ITCS 6156 Spring 2017
ASSIGNMENT : 2

SUPERVISED LEARNING

NEURAL NETWORKS

Rahul Rachapalli
800968032
rrachapa@uncc.edu

Neural Networks

(Supervised Learning)

Neural Network:

A technique in Machine Learning used to train a model based on the concept of a neural network. A neural network is formed using a set of Neurons that are connected biologically. Each neuron may be connected with many other neural networks. The function of a neuron is to combine all the inputs it has received, process it and produce an output if the output value is greater than the threshold value. A neural network is based on real numbers and the values typically lie in the range of 0.0 and 1.0.

The same concept is used in Artificial Neural Network. A neuron in this network is a perceptron that performs a simple summation function. There will be groups of perceptron in an artificial neural network that together produce a prediction based on the inputs provided. There are layers of perceptrons between the input and output with varied sizes. Each neuron has a weight that is associated with it and is used to emphasize its importance while its output is being fed as input to the other perceptrons. A perceptron is comparable to a neuron in the brain. They can be visualized as:

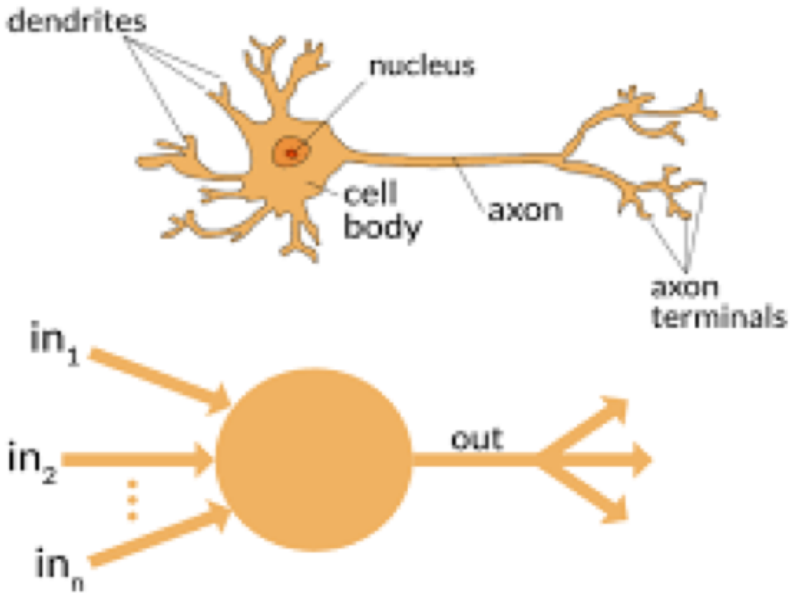


Fig1: Perceptron

The figure shows the relation between a perceptron and a neuron. When such perceptrons are connected, a neural network is formed.

The weights associated with each perceptron in a neural network should be different. If same weight is associated with every neuron, all of them compute the same values in the next stage until the end and we get the same input layer even after multiple computations. Hence, the weights should be different and small. Changing the weights by a small value results in a different neural network all together and results in varied predictions.

A sample neural network is as shown in fig 2. This network consists of an input layer with 6 perceptrons and an output layer with 2 perceptrons. There are 2 hidden layers with varied perceptron sizes. The number of layers is not fixed and can change based on the problem definition and requirements.

