EMPIRCAL INVESTIGATION OF NEURAL NETWORK USING BACK PROPOGATION

SIDDHARTH SHUKLA, PROFESSOR SANJAY GUPTA

Abstract— **Neural networks and AI (Artificial intelligence) are the most emerging technology for the computer to make things analyze and learn in a better way[1]. Back propogation is one of the most commonly used algorithm in the neural network for learning. We are proposing the back propogation algorithm for character (0-9) images for image recognition analyzing the way it works and also the impacts on the change of its paramaters as proposed in our previous papers[6] .This paper discusses how the neural network can be used to rcognize digits and also results and conclusion.**

***Index Terms*— Back propogation, cost function ,sigmoidal neuron.**

I. INTRODUCTION

Character recognition is becoming more and more important in the modern world. It helps humans ease their jobs and solve more complex problems. An example is handwritten character recognition.
Handwritten digit recognition is a system widely used in the United States.This system is developed for zipcode or postal code recognition that can be employed in mail sorting. This can help humans to sort mails with postal codes that are difficult to identify.
For more than thirtv years.researchers have been working on handwriting recognition. Over the past few years, the number of companies involved in research on handwriting recognition has continually increased. The advance of handwriting processing results from a combination of various elements, for example: improvements in the recognition rates, the use of complex systems to integrate various kinds of information,. and new technologies such as high quality high sped scanners and a cheaper and more powerful CPUs.
Some handwriting recognition system allows us to input our handwriting into the svstem. This can be done either by controlling a mouse or using a third-party drawing tablet. The input can be converted into typed text or can be left as an “ink object” in our own handwriting.

Pattern recognition system consists of two-stage process. The first stage is feature extraction and the second stage is classification. Feature extraction is the measurement on a population of entities that will be classified. This assists the classification stage by looking for features that allows fairly easy to distinguish between the different classes. Several different fearnres have to be used for classification. The set of features that are used makes up a feature vector, which represents each member of the population. Then. Pattern recognition system classifies each member of the population on the basis of information contained iii the feature vector. The following is an example of feature vectors that have been plotted on a graph.

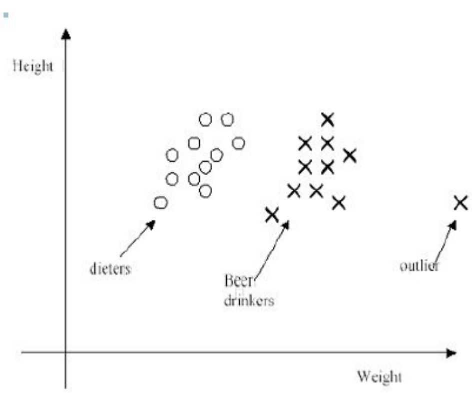


Figure 2.1 Two classes fully separated.

II. PATTERN RECOGNITION METHODS

1. Bayesian decision theory

The Bayesian decision theory is a system that minimizes the classification error.This theory plays a role of priori. This is when there is priority information about something that we would like to classify. For example. suppose we do not know much about the fruits in the conveyer belt. The only information we know is that 50 percent of the fruit in the conveyer belt areapples. and the rest of them are oranges. If this is the only information we have, then we can classify that a random fruit from the conveyer belt is apple. In this case. the a piiot information is the probability of either an apple or an orange is in the conveyer belt.

If we only have so little information, then we would have the following rule:

LetsDecide ‘apple’ if P(apple) > P(orange), otherwise decide ‘orange’
Here P(apple) is the probability of being an apple in the conveyer belt. This means that P(apple) = 0.8 (80 %). This is probably strange. because if the above nile is used.then we are classifying a random fruit as an apple. But if we use this

Manuscript received Dec 08, 2014
SIDDHARTH SHUKL, (VITS) Vindhya Institute of Technology & Science ,JABALPUR, INDIA
PROFESSOR SANJAY GUPTA, (VITS) Vindhya Institute of Technology & Science ,JABALPUR , INDIA

rule.we will be right 80 percent of the time.

This is a simple example and can be used to understand the basic idea of pattern recognition. In real life, there will be a lot more information given about things that we are trying to classify. For example. we know that the color of the apples is red.Therefore if we can observe a red fruit, we should be able to classify it as an apple.We can have the probabihtv distribution for the color of apples and oranges.

Let w, represent the state of nature where the fruit is an apple. let w0 represent the state of nature where the fruit is an orange. and let x be a continuous random variable that represents the color of a fruit. Then we can have the expression p(x|w) representing the density function for x given that the state of nature is an apple.

