Learning Deep Neural Networks

Group 8, Team EnvyUs School of Engineering and Applied Science,
Ahmedabad University
Courses: Machine Learning, Algorithms and Optimization for Big Data
March 27, 2017

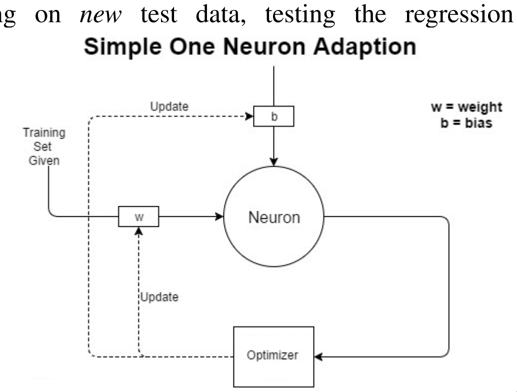
***Abstract*—This is a report of our study and implementation of Deep Neural Networks. Here we have demonstrated different uses of DNN as a 1. Classifier and 2. Predictor. We analyze the performance of implementations, based on different parameters. From this learning, we move toward the aim of our project, to build generative models via PPCA and Deep Neural Networks.**
***Index Terms*—DNN, PPCA, GM, Classification, Regression**

I. Introduction

Generative models are appreciated a lot by machine learning *Neural Networks*: Neural networks are attempts to give computers the ability to learn, process new information and provide output, functioning similar to the human brain. These networks have the ability to classify data, and to predict the outcome for new, unobserved input. *Deep Neural Networks* with a large number of nodes have the cability to process large dimensions of data.

II. Implementations

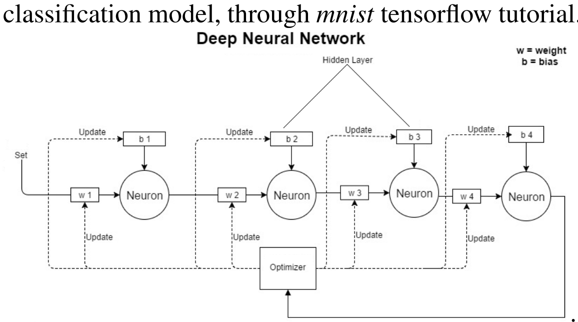
- We successfully trained a single neuron with $Y = WX + b$ (W :weight, b :bias) and observed it working on *new* test data, testing the regression model



.

- We partially implemented the training of a 3 layer neural network to perform $y = x^3$, on Tensorflow, and on Keras.

- We also trained a deep neural network for testing a classification model, through *mnist* tensorflow tutorial. [2]



.

III. Results

- Our Single neuron model works well as a predictor. Following is an observation table for some instances:

No.	Epoches	N_{train}/N_{test}	Learning Rate	Error
1.	500	1	0.001	0.10597
2.	500	1	0.01	0.00610882
3.	500	1	0.1	$2.484\text{x}10^{-7}$
4.	500	5	0.001	0.0029533
5.	500	5	0.01	0.00575019
6.	500	5	0.1	NaN
7.	500	10	0.001	0.00222705
8.	500	10	0.01	$7.32581\text{x}10^{9}$
9.	500	10	0.1	NaN
10.	1000	1	0.001	0.0834099
11.	1000	1	0.01	0.000621229
12.	1000	1	0.1	$1.51582\text{x}10^{-14}$
13.	1000	5	0.001	0.00239864
14.	1000	5	0.01	0.00175488
15.	1000	5	0.1	NaN
16.	1000	10	0.001	0.00190792
17.	1000	10	0.01	$7.99575\text{x}10^{10}$
18.	1000	10	0.1	NaN

Observations: A good value for learning rate is 0.1, if number of training samples is as much as of testing samples (12). For more training data, the error is lesser if epoches are more and the learning rate is average (14).

- The tensorflow example DNN (mnist) works well as a classifier. Following is an observation table with some parameters:

No.	Epoches	Batch Size	Accuracy
1	10	100	95.29 %
2	10	150	94.26 %
3	10	200	93.89 %
4	20	100	95.93 %
5	20	150	94.95 %
6	20	200	93.98 %

Observations: For this mnist database, a good batch size=200 and number of epoches=20, showing that our understanding of accuracy $\propto$ epoches holds correct.

IV. Conclusion

We can use neural networks for *regression* and *classification*. By observing performance of our implementations, we observed that accuracy improves with sample size, number of epoches and learning rate, whereas learning-time improves against accuracy.

References

[1] Simon Haykin, Neural Networks - A Comprehensive Approach, 2 Ed., Pearson Prentice Hall, 23-59
[2] Github | Tensorflow
[3] Mnist