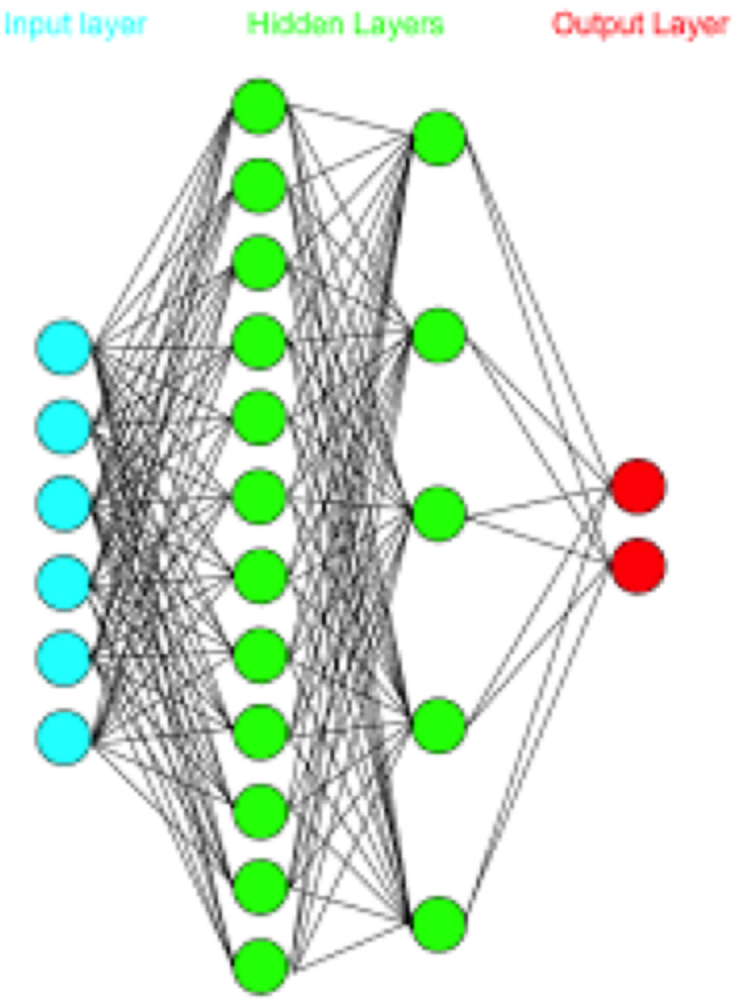


Fig 2: Neural Network

Data Sets

A dataset is a collection of data represented in as a single statistical matrix. Given are two data sets

1. *Amazon reviews - Sentiment Analysis*
2. *Optical recognition of handwritten digits*

1. Amazon reviews sentiment analysis dataset

This dataset consists of reviews of Amazon products which is a subset of large Amazon review collection. The data set has 3 columns and the type of data is as follows:

Name of the product	*-*	*String / text*
Review of the product	*-*	*String / text*
Rating given by the user	*-*	*Integer set {1,2,3,4,5}*

2. Optical recognition of handwritten digits

This data set consists of preprocessed normalized bitmaps of handwritten digits from a preprinted form. From a total of 43 people, 30 contributed to the training set and different 13 to the test set. 32x32 bitmaps are divided into non-overlapping blocks of 4x4 and the number of on pixels are counted in each block. This generates an input matrix of 8x8 where each element is an integer in the range 0..16. This reduces dimensionality and gives invariance to small distortions.

The dataset has 65 columns with 64 columns being the 8X8 matrix integer values and the last column being the digit that the 64 values represented.

Amazon Dataset Analysis and Prediction

Assumptions:

The name of the product is ignored as it is independent of the rating and ratings cannot be predicted using the name of the product. Hence, the rating depends on the reviews only.

Preprocessing:

Step 1: The reviews are separated and each review is converted into an array of words.

Step 2: Each word in the array is compared with stop words that are defined. If the word is present in the list of stop words, it is ignored and we proceed to the next word.

Step 3: If the word is not present in the list of stop words, we lemmatize the word and then stem it. At the end of step 3, we get a list of words that are lemmatized and stemmed.

Step 4: The words that are lemmatized and stemmed are converted into a single string and the word frequency is counted.

Step 5: The top 500 words or the whole list of words are considered as the attributes to dive the data.

Step 6: A sparse matrix is built based on the above attributes counting the occurrences of attributes in each review.

Stop Words:

The words 'the','a','an,'is','it' etc., are present in almost all the sentences and do not contribute to the decision process. All such words are called stop words and it is important to remove the stop words in pre-processing stage. So, we used some libraries like nltk to get a list of some pre-defined stop words.

Lemmatizing[2]:

The process of grouping together different forms of word to analyse it as a single word.

Stemming[3]:

The process of converting words into their stem or its root form. For sing, sang, sung, singing are represented in their base word form like sing*

Neural Network[1]:

With the sparse matrix built in the pre-processing stage and the ratings given by the user, the neural network to predict the rating of a review is built.

Prediction :

The rating of a review is predicted after the new review is pre-processed (lemmed and stemmed) and a sparse matrix is built for it using the same set of attributes that we decided to build the decision tree. The new sparse matrix is given as input to the neural network and the result is the predicted rating of the given review.