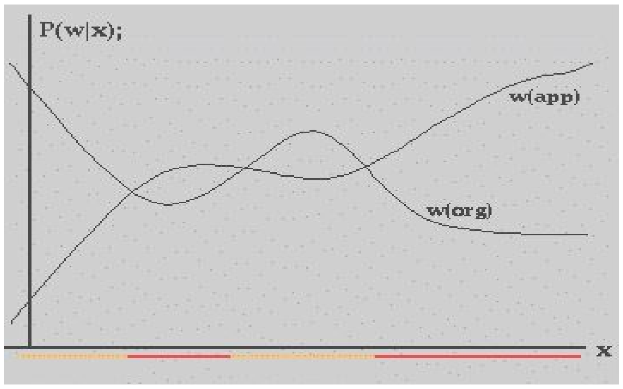
In a typical problem. we would be able to calculate the conditional densities p(xlwj) for j so it will be either an apple or an orange. We would also know the prior probabilities P(w) and P(w0). These represent the total number of apples versus oranges in the conveyer belt. Here we are looking for a formula that will tell us about the probability of a fruit being an apple or an orange just by observing a certain color x. If we have the probability, then for the given color that we observed, we can classify the fruit by comparing it to the probability that an orange had such a color versus the probability that an apple had such a color. If we were more certain that an apple had such a color, then the fruit would be classified as an apple. So. we can use

Baves formula. which states the following:
What the formula means is that using a priori information, we can calculate the a pasteriori probability of the state of nature being in state u; that we have given that the feature value x has been measured. So. if
we observe a certain x for a random fruitin the conveyer belt. then by calculating P(w |x) and P(wo |x) we would decide that the fruit is apple if the first value is greater than the second one. And if P(w0|x) is greater. then we would decide that the fruit is orange. So. the Bayesian decision rule can be stated as:

Decide wo if P(wo|x) > P(w|x) otherwise decide w
Since p(x) occurs on both sides of the comparison, the rule can also be equivalent to the following rule:

Decide w0 if p(x|wo)p(wo) > p(x|wo)p(wo) otherwise decide w.

The following graph shows the a posteriori probabilities for the two-class decision problem. For every x. the posteriors has to sum to 1. The red region on the x axes represents the values for x for which would decide as ‘apple’. The orange region represents values for x for which would decide as ‘orange’.



The probability that we would probably make an error is any miniimim of the 2

curves at any point, because it represents the smaller probability that we did not pick.

The formula for the error is the following:

P(error|x)=min(p(w0|x),p(w|x))

2. Nearest Neighbor rule.

The Nearest Neighbor (NN) rule is used to classify handwritten characters. The distance measured between the two character images is needed in order to use this rule. Without a priori assumptions about the distributions from which the training examples are drawn. the NN rule achieves very high performance. The rule involves a training set of both positive and negative cases. A new sample is classified by calculating the distance to the nearest training case. The sign of the point determines the classification of the sample. The following figure shows an example of NN rule:

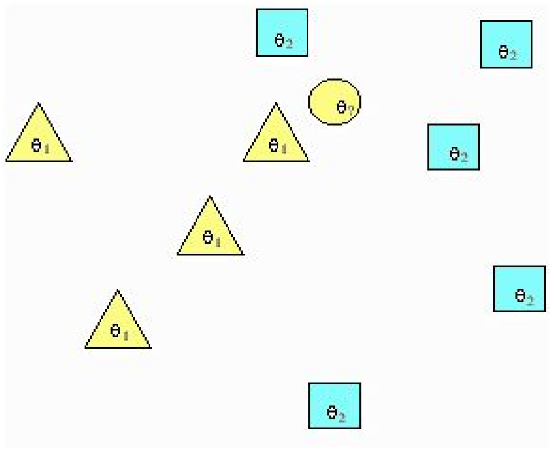


Figure 2.3 The Neural Network rule

In this example. there are two classes: θ1, which are the yellow triangles, and θ2 which are the blue squares. The yellow circle represents the unknown sample x.

We can see that the unknown sample’s nearest neighbor is from class θ1. therefore it is labelled as class .

When the amount of pre-classified points is large. it is good to use the majority vote of the nearest k neighbors instead of the single nearest neighbor. This method is called the k nearest neighbour k-NN) rule.
The k-NN rule extends the idea by taking and assigning the k nearest points the sign of the majority. It is common to choose k small (with respect to the number of samples) so that the points are closer to x to give accurate estimate of the the class of x. If the k values are large. it will help reduce the effects of noisy points within the training data set.

