning curve by plotting the objective function as a function of time. As the problem belongs to the class of binary classification, the loss is calculated using cross entropy. The batch size is set to 64 and epoch to 10.

The data is scaled and split into training and test data. The model is initialized and then with different optimizers the model fit to analyze the performance with respect to each optimizer under study.

TABLE 2 MODEL SUMMARY

Layer (type)	Output Shape	Param #
dense_1 (Dense)	(None, 6)	42
dense_2 (Dense)	(None, 4)	28
dense_3 (Dense)	(None, 1)	5
Total Params: 75 Trainable params: 75 Non-trainable params: 0		

IV. GRADIENT DESCENT

Given function $y = f(x)$, where x and y are some real numbers. The derivative $\frac{dy}{dx}$ of the function $f(x)$ gives the slope of $f(x)$ at a point x. Derivative is useful in minimizing the function as it tells how a small change in input x, makes corresponding change in the output y. To reduce $f(x)$, we can move x in small steps in opposite direction of the derivative. This technique is known as Gradient Descent [10].

Gradient descent is the way to minimize objective function by updating model's parameter in the opposite direction of the gradient. When the derivative of the function $f(x)$ is zero, then it provides no information of the direction to move, this point is known as critical point [2]. So, a critical point is a point with slope zero. When the critical point is lower than the neighboring points, then it is local minima. When the critical point is higher than the neighboring points then it is local maxima. When the critical point has both higher and lower points in its neighboring than it is called saddle point.

Gradient descent is effective for training neural network based on small local moves and reaching the global solution. In gradient descent the weights are updated incrementally after each epoch. There are limits on the performance of any optimization algorithm that are designed for neural network [11]. There are variants of gradient descent [12] and in this paper we discuss SGD, SGD with momentum, RMSprop, Adagrad, and Adam optimizers for analyzing their performance in terms of test accuracy.

V. OPTIMIZERS

For large training set Stochastic Gradient Descent (SGD) [13]is considered as good learning algorithm to train neural networks [10]. It updates the parameters using single or very few parameters, where the new update parameter is given by eq.1, here xi and yi are from the training set. It helps to reduce the variance and lead to stable convergence. α is the learning rate.

$$\theta = \theta - \alpha\nabla\theta J(\theta; xi, yi) \quad (1)$$

If the objective is shallow SGD may tend to oscillate. This problem can be overcome by adding momentum to SGD. ν is the current velocity.$\gamma \in (0,1]$ determines number of iterations of the previous gradients are incorporated into the current update.

$$\nu = \gamma\nu + \alpha\nabla\theta J(\theta; xi, yi) \quad (2)$$

$$\theta = \theta - \nu \quad (3)$$

While implementing SGD with momentum the value of momentum is set to 0.9 during the experiment.

The Adagrad [14] adapts all model parameters by scaling them inversely proportional to the square root of the sum of all the historical squared values of gradient. While training DNN models, from the beginning of training if the

squared gradient starts accumulating, it may lead to premature and excessive decrease in the effective learning rate.

RMSprop [15] is an improvement over Adagrad by changing the gradient accumulation into an exponentially weighted moving average.

Adam optimizer [16] short for 'adaptive moments' is considered as a variant of RMSprop and momentum with few variations. It is computationally efficient and requires very less memory. Adam includes bias correction, RMSprop lacks correction factor.

VI. RESULTS

In this section the results obtained for each optimizer are presented.

A. SGD and SGD with Momentum

Table 3 shows the results for SGD and SGD withmomentum.

TABLE 3. SGD AND SGD WITH MOMENTUM

Optimizer	Learning Rate	Momentum	Test Loss	Test Accuracy	Model Training Time (Sec)
SGD	0.01	-	0.2160	0.9337	2.54
SGD	0.001	-	0.2007	0.9364	2.34
SGD with Momentum	0.01	0.9	0.0115	0.9992	2.11

It is observed that no significant change in model performance is observed with change in learning rate from 0.01 to 0.001 in SGD. The time taken to train the model is comparatively less with learning rate 0.001.

