Handwritten Digit Recognition using Neural Networks and K-Nearest Neighbor (kNN)

Ketan Deshmukh (50097066)
Nitin Mendiratta (50097498)
Rahul Garg (50096987)

1. Abstract

This report explains, analyze and compare the task of classifying handwritten digits using Feedforward Neural Network and K-NN classifier methods. The training, validation and test sets were taken from the MNIST database. For neural network approach, backpropagation algorithm, specifically adapted to the problem, was used in the training phase to train the model. We used regularization to avoid the problem of over fitting. And for K-NN classifier, model is tested for different values of k to analyze the performance and accuracy of model.

2. Neural Network

Neural networks are typically organized in layers. Layers are made up of a number of interconnected '*nodes*' which contain an '*activation function*'. Patterns are presented to the network via the '*input layer*', which communicates to one or more '*hidden layers*' where the actual processing is done via a system of weighted '*connections*'. The hidden layers then link to an '*output layer*' where the answer is output as shown in the graphic below.

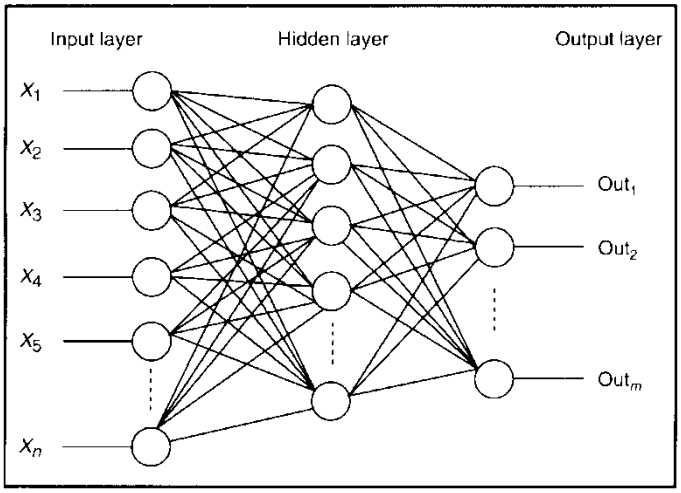


Figure 1

Neural Network Package:

The 3 layer network is chosen with 784 input features, 50 hidden nodes and 10 output unit to classify 10 digits.

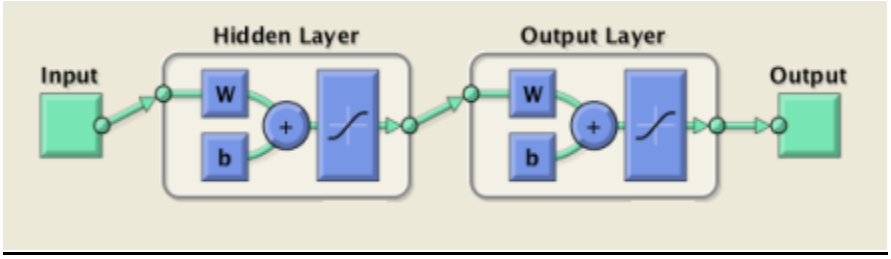


Figure 2

The Neural Network is implemented mainly in two main steps: *'Feed Forward'* and *'Backpropagation'*. First, we compute the probability that the given input vector belongs to a particular digit. Then we compare the answer we got with the actual answer and calculate the error. In the Backpropagation step, we do the backward propagation of the error and make an appropriate adjustment to our connection weights.

Output of neural network for digit 0:

1	0.8709	0.0138	0.0314	0.0262	0.0068	0.0304	0.0295	0.0119	0.0385	0.0297
2	0.6401	0.0362	0.0186	0.0114	0.0083	0.0213	0.0435	0.0118	0.1233	0.0412
3	0.7733	0.0262	0.0269	0.0096	0.0101	0.0408	0.0499	0.0120	0.0449	0.0131
4	0.8445	0.0084	0.0301	0.0207	0.0071	0.0333	0.0510	0.0201	0.0337	0.0343
5	0.8342	0.0175	0.0272	0.0123	0.0030	0.0226	0.0466	0.0133	0.0917	0.0251
6	0.7702	0.0106	0.0199	0.0285	0.0024	0.0321	0.0317	0.0435	0.0412	0.0204
7	0.8156	0.0109	0.0182	0.0192	0.0025	0.0421	0.0422	0.0274	0.0520	0.0222
8	0.8519	0.0091	0.0435	0.0312	0.0033	0.0223	0.0578	0.0188	0.0317	0.0195