Also. large k values will minimize the probability of misclassifying

The following figure shows an example of k-NN rule with k value equal 3:

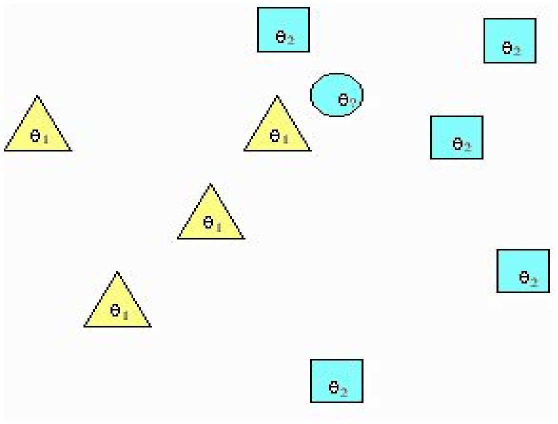


Figure 2.4 tne nearest K-neural network with K=3.

As before, there are two classes: θ1, which are the yellow triangles, and θ2
which are the blue squares. The blue circle represents the unknown sample x. We see that two of its nearest neighbors are from class 82 so it is labelled as class.

3. Linear Classification or Discrimination.
The goal of Linear Classification is to assign observations into the classes. This can be used to establish a classifier rule so that it can assign a new observation into a class. In another words. The rule deals with assigning a new point in a vector space to a class separated by a boundary. Linear classification provides a mathematical formula to predict a binary result. This result is a true or false (positive or negative)result or it can be any other pair of characters. In general. we will assume that our results are Boolean variable.

To do this prediction. we use a linear formula over the given input data. We refer this as inputs. The linear form is computed over the inputs and the result is compared against a basis constant. It is depending on the result of the comparison that we would be able to predict true or false. The following is the equation that can be stated as the discriminator:

a1 X1 ± a2 X2+ ± an Xn > X0

Here, a1,a2 are the variables that correspond to one observation. And X1. X2, together with X0 are the solution vector plus the basis constant. Corresponding to each of the input vector a. there is a variable b. This variable b is the dependent Boolean variable. We refer this as the output. We Normally have many m data points (a row vector of inputs a and the corresponding output b). For convenience, we place all the data in matrices and vectors. A denotes the matrix of the inputs and it is in the dimension of m by n. The in Boolean outputs will be stored in a vector B. Now, the problem of linear classification can be stated in terms of the matrices and vectors. For example: we want to find x and xo such that:

(Ax > x0) eguiv B

This equivalence means that every row of the left-hand-side is a relation. The relation for the given data will be true or false. This has to match the corresponding B value.

When we have more data points than columns, this is m > n. the problem does not have a solution. We cannot find a vector x that will satisfy all the true values.

Therefore we try to find a solution that can match vector B as good as possible. The best matching means that there is a minimum number of errors or there is a weighted minimum of errors. There will be a weight that is assigned to the true errors and another weight that is assigned to the false errors.

The idea of obtaining good classification is to be able to tise it to predict the output for new data points. Here. The discriminator is a prediction formula. A and B are the training data. and x and x0 are the parameters of the model fitted to the training data.

The following figure shows the main components of linear classification:

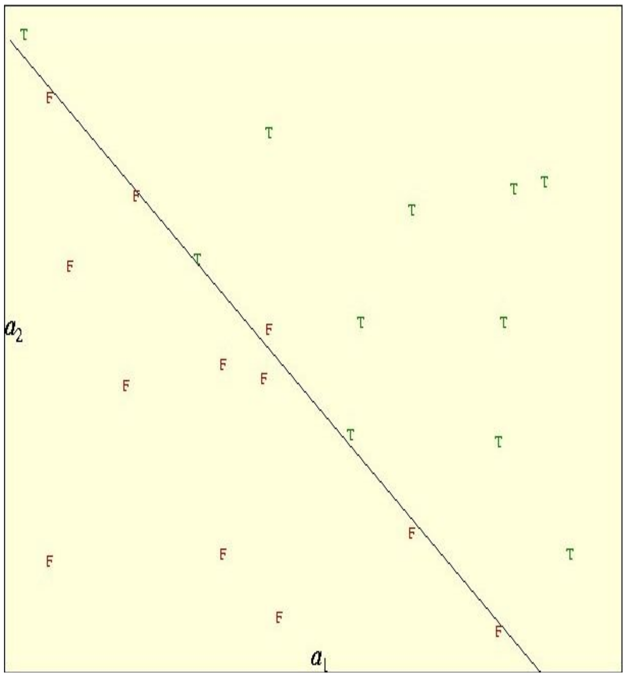


Figure 2.5 Linear Classification iii two dimensions.

Here, true are marked with a green T and false are marked with a red F. The inputs are two independent variables, a1 …a,. Here we try them to position the points in the plane. There are eleven Ts (positives) and twelve Fs (negatives).