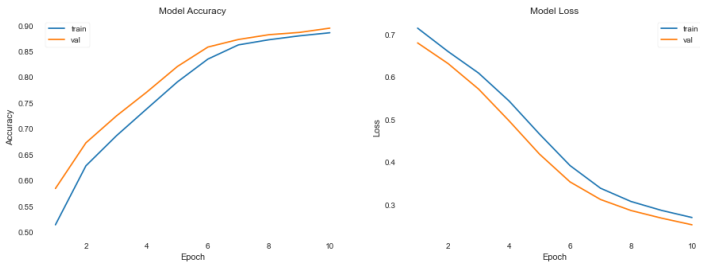

FIGURE 1 MODEL ACCURACY AND MODEL LOSS FOR SGD WITH LEARNING RATE 0.01

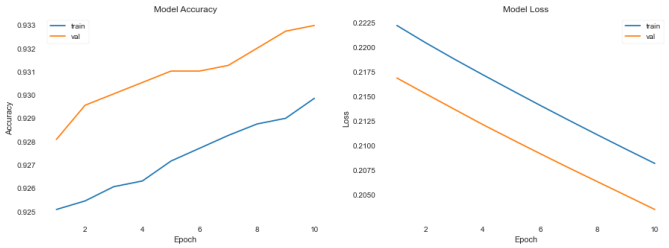

FIGURE 2 MODEL ACCURACY AND MODEL LOSS WITH LEARNING RATE 0.001

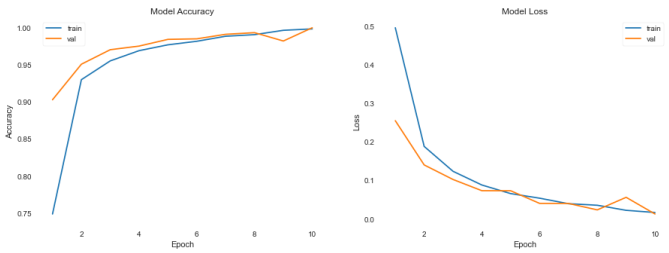

FIGURE 3 MODEL ACCURACY AND MODEL LOSS FOR SGD WITH MOMENTUM

In case of SGD with momentum a significant increase in test accuracy observed compared to SGD and time taken for training is also lowest. Figure 1,2 and 3 shows the accuracy and loss with respect training and validation data over 10 epochs while the model is being trained.

B. Adaptive Learning Algorithms

Table 4 shows the results for adaptive learning algorithms-Adagrad, RMSprop and Adam optimizers. From all the three algorithms it is observed that the model performs best with Adam optimizer. However, time taken to train the model Adagrad is low.

TABLE 4 ADGRAD, RMSPROP AND ADAM

Optimizer	Learning Rate	Test Loss	Test Accuracy	Model Training Time (Sec)
Adagrad	0.01	0.2667	0.8963	2.12
RMSprop	0.01	0.1407	0.9505	2.28
Adam	0.01	0.0837	0.9771	2.26

Figure 4,5 and 6 shows the accuracy and loss for the training and validation data for Adagrad, RMSprop and Adam optimizer respectively. The blue line indicates the training data and the orange indicate the validation data in each of the figure above.