Advantages

- Neural Networks are quite simple to implement.
- Depending on the nature of the application and the strength of the internal data patterns you can generally expect a network to train quite well
- Neural Network can be applied to problems where the relationships may be quite dynamic or non-linear.
- A neural network learns and does not need to be reprogrammed.

Disadvantages

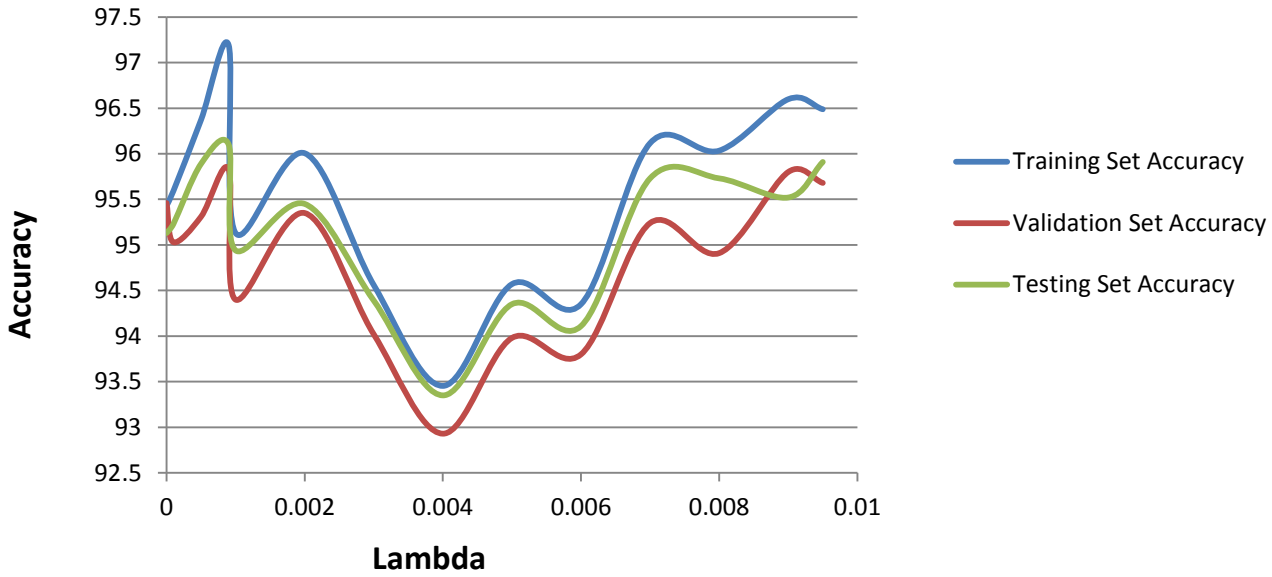
- Neural networks cannot be retrained. If you add data later, this is almost impossible to add to an existing network.
- The neural network needs training to operate.
- Requires high processing time for large neural networks Depending on the training data a Neural Network could suffer from problem of 'Overfitting' or 'Underfitting'.

2.1. Tuning Hyper-parameters

Its goal is to tweak the hypermeters, i.e. hyperparameter optimization, in such a way to obtain good generalization and regularization (avoid overfitting problem).

a) **Lambda(regularization parameter)**

Varying the value of lambda helps regularising the model and thus ensures there is no problem of overfitting. At λ= 0.0009, maximum accuracy of approx **97.3%** is observed. Following is the variation of accuracy with lambda(**λ**).