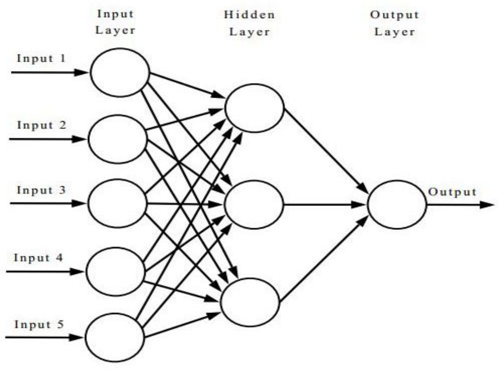
This linear classification in two dimensions is a straight line. The straight line drawn is a very good classification because the line separates most of the points correctly.

We can see that all the Ts are on one side of the graph and all the Fs. but one, are on the other side of the graph. Therefore this classification lias one "false positive" and no "false negatives". A false positive is a data point that is classified as positive but it is negative and vice versa for false negatives.

We will be using neural networks for image recognition. Image recognition plays an important role in pattern recognition. Neural network is a network based on the model of the brain.It consists of several neuron and layers. Which are interconnected with the help of weights. The neural networks are trained by adjusting of weights to get the desired output. The handwritten digits recognition of characters is already done using various ways having a high accuracy for devnagiri scripts but we will be using the method of backpropogation algorithm and neurons to train the network understaning the way the system do the learning. .We will use MNIST IMAGES(data set it is a data set of around 50000 images from handwritten experts.

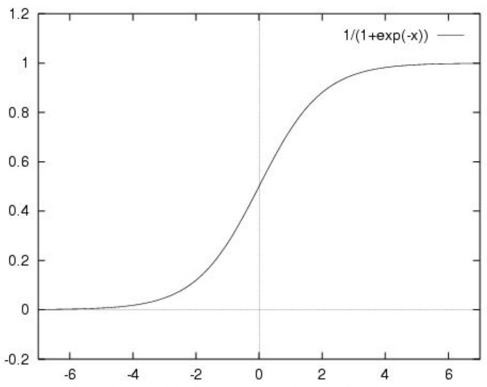
III. PREVIOUS WORK

It has been proposed in the research paper by DR RAMA KISHORE that the 3 layer network consisting of input, hidden and output layer can be used to used to perform the image recognition task . It consists of neuron interconnected by weight.They have used the below model ,however the output layers can change as per the requirement.They have also achieved good accuracy,we will try to analyze it in more details.Below is the given figure of the model consisting of input neuron,output neurons and hidden neurons.



IV. PROPOSED METHOLODOGY

We will be using neural network with 3 layers
1)Input layer
2)Hidden layer
3)O/p layer
We will be using sigmoid neuron which can takes input between 0 and 1 and generates the output on the basis of the sigmoidal function. Here is the graph for the the sigmoid function ,where sigmoid function is
$\sigma(z) \equiv 1/1+e^{-z}$ (1)



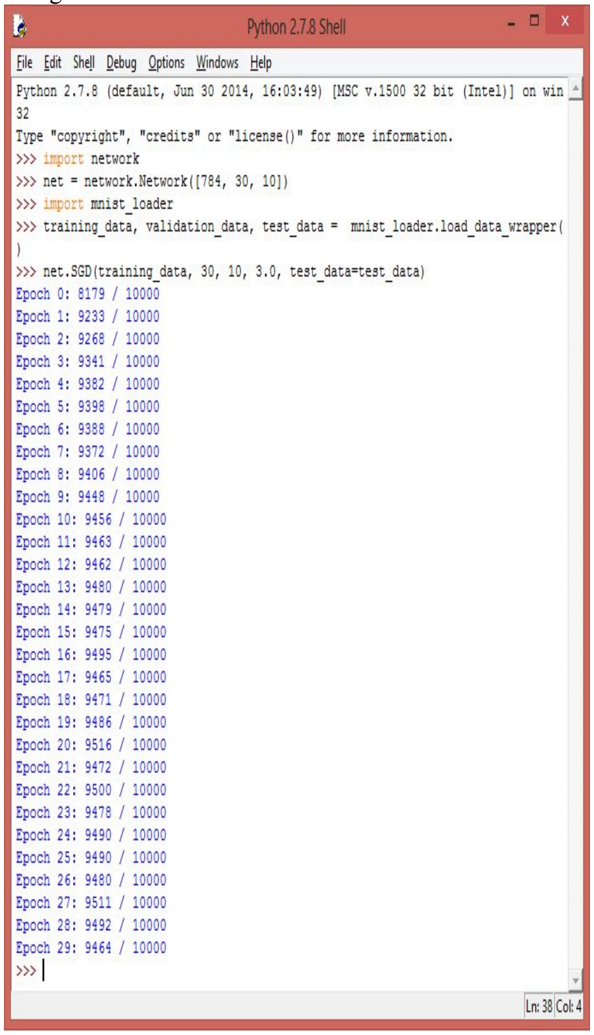
We are using the sigmoidal neuron since the input can vary as per the weights.
Below are the steps of methodology which are applied to train the network. We will be using stochastic gradient descent and back propogation using fixed certain epochs and training the images in the form of batches of m training examples .