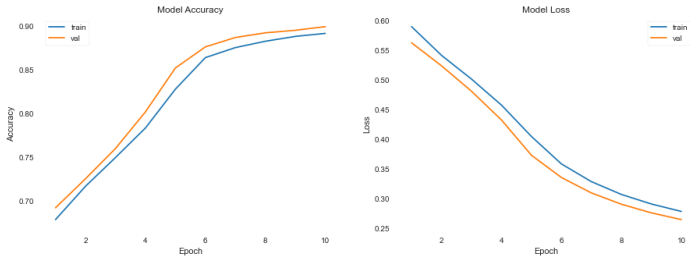

FIGURE 4 MODEL ACCURACY AND MODEL LOSS FOR ADAGRAD

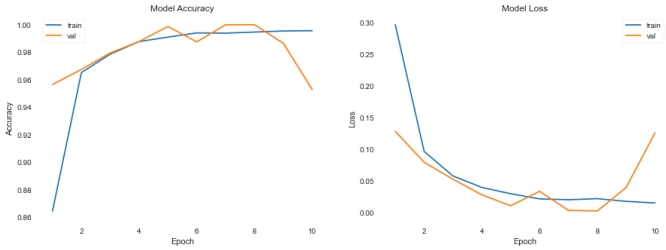

FIGURE 5 MODEL ACCURACY AND MODEL LOSS OR RMSPROP

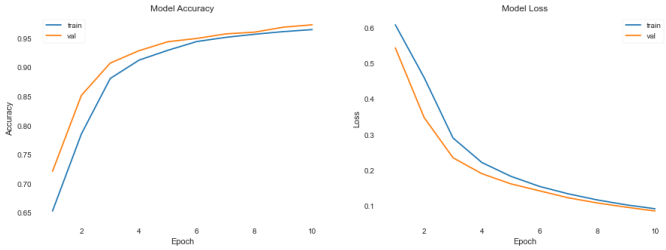

FIGURE 6 MODEL ACCURACY AND MODEL LOSS FOR ADAM

VII. CONCLUSION

The paper presented the impact of different optimizers on the chosen labeled data set. The comparison was mainly aimed to ensure that for labeled data set a default choice of Adam, which a adaptive learning algorithm will give best model performance. However, when SGD with momentum was used, it gave comparatively better result then the adaptive learning algorithms and in particular Adam optimizer. The model training time was also the lowest for SGD with momentum compared to other optimizers.

REFERENCES

[1]. S. S. Du, J. D. Lee, H. Li, L. Wang, and X. Zhai, "Gradient descent finds global minima of deep neural networks," ICML, arXiv:1811.03804, 2018.
[2]. Ian Goodfellow, Yoshua Bengio and Aaron Courville, Deep Learning, MIT Press, 2016
[3]. Robbins, H., & Monro, S. (1951). A stochastic approximation method. The annals of mathematical statistics, 400-407
[4]. Huang, G., Liu, Z., Weinberger, K. Q., & van der Maaten, L. (2017). Densely Connected Convolutional Networks. In Proceedings of CVPR 2017
[5]. Wu, Y., Schuster, M., Chen, Z., Le, Q. V, Norouzi, M., Macherey, W., Dean, J. (2016). Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. arXiv Preprint arXiv:1609.08144.
[6]. Wilson, A. C., Roelofs, R., Stern, M., Srebro, N., & Recht, B. (2017). The Marginal Value of Adaptive Gradient Methods in Machine Learning. arXiv Preprint arXiv:1705.08292. Retrieved from http://arxiv.org/abs/1705.08292
[7]. https://www.kaggle.com/rtatman/did-it-rain-in-seattle-19482017
[8]. Chitra Desai, " Rainfall Prediction Using Deep Neural Network," unpublished
[9]. D. Zou, Y. Cao, D. Zhou, and Q. Gu, "Stochastic gradient descent optimizes over-parameterized deep relu networks," arXiv preprint arXiv:1811.08888, 2018.
[10]. A. Cauchy. Methodes generales pour la resolution des syst'emes dequations simultanees,. C.R. Acad. Sci. Par., 25:536–538, 1847.
[11]. AVRIM L. BLUM* AND RONALD L. RIVEST, " Training a 3-Node Neural Network is NP-Complete", Neural Networks, Vol. 5, pp. 117-127, 1992
[12]. Ruder, S. (2016). An overview of gradient descent optimization algorithms. *arXiv preprint* arXiv:1609.04747..
[13]. Bottou L. (2012) Stochastic Gradient Descent Tricks. In: Montavon G., Orr G.B., Müller KR. (eds) Neural Networks: Tricks of the Trade. Lecture Notes in Computer Science, vol 7700. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-35289-8_25
[14]. Duchi, J., Hazan, E., & Singer, Y. (2011). Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. Journal of Machine Learning Research, 12, 2121–2159.
[15]. Geoffrey Hinton Neural Networks for machine learning nline course. https://www.coursera.org/learn/neural-networks/home/welcome
[16]. Diederik P. Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. 2014. arXiv:1412.6980v9 (2014)