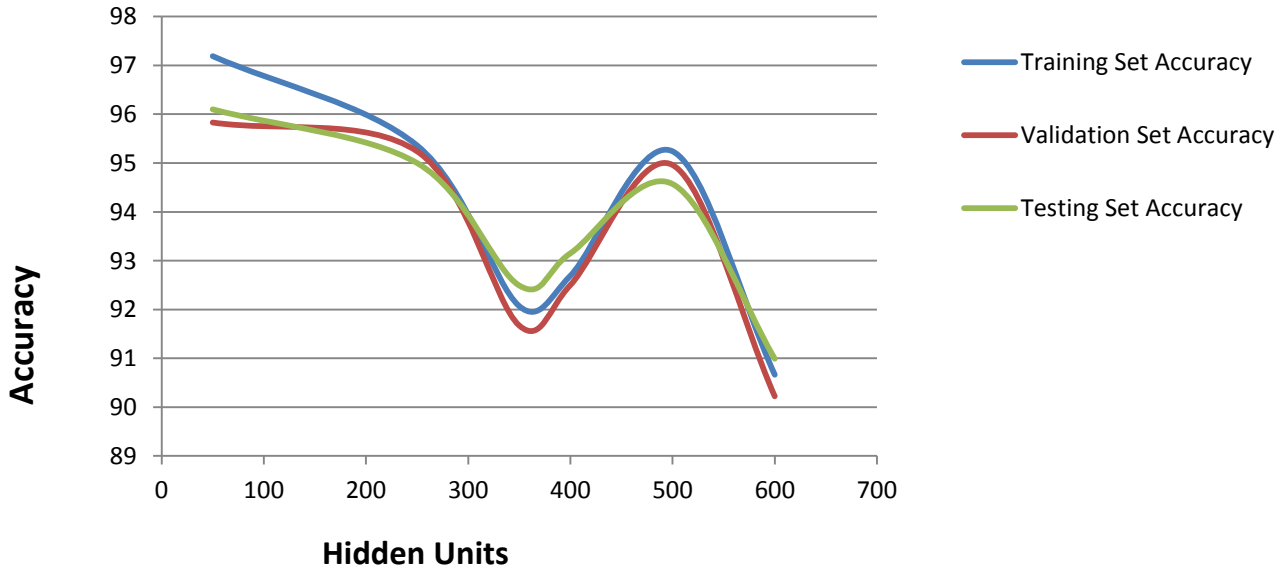


b) No of iterations (Maxiter)

As we increase number of iterations for convergence, we observe that accuracy is increased but to a particular extent and then it becomes stable. At **Maxiter=300**, we get the better test results with accuracy close to **97.3%.**

c) Hidden nodes (n_hidden)

Number of hidden units in our neural network. With the increase in the number of hidden units we observed a decrease in accuracy. We get a maximum of accuracy at **n_hidden=50.**



3. K-NN Classifier

K-NN is a non-parametric data distribution algorithm, also known as lazy algorithm. It means that k-NN does not make any assumptions on the underlying distribution of data i.e. it does not use the training data points to do any generalization. In other words, there is a very minimal or no training phase involved. K-NN makes decision based on the entire training data set. In order to implement k-NN, we have used MATLAB's inbuilt 'knnClassify' function.

Advantages

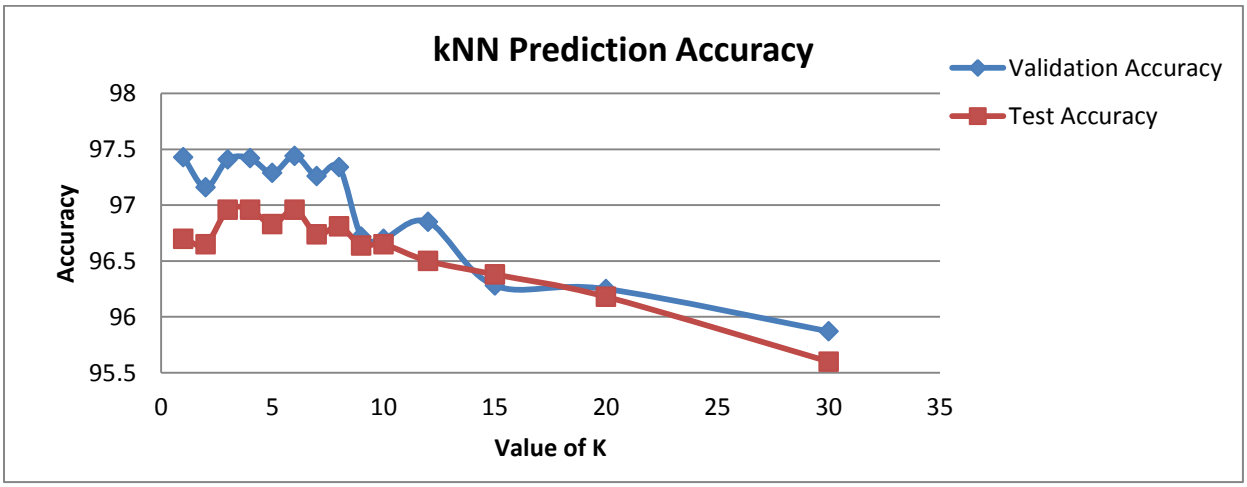
- It is robust to noisy training data
- Quite effective when provided a sufficiently large set of training data.
- It can smooth out the impact of isolated noisy training examples by taking the weighted average of the k neighbors nearest to the query point

