# ORIE4741 Midterm Report - Analysis of the Learning Rate in Neural Networks

Henry C. Ogworonjo (hco23), Shishi Zhang (sz559)

## I. Introduction

The accuracy and efficiency of trained neural networks are greatly affected by the choice of the learning rate. Like it has been repeated over and over in the course ORIE 4741, learning rate controls the speed at which weights are adjusted with respect to the gradient of the loss function. Small learning rate takes a long time to converge while large learning rate could result in divergence problem. Therefore, optimizing learning rate is crucial for improving the performance of any learning algorithm. The objective of this project is to study existing algorithms for choosing learning rate, evaluate the performance of these algorithms in accuracy and computation time through experiments, and attempt to propose a new method that can identify an optimal learning rate for neural networks.

## II. Literature Review

Typically, the learning rate is configured randomly or set by the user according to intuition or past experiences. However, this method is not only time-consuming but also ineffective for proper choice of learning rate. Several algorithms have been proposed by researchers to address the problem [1,2]. Because of space constraints, we briefly describe only the baseline and 3 other learning rate algorithms.

- **Stochastic Gradient Descent (SGD):** SGD is the baseline algorithm. SGD updates the current weight using gradient multiplied by the learning rate, $\alpha$. The update for SGD is $w_{t+1} = w_t - \alpha \frac{\partial L}{\partial w_t}$
- **Momentum:** Momentum helps accelerate SGD in the relevant direction and dampens oscillations by adding a fraction of the update vector of the past time step to the current update vector. The update for Momentum is $w_{t+1} = w_t - \alpha V_t$, where $V_t = \beta V_{t-1} + (1-\beta)\frac{\partial L}{\partial w_t}$
- **Root mean square prop (RMSprop):** RMSprop is an adaptive learning rate algorithm that takes the exponential moving averages of the gradients. The RMSprop updates is $w_{t+1} = w_t - \frac{\alpha}{\sqrt{S_t+\epsilon}} \cdot \frac{\partial L}{\partial w_t}$ where $S_t = \beta S_{t-1} + (1-\beta)\left(\frac{\partial L}{\partial w_t}\right)^2$
- **Adaptive moment estimation (Adam):** Adam is a combination of momentum and RMSprop. The update equation is $w_{t+1} = w_t - \frac{\alpha}{\sqrt{S_t^*+\epsilon}} \cdot V_t^*$, where $V_t^* = \frac{V_t}{1-\beta_1^t}, \quad S_t^* = \frac{S_t}{1-\beta_2^t}, \quad V_t = \beta_1 V_{t-1} + (1-\beta_1)\frac{\partial L}{\partial w_t}, \quad S_t = \beta_2 S_{t-1} + (1-\beta_2)\left(\frac{\partial L}{\partial w_t}\right)^2$

In the following sections, we evaluate the performance of these algorithms on image and stock data.

## III. Performance of Learning Rate Algorithms on Image Data

### A. Data Description

We experimented on two popular image recognition data - the MNIST and the CIFAR-10. The MNIST is a database of handwritten digits that has a training set of 60,000 examples, and a test set of 10,000 examples. The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. It contains 50000 training images and 10000 test images. Since we only focus on evaluating the performance of the algorithms and there is no missing data in the dataset, the data cleaning is not necessary.

### B. Result

Since it is a classification problem, we evaluated the performance of the model using accuracy defined as: $Accuracy = (TP + TN)/(TP + TN + FP + FN)$. Table I shows the result obtained on a Dell Inspiron 7000 core i7 laptop. It can be seen that the Adam and RMSProp algorithms outperform the SGD and Momentum. In particular, the RMSprop achieved high accuracy while taking less time.

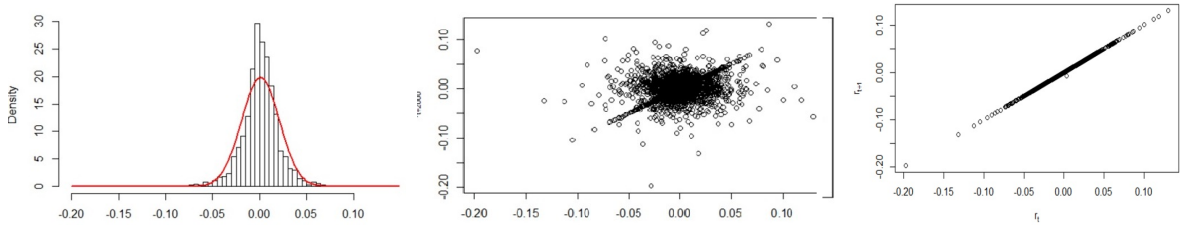
| CIFAR-10 | | | | MNIST | | | |
|---|---|---|---|---|---|---|---|
| Algorithm | Epochs | Val. Acc | Time(s) | Algorithm | Epochs | Val. Acc | Time(s) |
| SGD | 10 | 0.4831 | 13812.77 | SGD | 5 | 0.9493 | 6453.97 |
| Momentum | 10 | 0.4875 | 15866.79 | Momentum | 5 | 0.9423 | 7787.81 |
| Adam | 10 | 0.7655 | 15272.07 | Adam | 5 | 0.9796 | 6077.97 |
| RMSProp | 10 | 0.7640 | 13812.77 | RMSProp | 5 | 0.9882 | 6048.90 |