Results:

The experiment is conducted with different number of attributes that are considered after the pre-processing stage. 1000,800,750 attribute sizes have been running on the High Performance Cluster (HPC) for about 8 hours now and the results are not yet visible. I will update them in the comments section once the run is complete. The prediction rates are as follows:

Number of Decision Attributes	Prediction Percentage
100	81.44
200	90.56
250	90.71
300	90.81
350	90.97
400	90.96
450	91.12
550	91.09

Table 1: Effect of the number of decision attributes

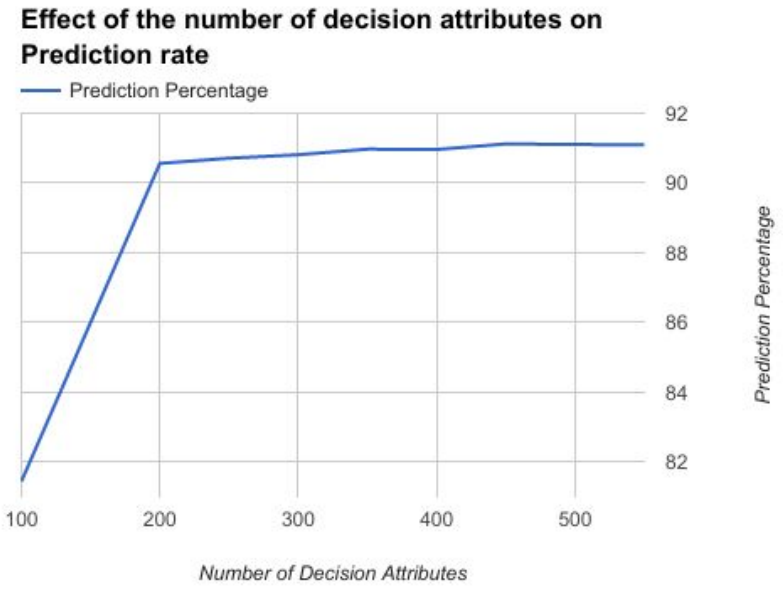


Fig 3: Effect of the number of decision attributes

The change in prediction rate with the change in the number of hidden layers are as follows:

Number of Layers	Prediction Percentage
500	90.86
650	90.96
750	91.03
1000	90.86

Table 2: Effect of the number of layers

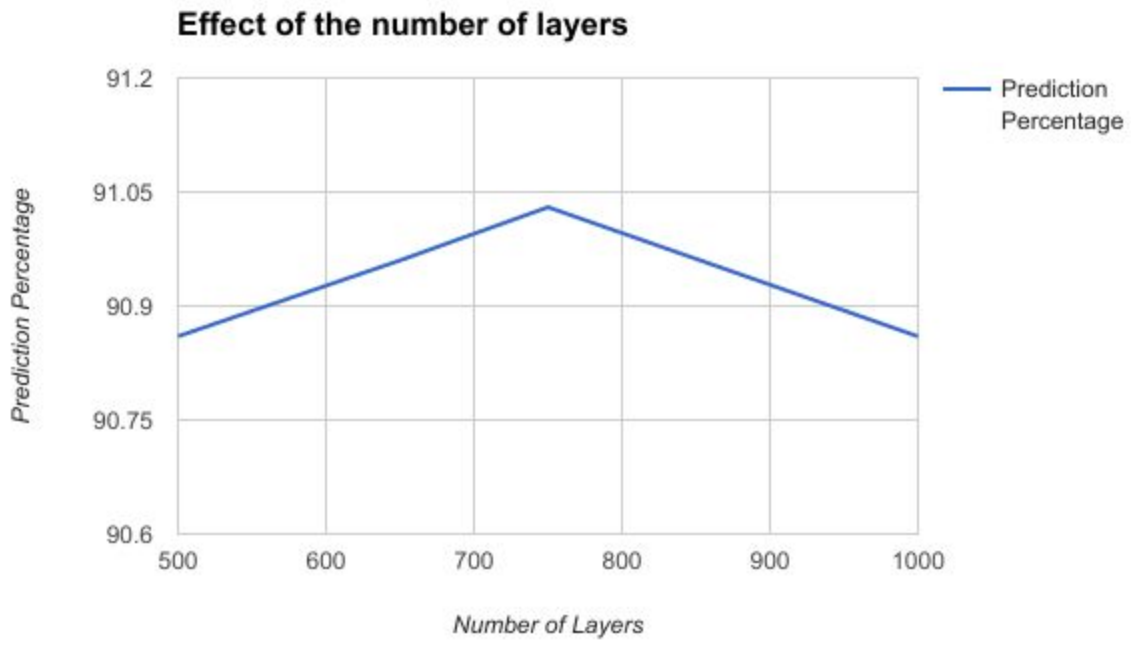


Fig 4: Effect of the number of layers

Experiments with activation and solver functions are also performed. The activation functions available in Scikit learn library are : **tanh. Identity, logistic** and **relu.** The corresponding solver functions are **lbfgs, adam** and **sgd.** The prediction results for partial Amazon data set of 100 training reviews, **500 attributes and 500 layers** are as follows:

Solver Function	Activation Function	Prediction Percentage
tanh	**lbfgs**	54.34
	adam	56.61

	sgd	57.59
identity	**lbfgs**	54.24
	adam	56.41
	sgd	57.09
logistic	**lbfgs**	52.78
	adam	56.68
	sgd	58.00
relu	**lbfgs**	54.74
	adam	57.19
	sgd	57.98

Table 3: Effect of different activation and solver functions

****These results are for neural nets trained with 100 reviews only.**
There was a memory error when I ran them on HPC.

Analysis:

From the above tables, it can be seen that 500 hidden layers gave up pretty good results with **logistic** as activation function and **sgd** as the solver function along with 500 decision attributes.

Sample Run:

```
[rrachapa@mba-i1 ~]$  qsub -N "Rahul_amazon_review_analysis1000" -q mamba
Amazon/NNsubmit_500.sh -l procs=8,mem=8gb,vmem=8gb,pmem=8gb

Setting up the system...
Importing Libraries
Defining grammar..
Preparing set of stop words
Analysing training dataset..
Loading the dataset...
Analysing testing dataset..
Loading the dataset...
Preprocessing the data set...
Predicting...
------------------------------------------------------------

Percentage of data that is predicted CORRECT is : 90.8653531809
Percentage of data that is predicted WRONG is : 9.13464681908
Prediction rate is 90.8653531809
```

```
------------------------------------------------------------
```

Amazon data set being very huge cannot be run on normal environments. Hence, I have formed a neural net with partial training set of 100 reviews and tried to predict the whole test data. The results are as follows:

```
[rrachapa@mba-i1 ~]$ /users/rrachapa/anaconda2/bin/python dLAmazonAnalysis.py
Setting up the system...
Importing Libraries
Defining grammar..
Preparing set of stop words
Analysing training dataset..
Loading the dataset...
Analysing testing dataset..
Loading the dataset...
Preprocessing the data set...

Prediction SCORE for 100 layers is 55.5667246251
------------------------------------------------------------
Prediction SCORE for 200 layers is 55.7514670724
------------------------------------------------------------
Prediction SCORE for 250 layers is 55.8900239078
------------------------------------------------------------
Prediction SCORE for 300 layers is 56.3111280156
------------------------------------------------------------
Prediction SCORE for 350 layers is 56.0693327537
------------------------------------------------------------
Prediction SCORE for 400 layers is 56.2785264073
------------------------------------------------------------
Prediction SCORE for 450 layers is 56.1508367746
------------------------------------------------------------
Prediction SCORE for 550 layers is 56.4089328407
------------------------------------------------------------
Prediction SCORE for 650 layers is 56.4007824386
------------------------------------------------------------

**************************************************************************

------------------------------------------------------------
ACTIVATION Function : tanh
------------------------------------------------------------
******************************
SOLVER Function : lbfgs
Prediction SCORE for 500 layers is 54.0942186481
******************************
SOLVER Function : adam
Prediction SCORE for 500 layers is 56.1263855684
******************************
SOLVER Function : sgd
Prediction SCORE for 500 layers is 57.1723538361
******************************
------------------------------------------------------------
ACTIVATION Function : identity
------------------------------------------------------------
******************************
SOLVER Function : lbfgs
Prediction SCORE for 500 layers is 53.9692458161
******************************
SOLVER Function : adam
Prediction SCORE for 500 layers is 56.0530319496
******************************
```

```
SOLVER Function : sgd
Prediction SCORE for 500 layers is 57.0120625951
*******************************
----------------------------------------------------------------
ACTIVATION Function : logistic
----------------------------------------------------------------
SOLVER Function : lbfgs
Prediction SCORE for 500 layers is 53.765485764
*******************************
SOLVER Function : adam
Prediction SCORE for 500 layers is 56.7295153228
*******************************
SOLVER Function : sgd
Prediction SCORE for 500 layers is 58.0009780483
*******************************
----------------------------------------------------------------
ACTIVATION Function : relu
----------------------------------------------------------------
SOLVER Function : lbfgs
Prediction SCORE for 500 layers is 54.3604651163
*******************************
SOLVER Function : adam
Prediction SCORE for 500 layers is 57.191371441
*******************************
SOLVER Function : sgd
Prediction SCORE for 500 layers is 57.949358835
*******************************
```