Disadvantages

- One practical issued in applying kNN is that distance between instances is calculated based on all attributes of the instance. As attributes can be irrelevant, the result obtained could sometimes be misleading
- Attributes can have widely different ranges (In this case we may prefer doing Normalization or Standardization) or they can be redundant.

3.1. Tweaking value of k

With different value of k, accuracy varies and gradually decreases. And we got the maximum accuracy with k=6. Following is the graphical representation of k with accuracy.



4. Comparison of performance for both classification methods

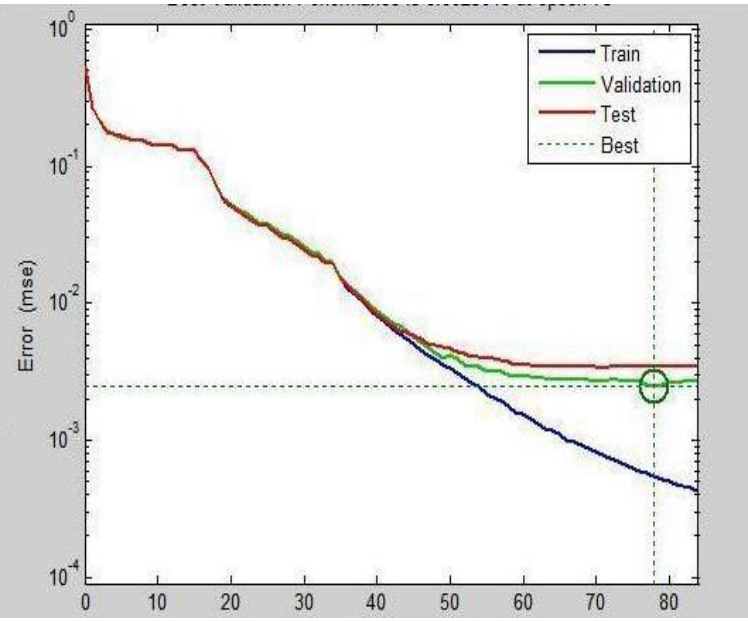
After an overall estimation and analysis of the performance of both classifiers, we report here the results. Below results are obtained with below optimized values of hyperparameters and k:

- Neural Network with λ=0.0009, Maxiter=300 and n_hidden=50.
- K-NN with k=6

	Neural Network	K-NN
Training Accuracy	99.99%	97.44%
Test Accuracy	97.20	96.66%
Training Error	0.01%	3.56%
Test Error	2.80%	3.04%
Learning Time	62.77 sec	173.45 sec

.

Cost variation in Neural Network: Error keeps on decreasing with each iteration till a specific point where it is minimum. Following is the graphical representation of variation of cost with each iteration of gradient descent convergence.



5. Conclusion

- Learning time for Neural Network is lesser compared to the learning time for k-NN. K-NN, being a lazy algorithm takes more time to learn compared to a NN. In fact, we observed the fact that running time for k-NN was almost 3 times greater than the running time of a neural network for the same set of inputs.
- In case of k-NN, accuracy seems quite constant for varying numbers of 'k'. Although we achieved maximum accuracy for the value of k = 6.