1) We will initializes the weight and biases and create the network using neurons
2) The Output of the neuron will be calculated using below sigmoid function
$1+\exp(-\sum j\ wjxj-b))$ (2)
3) we will be using the below cost function and try to minimize it using stochastic gradient descent
$C(w,b) \equiv 1/2n\sum x\ \|y(x)-a\|^2$ (3)
Where w,b are the weights and the biases and the y(x) are the desired output and a is the actual output from the current state of the network.
4)**For each training example *x*:** Set the corresponding input activation $a1$, and perform the following steps:
Feedforward: For each $l=2,3,\dots,L$ compute $zx,l=wla,l-1+bl$ and $ax,l=\sigma(zx,l)$.
Output error $\delta x,L$: Compute the vector $\delta x,L$=delta $aCx\odot\sigma'(zx,L)$.
Backpropagate the error: For each $l=L-1,L-2,\dots,2$ compute $\delta x,l=((wl+1)T\delta x,l+1)\odot\sigma'(zx,l)$.
Where delta aC=(a,L−y)
Gradient descent: For each l=L,L−1,…,2 update the weights according to the rule $wl\rightarrow wl-\eta/m\sum x\ \delta x,l(ax,l-1)T$, and the biases according to the rule $bl\rightarrow bl-\eta/m\sum x\delta x,l$.
RESULTS AND SCREENSHOTS
We will be running the network 3 times to determine average accuracy
1st time

```
>>> import network
>>> net = network.Network([784, 30, 10])
>>> import mnist_loader
>>> training_data, validation_data, test_data = mnist_loader.load_data_wrapper()
>>> net.SGD(training_data, 30, 10, 3.0, test_data=test_data)

Epoch 0: 8179 / 10000
Epoch 1: 9233 / 10000
Epoch 2: 9268 / 10000
Epoch 3: 9341 / 10000
Epoch 4: 9382 / 10000
Epoch 5: 9398 / 10000
Epoch 6: 9388 / 10000
Epoch 7: 9372 / 10000
Epoch 8: 9406 / 10000
Epoch 9: 9448 / 10000
Epoch 10: 9456 / 10000
Epoch 11: 9463 / 10000
Epoch 12: 9462 / 10000
Epoch 13: 9480 / 10000
Epoch 14: 9479 / 10000
Epoch 15: 9475 / 10000
Epoch 16: 9495 / 10000
Epoch 17: 9465 / 10000
Epoch 18: 9471 / 10000
Epoch 19: 9486 / 10000
Epoch 20: 9516 / 10000
Epoch 21: 9472 / 10000
Epoch 22: 9500 / 10000
Epoch 23: 9478 / 10000
Epoch 24: 9490 / 10000
Epoch 25: 9490 / 10000
Epoch 26: 9480 / 10000
Epoch 27: 9511 / 10000
Epoch 28: 9492 / 10000
Epoch 29: 9464 / 10000
```