Digit Recognition Dataset Analysis and Prediction

Assumptions:

There are no missing values in the dataset and the given data is accurate.

Preprocessing:

The data is arranged in the form of a list, read from the file and is represented as a Data Frame.

Neural Network :

A Neural Network is built from the data frame that is obtained from the pre-processing step. This data frame itself acts as a matrix to build the decision tree.

Prediction :

A new dataset can be predicted using the decision tree built. The only step that is needed for it to be predicted is that there are **64** columns in the given integer data set. The pre-processing stage allows us to represent this data in the form of a list. The new list is input to the decision tree for prediction.

Results:

There are 64 columns and all of them are used as attributes to predict the results. The prediction probability for this dataset is **94.5%** with **tanh** as the Activation function, **adam** as the Solver function and a maximum of **20** hidden layers**.**

The results of various activation functions, solver functions and layers are as shown in the table:

Layers	Activation Function	Solver Function	Prediction Percentage
20	**tanh**	**lbfgs**	**93.37**
		adam	**94.10**
		sgd	**91.54**
	identity	**lbfgs**	**94.26**
		adam	**93.82**
		sgd	**93.54**
	logistic	**lbfgs**	**91.59**

		adam	95.21
		sgd	93.26
	relu	lbfgs	95.04
		adam	94.15
		sgd	94.37
50	tanh	lbfgs	94.10
		adam	95.71
		sgd	95.15
	identity	lbfgs	94.04
		adam	94.99
		sgd	94.93
	logistic	lbfgs	94.15
		adam	96.10
		sgd	94.99
	relu	lbfgs	95.43
		adam	96.88
		sgd	95.82
100	tanh	lbfgs	95.38
		adam	96.82
		sgd	95.15
	identity	lbfgs	94.32
		adam	94.82
		sgd	94.99
	logistic	lbfgs	94.71
		adam	96.99
		sgd	95.21

	relu	**lbfgs**	**95.49**
		adam	**96.77**
		sgd	**96.04**

Table 4 : Change in Prediction Rate with different solver and activation functions

The results with **200** layers have not deviated much from that of **100** hidden layers.

Analysis:

From the above table, we can see that the prediction rates for 100 and 200 hidden layers have a maximum value of **96.99%** with **logistic** activation function and adam as the solver function.

Note that all the above samples have been performed with time as the random seed. The effect of having a constant seed for the random function is not huge. The results lie close to that of the ones with random time seed.

The influence of hidden layers on accuracy with constant and random seeds are as shown below:

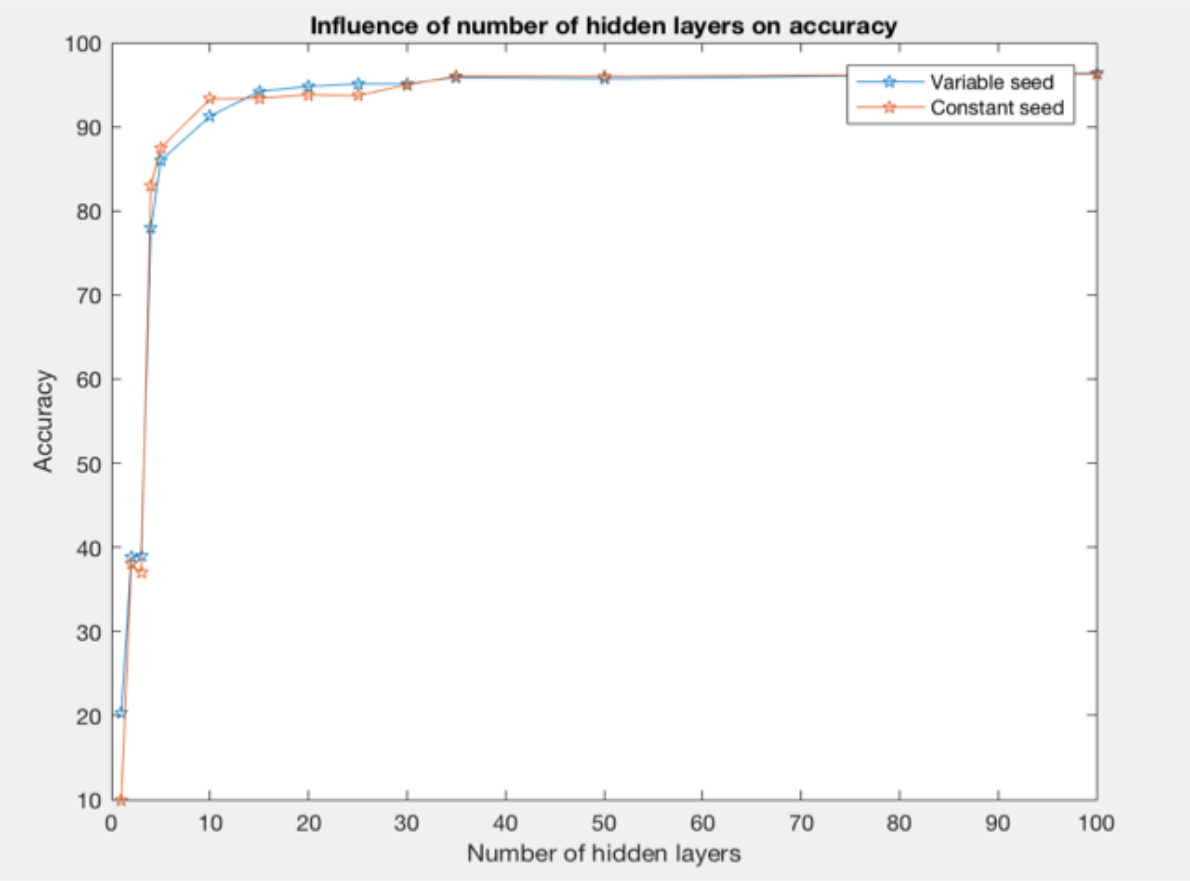
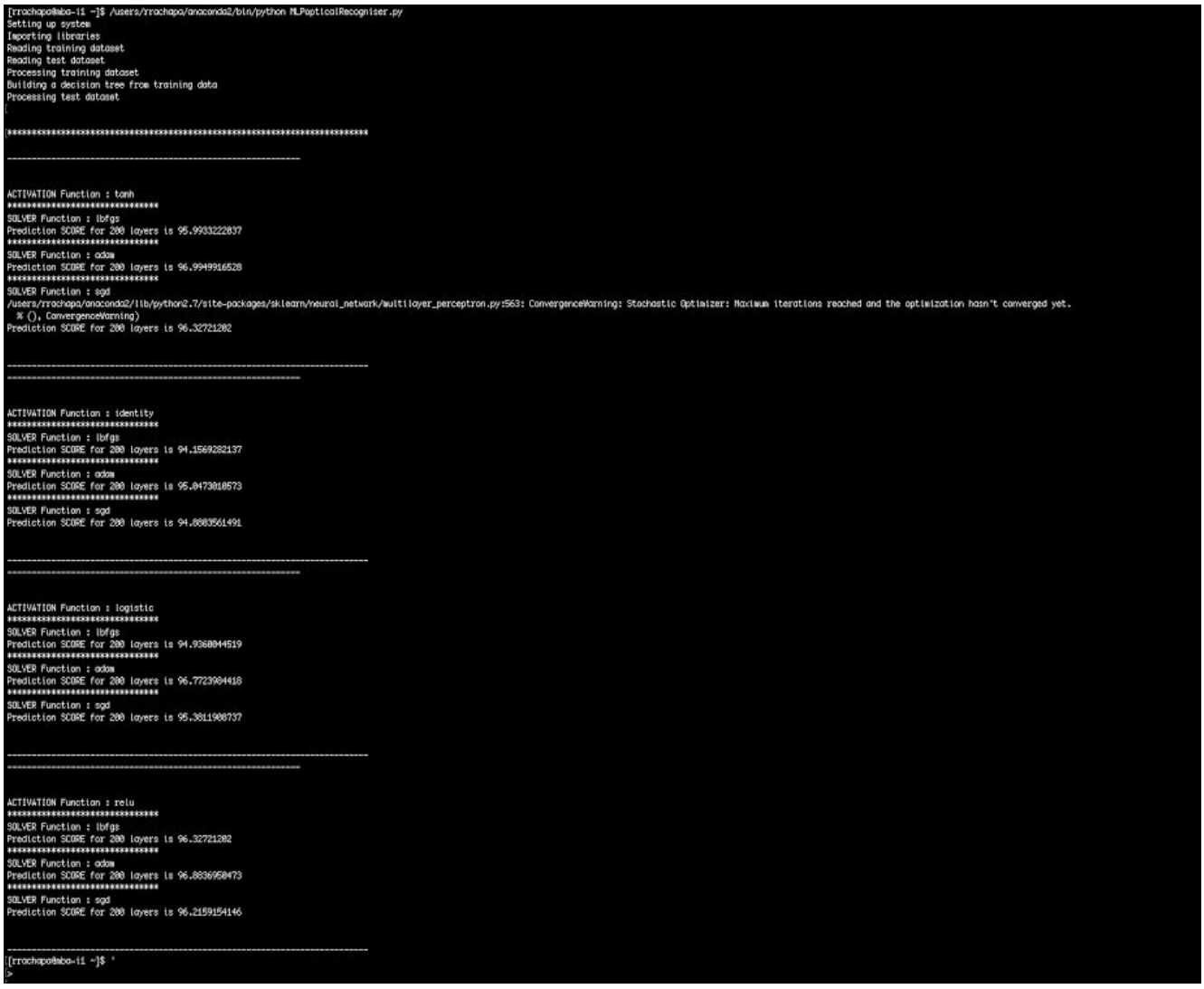