TABLE I: Performance of learning rate algorithm on CIFAR-10 and MNIST datasets

## IV. Performance of Learning Rate Algorithms on Stock Data

### A. Data Description and Exploration

We analyzed stock data gleaned from yahoo finance across several years. The raw data contains the stock's open price, close price, high, low, volume and adjusted close. Since we will like to eventually use an algorithm to predict stock trend, we posed and answered a question about stock returns. Specifically, we asked that given a stock data, can we validate the Black-Schole's assumptions that returns are mutually independent, identical and normally distributed? Fig. (1a) shows the histogram of the stock's return plotted alongside a normal distribution. It can be seen that the returns is not normally distributed like Black Schole's assumption suggests. Instead it's more negatively skewed because of extreme events. Fig. (1b) shows the correlation of returns between two time periods that are separated by 2000 trading days. The plot seems to validate Black-Scholes assumption that returns are independent. However, Fig. (1c) shows the correlations of returns between two successive days. It can be seen from the figure that there is some correlation between the returns of two successive trading days. It may therefore be concluded that short term correlation exists between returns.




(a) Histogram of stock returns over the period

(b) Correlation of 2000 trading days apart

(c) Correlation of two successive days

Fig. 1: Stock distribution and correlation

### B. Feature Engineering

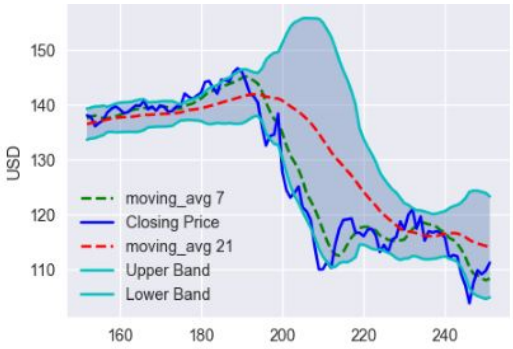
Since the original datasets only has few features, to incorporate more useful information, we added some popular technical indicators used by investors and performed feature engineering to potentially improve the accuracy of the model. The features we added include 7 and 21 days Moving Average, Exponential Moving Average, Momentum, Bollinger Bands and MACD. We also scaled the features using data normalization. Fig. 2 below shows part of these features.

### C. Result

We evaluated the performance of the algorithms using MSE and computation time. As it can be seen from Table II, the Adam and RMSProp algorithms outperform the SGD and Momentum in accuracy. However, unlike the image data, that Momentum takes shortest computation time, which comes as a surprise to us. One explanation could be that momentum works faster for the stock data while Adam and RMSProp work faster for the sparse data like the image data. We will research this in the future.

| | Adj Close | moving_avg21 | 12ema | upper_band | lower_band |
|---|---|---|---|---|---|
| **244** | 109.121124 | 116.276149 | 114.208759 | 122.184726 | 110.367572 |
| **245** | 107.112885 | 115.879328 | 113.117086 | 122.956628 | 108.802027 |
| **246** | 103.859131 | 115.370508 | 111.692785 | 124.120526 | 106.620489 |
| **247** | 107.547356 | 114.979478 | 111.055027 | 124.344086 | 105.614871 |
| **248** | 109.854912 | 114.750332 | 110.870394 | 124.332620 | 105.168045 |
| **249** | 109.130783 | 114.498980 | 110.602761 | 124.115053 | 104.882907 |
| **250** | 109.748711 | 114.288822 | 110.471369 | 123.930562 | 104.647082 |

(a) Generated Features from Stock Data




(b) Plot of Important Features

Fig. 2: Sample generated features

| Algorithm | Nos. of epochs | MSE | Time(s) |
|---|---|---|---|
| SGD | 100 | 0.0036 | 1178.62 |
| Momentum | 100 | 0.0043 | 658.64 |
| Adam | 100 | 0.0014 | 1994.69 |
| RMSProp | 100 | 0.0017 | 1180.62 |

TABLE II: Performance of learning rate algorithm on stock data

## V. Avoiding Overfitting

Since we will train all the models with neural network which has high chance to over-fit the data, we used the following techniques to prevent overfitting:

- For the image data, we performed data augmentation to increase the diversity of the dataset and reduce the chance of over-fitting. We also scaled the data during implementation.
- We also used $\ell_2$-regularization for the image data. For the stock data, we used dropout regularization where the neural network layers randomly remove nodes in the hidden layers.
- We imposed an upper limit on the number of runs thereby achieving early stopping.

## VI. Conclusion/Future Work

We have done some initial experiments with some learning algorithms on image and stock data. We performed basic data analysis and the progress made so far has been encouraging. For the future, we plan to do understand the reasons for the observed results. Specifically we will do the following:

- Perform more simulations and run for hundreds of epochs to clearly see how the algorithms perform in the long run. We will also explore more algorithms and include the algorithms we do not present in this mid-term report(e.g Nadam, Adagrad, NAG, Adamax etc).
- Research and hopefully improve the existing learning rate algorithms.
- Do a more elaborate feature engineering for all the datasets. For the stock data, we will make use of all important features to try to improve the accuracy of prediction. In addition to the MNIST, CIFAR-10 and stock data that are currently reported, we will also perform simulations with the CIFAR-100. The goal is to see how these algorithms perform in different datasets.
- Polish our final report. Present a more complete graphical representation of results for ease of interpretation and write a more comprehensive report which includes all details and references.

## VII. Reference

1. R. Sebastian, “An overview of of gradient descent optimization algos” 2016, ArXiv abs/1609.04747.
2. R.Yedida, S. Saha, “A novel adaptive learning rate scheduler for DNN.” 2019, ArXiv abs/1902.07399.