CLASSIFICATION TECHNIQUES

ARTIFICIAL NEURAL NETWORKS

&

K NEAREST NEIGHBOURS

AUTHORS:

ANGAD GADRE

HARISH MANGALAMPALLI

RAJARAM RABINDRANATH

Contents

OBJECTIVE

Compare and contrast two classifications techniques; Artificial Neural Networks and K-Nearest Neighbor by using them to classify handwritten digits.

PREPROCESSING

1. Preprocess the data, so that we have three data sets (training set, validation set, testing set) for the classification using neural networks and two data sets (training set and testing set) for K-NN classifier.
2. We proceed to do feature selection and here is how we do it. Feature selection allows for removal of non-discriminative features so the machine learning algorithm can learn a model faster and more accurately. The first pre-processing step we apply for performing feature selection is summing up individual cell values for each of the 784 features for all training examples and removing those features for which the sum is zero. At the end of this step we have 717 features remaining. We then proceed to find standard deviation of all features and purge those columns which exhibit minimum standard deviation

EXPERIMENTS

Following are the list of experiment performed. We had introduced randomness in the assignment of records to the training and validation datasets. This randomness was introduced to observe the effects of different training examples on the parameter that we set out to tune and to also find for ourselves if and how the choice of examples for training affect the prediction accuracy.

Experiment 1: Find the optimal value of lambda – **Finding Regularization parameter**

- Number of hidden units (constant): **50**
- Range of lambda: **0-800**
- Step Size: **20**

Experiment 2: Find the optimal number of hidden units – **Finding Right number of hidden units**

- Lambda: **20**
- Range of hidden units: **31-60**
- Step Size: **1 (31 - 40) and 2 (42 - 62)**

Experiment 3a: Find the optimal value of k for K-NN

- K: **1-10**
- Distance measure: **Cosine**
- Step Size: **1**
- No tie-breaking rule

Experiment 3b: Find the optimal value of k for K-NN

- K: **1-10**
- Distance measure: **Cosine**
- Step Size: **1**
- Used within class sum of distance for tie-breaking

HOW TO CHOOSE HYPER PARAMETERS

Hyper parameters i.e the regularization parameter and count of hidden nodes can be chosen using the following truths:

- Regularization parameter directly impact the accuracy of predictions made by an Artificial Neural Network – **Experiment 1**
    - We can find the right value of lambda by looking at the validation accuracy of ANN's learned model for different values of lambda and choosing that value of lambda for which one has the highest validation accuracy
    - Lambda values also have an impact on the time taken by ANN to learn the right weights
- Hidden Nodes impact the speed at which an ANN learns the weights to rightly classify an example -- **Experiment 2**
    - The ANN shall try and learn the right weights and the time taken to learn is a function of the number of hidden nodes
    - We find the right number of hidden nodes by running ANN for different hidden node count and while keeping the lambda value static and choose the count for which the time taken to learn the weights is minimum provided that the accuracy of predictions is in the right range
    - One needs to be careful not to associate accuracy of predictions with the count of hidden nodes as in accuracy is primarily a function of lambda so one must find the best value of lambda ("best" is defined above) and then proceed to find the count of hidden nodes for which the time taken to learn is minimum [1]

CLASSIFICATION TECHNIQUES PERFORMANCE COMPARISION

Experiment 1: Find optimal value of lambda

Lambda vs Accuracy & Time Taken

Lambda	Train Accuracy	Validation Accuracy	Test Accuracy	Time in Minutes
0	93.93%	93.40%	93.60%	25.94523008
20	93.90%	93.43%	93.82%	23.04805113
40	93.16%	92.57%	93.19%	25.43447837
60	92.89%	92.35%	92.92%	24.88755522
80	92.25%	91.96%	92.53%	24.36446502
100	91.00%	91.00%	92.00%	48.91666667
150	91.00%	91.00%	92.00%	44.81666667
200	91.00%	90.00%	91.00%	44.85

250	90.00%	90.00%	91.00%	46.68333333
300	90.00%	90.00%	91.00%	631.4333333
350	90.00%	89.00%	90.00%	77.08333333
400	90.00%	89.00%	90.00%	76.88333333
450	89.00%	89.00%	90.00%	69.91666667
500	89.00%	88.00%	89.00%	76.86666667
550	89.00%	88.00%	89.00%	72.86666667
600	89.00%	88.00%	89.00%	67.95
650	88.00%	88.00%	89.00%	58.65
700	88.00%	88.00%	89.00%	62.56666667
750	88.00%	87.00%	88.00%	52.86666667
800	87.00%	87.00%	88.00%	54.2