Fig 5: Influence of number of hidden layers on accuracy with constant and variable seeds

From the fig, it is clear that there is a possibility of local maximum. Hence, rather than stopping the execution at that stage, we have to proceed for a few more iterations with different sets of layers. After reaching a constant value, the prediction rate remains constant even with a change in number of hidden layers.

Sample Run:



Other Methods

We have tried to use other libraries like caffe and tensor flow but prefered tensor flow because of it vast library options and processing speeds. Caffe offered us great customizations using JSON file inputs that had the configuration. We could not explore much in this library due to time constraint.

We have also designed and tried our own neural network and it is included in the report as **ownNeuralNetDigitRecognition.py** This did not work out much yet, the prediction rate is **20.51.**

Acknowledgement

This assignment is a joint work with Sujal Vijayaraghavan.

The data exploratory questions have been answered in Assignment 1 and have not been included again.

References

1. ITCS6156_SLProject.pdf
2. https://en.wikipedia.org/wiki/Lemmatisation
3. https://en.wikipedia.org/wiki/Stemming
4. http://documents.software.dell.com/Statistics/Textbook/Text-Mining
5. http://archive.ics.uci.edu/ml/datasets/Optical+Recognition+of+Handwritten+Digits
6. https://en.wikipedia.org/wiki/Deep_learning
7. http://computing.dcu.ie/~humphrys/Notes/Neural/single.neural.html
8. http://www.inf.fu-berlin.de/inst/ag-ki/rojas_home/documents/1996/NeuralNetworks/K4.pdf
9. http://www.inf.fu-berlin.de/inst/ag-ki/rojas_home/documents/1996/NeuralNetworks/K3.pdf