Mini Project 3: Momentum and Second Order Gradient Descent Methods

Siddhant Prakash
1211092724
sprakas9@asu.edu

***Abstract*—Report for mini-project 3 to study optimal settings for training a neural network.**

1. Introduction

In this mini-project we are provided with the YANN [1] toolbox using which, we train a multi-layer neural network on the MNIST [2] data set. The aim of the project is to explore and get an intuition of how various settings like momentum, learning rate and optimizers, affect the training of a neural network. With the standard architecture provided to us, we try to come up with a good combination of these settings, to obtain the best local minima through experiments. Learning to use the toolbox is also one of the goals of this mini-project.

2. Network Architecture & Training

The input to the network is the MNIST data set which has images of size 28X28. So, for a fully connected input, we get 784 nodes in input layer. The network consists of 2 dot product layers having 800 neurons each with a soft-max classifier. The MNIST dataset is for 10 classes (digits), hence, the number of classes is set to 10. Thus, the overall architecture for the network becomes, 784-800-800-10. We train all the network setting for 40 epochs covering 2 eras, with 20 epochs in each era, to maintain same architecture. The momentum coefficient at start is 0.6 and ends at 0.95 which stops increasing at epoch 30. The regularization constants are (0.0001, 0.0002).

3. Results

The network was trained in 27 (3X3X3) configurations of Momentum-Optimzer-Learning Rate combination. The test accuracy using the different parameters for the given architecture are shown in Table 1, Table 2 and Table 3.

	SGD	Adagrad	RmsProp
Nil - "False"	**70.96**	98.15	98.27
Polyak	71.57	98.09	**98.29**
Nesterov	85.79	98.11	98.09

TABLE 1. LEARNIN RATE - (0.05, 0.01, 0.001)

	SGD	Adagrad	RmsProp
Nil - "False"	**90.21**	97.37	93.49
Polyak	90.25	97.61	**97.69**
Nesterov	92.36	96.96	93.94

TABLE 2. LEARNIN RATE - (0.1, 0.05, 0.01)

	SGD	Adagrad	RmsProp
Nil - "False"	**17.73**	94.44	98.32
Polyak	21.69	94.66	98.33
Nesterov	26.87	96.1	**98.5**

TABLE 3. LEARNIN RATE - (0.01, 0.001, 0.0001)

Analysis. For learning rate, we can observe that starting with a high learning rate (0.1, 0.05, 0.01) although gives the best accuracy of 97.69, but still can not cross 98% mark which other networks, with a lower starting point, easily crossed.

In terms of momentum, both Polyak [3] and Nesterov [4] marks considerable improvement in test accuracy. While, Polyak provides the best accuracy in starting with higher learning rates, Nesterov provides the overall best test accuracy with lower learning rate start point and learning slowly, both using RMSProp.

With respect to optimization in learning rate, RMSProp [5] is clearly the best amongst all three. With SGD performing the worst in all momentum-learning rate combination, the need for optimizers is justified, specially in starting with a low learning rate. Adagrad [6] performs well, but not as well as RMSProp.

4. Discussion

As we can see, the best test accuracy was obtained for the combination of Nesterov accelerated gradient, with RMSProp optimizer for learning rate, with training using a learning rate in increment of (0.01, 0.001, 0.0001). Thus, we propose using Nesterov momentum with RMSProp optimizer, with a low learning rate provides best results. Although, it must be noted that using Polyak momentum with RMSProp optimizer prodces the second best test accuracy overall (by 0.17%), and the best with higher learning rates. Hence, using a Polyak-RMSProp combination is also not bad in general. The screen-shot of the test output is shown in Figure 1

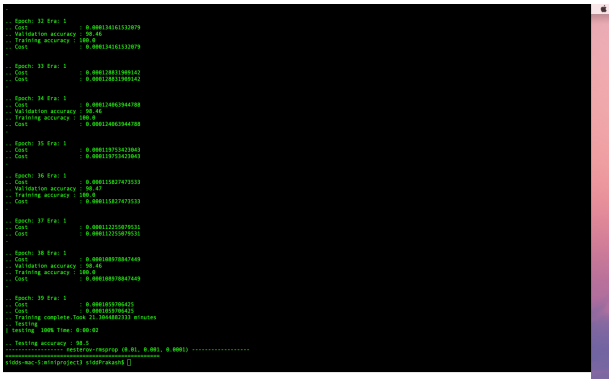

Figure 1. Output of Nesterov-RMSProp with LR (0.01, 0.001,0.0001)

References

[1] Yet Another Neural Network [YANN] Toolbox. https://github.com/ragavvenkatesan/yann

[2] LeCun, Yann, Corinna Cortes, and Christopher JC Burges. "The MNIST database of handwritten digits." (1998).

[3] Polyak, Boris T. "Some methods of speeding up the convergence of iteration methods." USSR Computational Mathematics and Mathematical Physics 4.5 (1964): 1-17.

[4] Nesterov, Yurii. "A method of solving a convex programming problem with convergence rate O (1/k2)." Soviet Mathematics Doklady. Vol. 27. No. 2. 1983.

[5] Tieleman, Tijmen, and Geoffrey 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).

[6] Duchi, John, Elad Hazan, and Yoram Singer. "Adaptive subgradient methods for online learning and stochastic optimization." Journal of Machine Learning Research 12.Jul (2011): 2121-2159.