Table 1: Variation of prediction accuracy and time with lambda

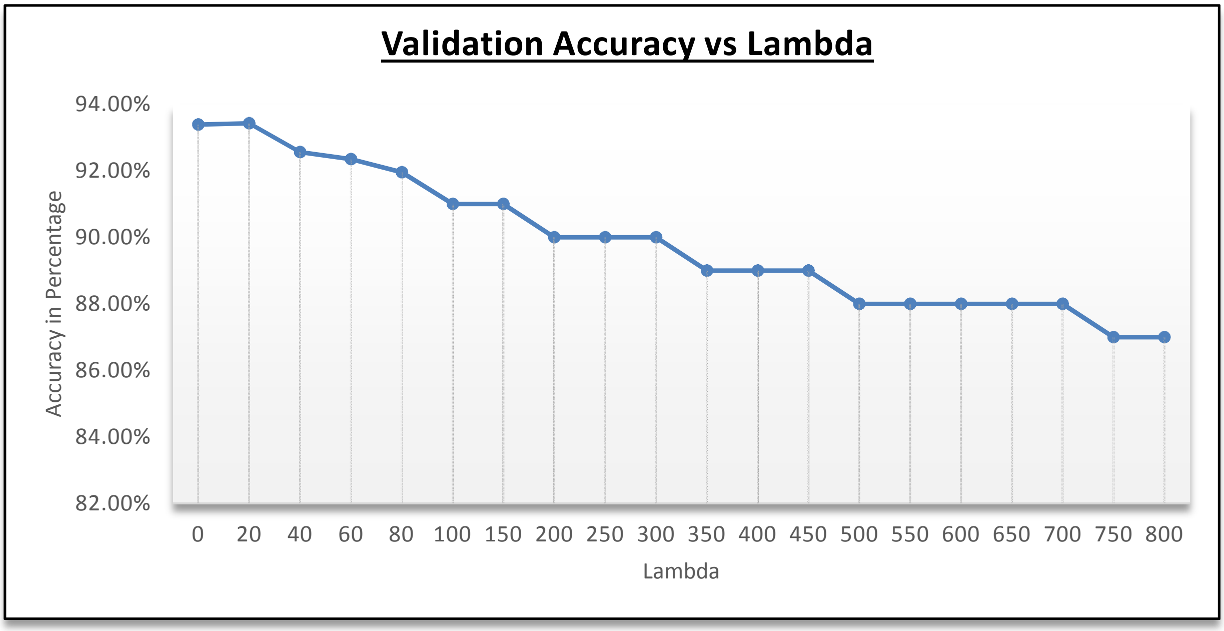

Figure 1: Plot of Validation accuracy against Lambda

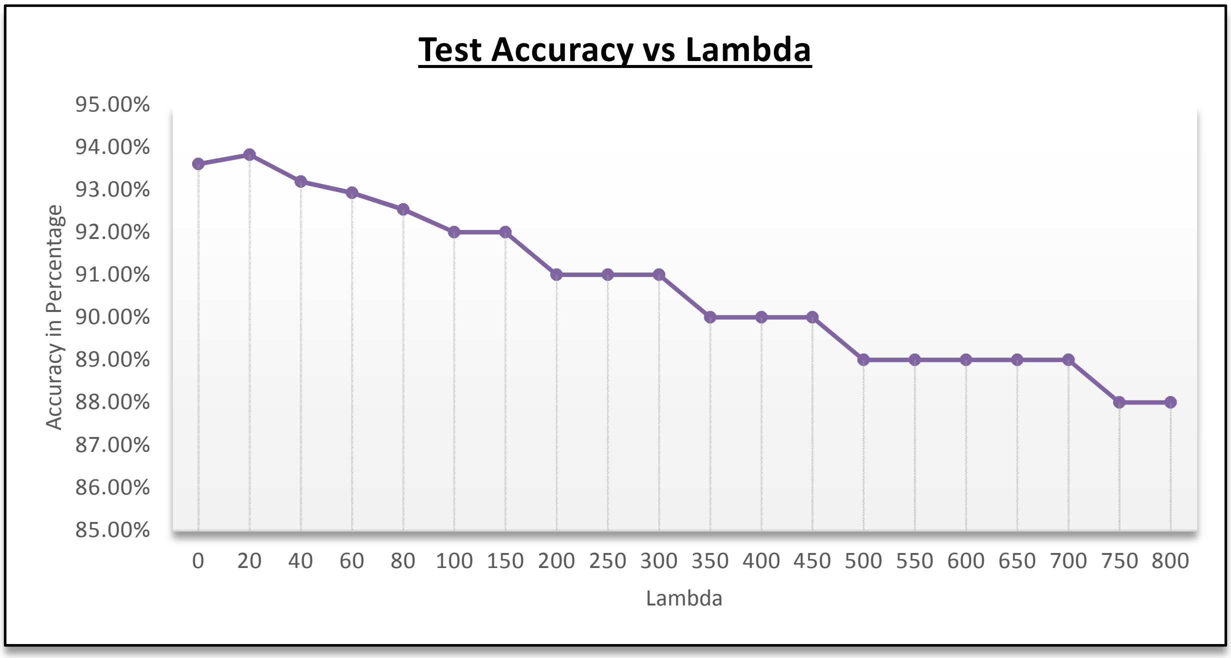

Figure 2: Plot of Test accuracy against Lambda

We change regularization parameter (lambda) values from 0 to 800 to find the best value of lambda; best value of lambda here refers to the value of lambda for which one gets the highest prediction accuracy w.r.t to the validation dataset. In this experiment we start with small values of lambda and subsequently increase the values of lambda and check the prediction accuracy on validation and test datasets. A lambda value that is low shall minimize the significance of the weights learned in the previous iteration of ANN and thereby result in higher significance given to the error value of the current iteration and a large value of lambda shall accomplish the exact opposite.

It is quite evident from the graphs above that the prediction accuracies suffer with increasing values of lambda and this is so due to the under-fitting that takes place because higher lambda values tend to give more importance to the weights at the expense of the negative log likelihood error function.