During first run



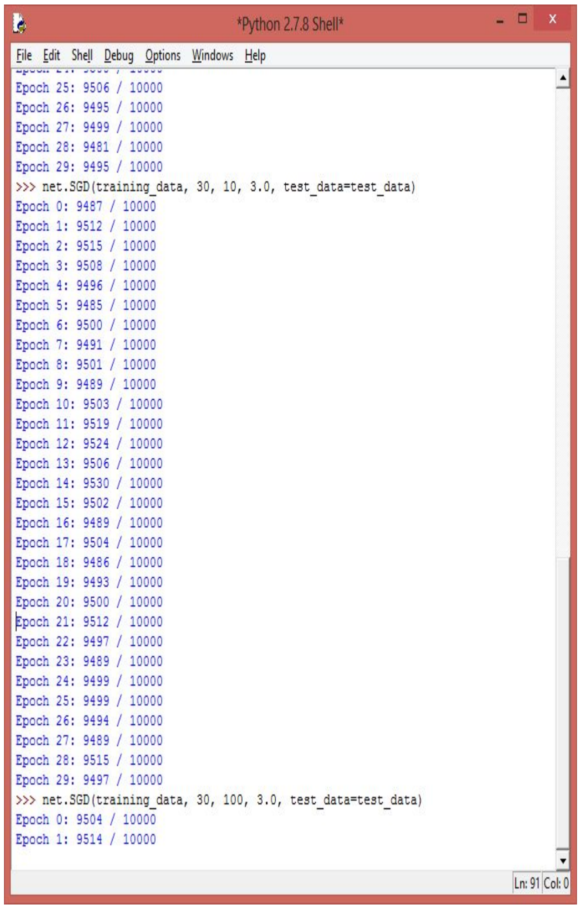
```
Python 2.7.8 (default, Jun 30 2014, 16:03:49) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> import network
>>> net = network.Network([784, 30, 10])
>>> import mnist_loader
>>> training_data, validation_data, test_data =  mnist_loader.load_data_wrapper(
)
>>> net.SGD(training_data, 30, 10, 3.0, test_data=test_data)
Epoch 0: 8179 / 10000
Epoch 1: 9233 / 10000
Epoch 2: 9268 / 10000
Epoch 3: 9341 / 10000
Epoch 4: 9382 / 10000
Epoch 5: 9398 / 10000
Epoch 6: 9388 / 10000
Epoch 7: 9372 / 10000
Epoch 8: 9406 / 10000
Epoch 9: 9448 / 10000
Epoch 10: 9456 / 10000
Epoch 11: 9463 / 10000
Epoch 12: 9462 / 10000
Epoch 13: 9480 / 10000
Epoch 14: 9479 / 10000
Epoch 15: 9475 / 10000
Epoch 16: 9495 / 10000
Epoch 17: 9465 / 10000
Epoch 18: 9471 / 10000
Epoch 19: 9486 / 10000
Epoch 20: 9516 / 10000
Epoch 21: 9472 / 10000
Epoch 22: 9500 / 10000
Epoch 23: 9478 / 10000
Epoch 24: 9490 / 10000
Epoch 25: 9490 / 10000
Epoch 26: 9480 / 10000
Epoch 27: 9511 / 10000
Epoch 28: 9492 / 10000
Epoch 29: 9464 / 10000
>>> |
```

2nd run and 3rd run

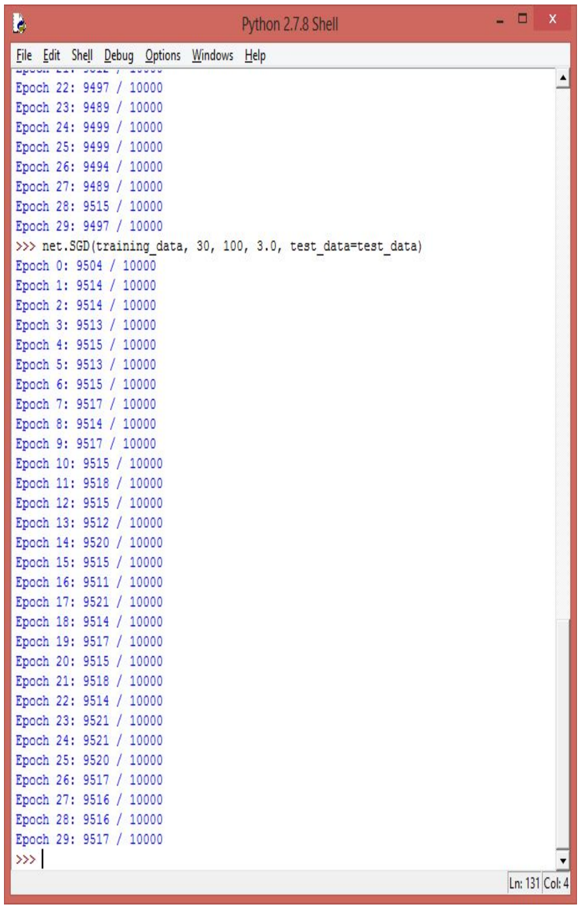
```
>>> net.SGD(training_data, 30, 10, 3.0, test_data=test_data)
Epoch 0: 9487 / 10000
Epoch 1: 9491 / 10000
Epoch 2: 9504 / 10000
Epoch 3: 9518 / 10000
Epoch 4: 9509 / 10000
Epoch 5: 9493 / 10000
Epoch 6: 9512 / 10000
Epoch 7: 9511 / 10000
Epoch 8: 9488 / 10000
Epoch 9: 9515 / 10000
Epoch 10: 9502 / 10000
Epoch 11: 9487 / 10000
Epoch 12: 9496 / 10000
Epoch 13: 9515 / 10000
Epoch 14: 9499 / 10000
Epoch 15: 9485 / 10000
Epoch 16: 9518 / 10000
Epoch 17: 9514 / 10000
Epoch 18: 9504 / 10000
Epoch 19: 9503 / 10000
Epoch 20: 9508 / 10000
Epoch 21: 9485 / 10000
Epoch 22: 9493 / 10000
Epoch 23: 9511 / 10000
Epoch 24: 9500 / 10000
Epoch 25: 9506 / 10000
Epoch 26: 9495 / 10000
Epoch 27: 9499 / 10000
Epoch 28: 9481 / 10000
Epoch 29: 9495 / 10000
```