Experiment 2: Find the optimal number of hidden units

Hidden Nodes Analysis for best Lambda = 20					
Hidden Nodes	% Train Accuracy	% Valid Accuracy	% Test Accuracy	Time seconds	Time in minutes
31	93.53%	93.88%	93.49%	1358.24074	22.63734567
32	93.30%	93.83%	93.29%	2073.759992	34.56266653
33	93.12%	93.66%	93.28%	2156.97862	35.94964367
34	94.22%	94.38%	93.94%	1291.142665	21.51904442
35	93.63%	94.08%	93.59%	1580.839388	26.34732313
36	93.36%	93.78%	93.40%	1886.606277	31.44343795
37	92.98%	93.43%	93.29%	1492.073535	24.86789225
38	93.45%	93.91%	93.63%	1415.519804	23.59199673
39	92.57%	93.21%	92.76%	6316.5535	105.2758917
40	93.17%	93.60%	93.23%	1790.180831	29.83634718
44	92.95%	93.60%	93.33%	1786.824092	29.78040153
46	93.66%	94.13%	93.67%	1741.98089	29.03301483
48	94.14%	94.67%	94.23%	2526.269518	42.10449197
50	93.15%	93.74%	93.18%	3585.191874	59.7531979
52	93.06%	93.78%	93.37%	2485.267583	41.42112638
54	94.43%	94.71%	94.54%	2495.555803	41.59259672
56	93.04%	93.62%	93.32%	3171.10547	52.85175783
58	93.66%	94.15%	93.69%	1751.545875	29.19243125
60	92.37%	92.94%	92.92%	1775.818657	29.59697762
62	94.09%	94.63%	94.37%	1793.620797	29.89367995