For third observation:

```
>>> net.SGD(training_data, 30, 10, 3.0, test_data=test_data)
Epoch 0: 9487 / 10000
Epoch 1: 9512 / 10000
Epoch 2: 9515 / 10000
Epoch 3: 9508 / 10000
Epoch 4: 9496 / 10000
Epoch 5: 9485 / 10000
Epoch 6: 9500 / 10000
Epoch 7: 9491 / 10000
Epoch 8: 9501 / 10000
Epoch 9: 9489 / 10000
Epoch 10: 9503 / 10000
Epoch 11: 9519 / 10000
Epoch 12: 9524 / 10000
Epoch 13: 9506 / 10000
Epoch 14: 9530 / 10000
Epoch 15: 9502 / 10000
Epoch 16: 9489 / 10000
Epoch 17: 9504 / 10000
Epoch 18: 9486 / 10000
Epoch 19: 9493 / 10000
Epoch 20: 9500 / 10000
Epoch 21: 9512 / 10000
Epoch 22: 9497 / 10000
Epoch 23: 9489 / 10000
Epoch 24: 9499 / 10000
Epoch 25: 9499 / 10000
Epoch 26: 9494 / 10000
Epoch 27: 9489 / 10000
Epoch 28: 9515 / 10000
Epoch 29: 9497 / 10000
```



```
Epoch 25: 9506 / 10000
Epoch 26: 9495 / 10000
Epoch 27: 9499 / 10000
Epoch 28: 9481 / 10000
Epoch 29: 9495 / 10000
>>> net.SGD(training_data, 30, 10, 3.0, test_data=test_data)
Epoch 0: 9487 / 10000
Epoch 1: 9512 / 10000
Epoch 2: 9515 / 10000
Epoch 3: 9508 / 10000
Epoch 4: 9496 / 10000
Epoch 5: 9485 / 10000
Epoch 6: 9500 / 10000
Epoch 7: 9491 / 10000
Epoch 8: 9501 / 10000
Epoch 9: 9489 / 10000
Epoch 10: 9503 / 10000
Epoch 11: 9519 / 10000
Epoch 12: 9524 / 10000
Epoch 13: 9506 / 10000
Epoch 14: 9530 / 10000
Epoch 15: 9502 / 10000
Epoch 16: 9489 / 10000
Epoch 17: 9504 / 10000
Epoch 18: 9486 / 10000
Epoch 19: 9493 / 10000
Epoch 20: 9500 / 10000
Epoch 21: 9512 / 10000
Epoch 22: 9497 / 10000
Epoch 23: 9489 / 10000
Epoch 24: 9499 / 10000
Epoch 25: 9499 / 10000
Epoch 26: 9494 / 10000
Epoch 27: 9489 / 10000
Epoch 28: 9515 / 10000
Epoch 29: 9497 / 10000
>>> net.SGD(training_data, 30, 100, 3.0, test_data=test_data)
Epoch 0: 9504 / 10000
Epoch 1: 9514 / 10000
```



```
Epoch 22: 9497 / 10000
Epoch 23: 9489 / 10000
Epoch 24: 9499 / 10000
Epoch 25: 9499 / 10000
Epoch 26: 9494 / 10000
Epoch 27: 9489 / 10000
Epoch 28: 9515 / 10000
Epoch 29: 9497 / 10000
>>> net.SGD(training_data, 30, 100, 3.0, test_data=test_data)
Epoch 0: 9504 / 10000
Epoch 1: 9514 / 10000
Epoch 2: 9514 / 10000
Epoch 3: 9513 / 10000
Epoch 4: 9515 / 10000
Epoch 5: 9513 / 10000
Epoch 6: 9515 / 10000
Epoch 7: 9517 / 10000
Epoch 8: 9514 / 10000
Epoch 9: 9517 / 10000
Epoch 10: 9515 / 10000
Epoch 11: 9518 / 10000
Epoch 12: 9515 / 10000
Epoch 13: 9512 / 10000
Epoch 14: 9520 / 10000
Epoch 15: 9515 / 10000
Epoch 16: 9511 / 10000
Epoch 17: 9521 / 10000
Epoch 18: 9514 / 10000
Epoch 19: 9517 / 10000
Epoch 20: 9515 / 10000
Epoch 21: 9518 / 10000
Epoch 22: 9514 / 10000
Epoch 23: 9521 / 10000
Epoch 24: 9521 / 10000
Epoch 25: 9520 / 10000
Epoch 26: 9517 / 10000
Epoch 27: 9516 / 10000
Epoch 28: 9516 / 10000
Epoch 29: 9517 / 10000
>>>
```

```
>>> net.SGD(training_data, 30, 100, 3.0, test_data=test_data)
Epoch 0: 9504 / 10000
Epoch 1: 9514 / 10000
Epoch 2: 9514 / 10000
Epoch 3: 9513 / 10000
Epoch 4: 9515 / 10000
Epoch 5: 9513 / 10000
Epoch 6: 9515 / 10000
Epoch 7: 9517 / 10000
Epoch 8: 9514 / 10000
Epoch 9: 9517 / 10000
Epoch 10: 9515 / 10000
Epoch 11: 9518 / 10000
Epoch 12: 9515 / 10000
Epoch 13: 9512 / 10000
Epoch 14: 9520 / 10000
Epoch 15: 9515 / 10000
Epoch 16: 9511 / 10000
Epoch 17: 9521 / 10000
Epoch 18: 9514 / 10000
Epoch 19: 9517 / 10000
Epoch 20: 9515 / 10000
Epoch 21: 9518 / 10000
Epoch 22: 9514 / 10000
Epoch 23: 9521 / 10000
Epoch 24: 9521 / 10000
Epoch 25: 9520 / 10000
Epoch 26: 9517 / 10000
Epoch 27: 9516 / 10000
Epoch 28: 9516 / 10000
Epoch 29: 9517 / 10000
```

V. RESULTS AND CONCLUSION

It is been observed that we get the accuracy of about 94 percent when we use the medium range of neurons however it increases the accuracy slightly to 95 percent but the timing is also increased as the timing complexity increase due to the number of neurons.

VI. FUTURE WORK

Its hard to determine the parameters which we should take if we don't know about the learning like learning rate, number of neurons considering the complexity of the problem.

REFERENCES

[1] Dr. Rama Kishore, Taranjit Kaur, ",Backpropagation Algorithm: An Artificial Neural Network Approach for Pattern Recognition" International Journal of Scientific & Engineering Research, Volume 3, Issue 6, June-2012

[2] COURESA LECTURES BY STANDFORD

[3] Use of Artificial Neural Network in Pattern Recognition Jayanta Kumar Basu 1, Debnath Bhattacharyya 2, Tai-hoon Kim 2*," in INTERNATIONAL JOURNAL OF software engineering and its applications april 2010 vol4 VER2

[4] Xinyu Guo, Xun Liang and Xiang Li, "A Stock Pattern Recognition Algorithm Based on Neural Networks", Third International Conference on Natural Computation, Volume 02,2007.

[5] V. Chakravarthy, Z. Wu, A. Shaw, M. Temple, R. Kannan, and F. Garber, "A general overlay/underlay analytic expression for cognitive radio waveforms," in Proc. Int. Waveform [2] A.K. Jain, J. Mao, and K.M. Mohiuddin, "Artificial Neural Networks: A Tutorial", Computer, pp. 31- 44, Mar, 1996. Diversity Design Conf., 2007.

[6] USE OF NUERAL NETWORK FOR HANDWRITTEN DIGITS RECOGNITIONS
Author(s) : SIDDHARTH SHUKLA , PROFESSOR SANJAY GUPTA in IJERM INTERNATIONAL JOURNAL OF RESEARCH AND MANAGEMENT VOL01 ISSUE- 07 ,October 14

[7] [7} Zurada J.M., "An introduction to artificial neural networks systems", St. Paul: West Publishing (1992) *10+ Sergiy Stepanyuk, "Neural Network Information Tech-nologies of Pattern Recognition", MEMSTECH '2010, 20-23 April 2010, Polyana-Svalyava(Zakarpattaya),Ukraine.

[8] [8] Lin He, Wensheng Hou and Chenglin Peng from Biomed-ical Engineering College of Chongqing University on "Recog-nition of ECG Patterns Using Artificial Neural Network. *12+John Makhoul, "Pattern Recognition Properties of Neural Networks", BBN Systems and Technologies. *13+Kurosh Madani,"Artificial Neural Networks Based Image Processing & Pattern Recognition: From Concepts to Real-World Applications, Image, Signals and Intelligent Systems Laboratory.

[9] [9] L.Wallman, T.Laurell and C.Balkenius,L.Montelius, F.Sebelius, H.Holmberg, "Pattern recognition of nerve signals using an artificial neural network".

[10] [10]U.Bhattacharya, B.B.Chaudhari, "Handwritten Numeral Databases of Indian Scripts and Multistage Recognition of Mixed Numerals", IEEE Trans. on PAMI, Vol.31, No.3, 2009, pp.444-457. X. Li, R. Zhou, V. Chakravarthy, and Z. Wu, "Intercarrier interference immune single carrier OFDM via magnitude shift keying modulation," in Proc. IEEE Global

[11] [11] [15] http://en.wikipedia.org/wiki/Artificial_neural_network.