Table 2: Variation of Accuracy with Number of Hidden Units

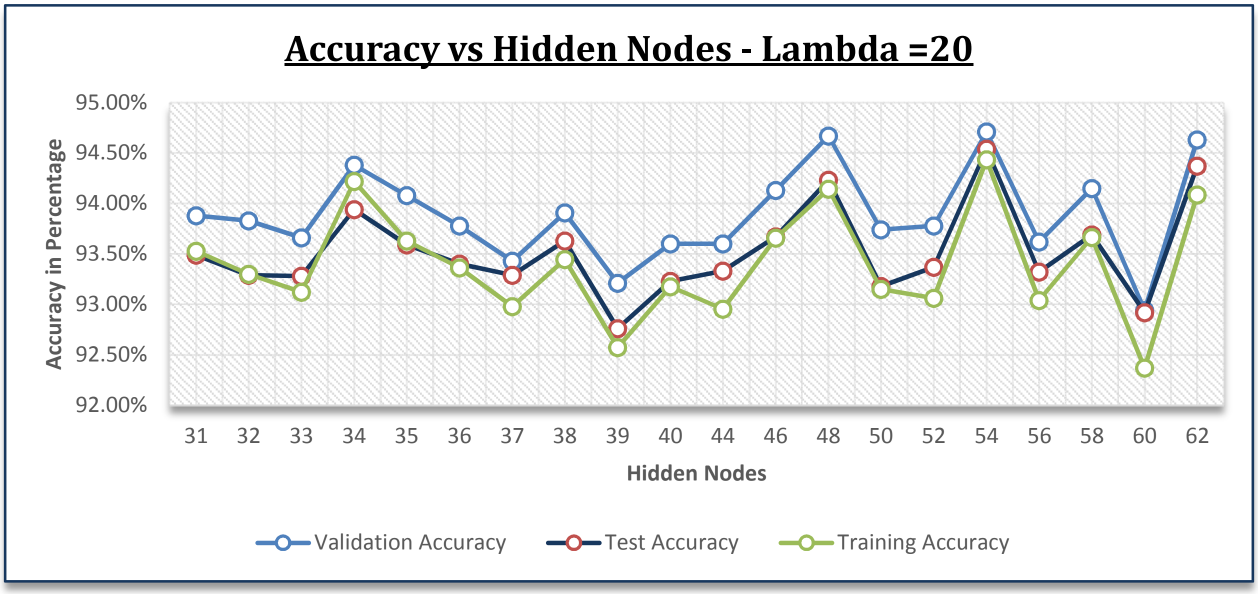

Figure 3: Plot of Accuracy against Number of Hidden Units

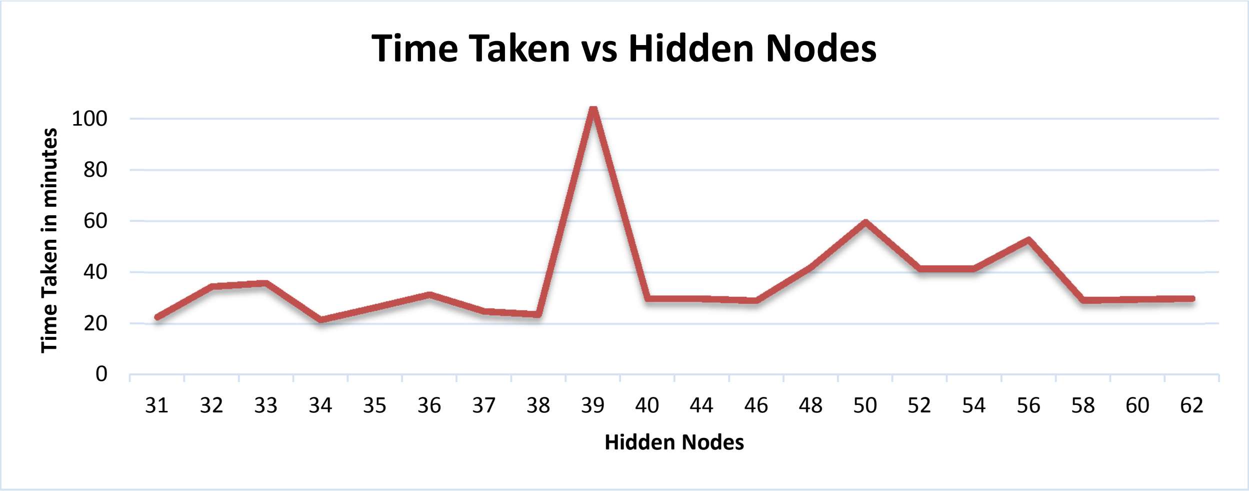

Figure 4: Plot of Learning Time against Number of Hidden Units

It is quite evident from the graphs above that the time taken by ANN to learn the right weights increases with the increase in the number of nodes in the hidden layer.

Observation: Given a node count, whenever the training time is large (do look at the outlier in above graph – node 39) we see that the corresponding prediction accuracy is very low. This can be attributed to non-convergence of conjugate gradient descent algorithm.

The running time increases significantly as we increase the hidden nodes from 40 to 52 and gradually reduces from 54 to 58. It can be explained by the increase in the amount of time required for matrix multiplication from input nodes to hidden nodes to output nodes.

Experiment 3a: Find the optimal value of k for K-NN (NO TIE BREAKING)

KNN - ACCURACY & TIME TAKEN (NO TIE BREAKING)			
K	**Validation Accuracy**	**Time Taken**	**Scaled**
1	97.61%	770.516966	1
2	97.18%	768.349507	0.99718701
3	97.68%	768.724842	0.99767413
4	97.69%	768.591207	0.99750069
5	97.63%	768.831794	0.99781293
6	97.53%	768.916971	0.99792348
7	97.50%	768.80312	0.99777572
8	97.41%	768.244467	0.99705068
9	97.36%	768.444036	0.99730969
10	97.36%	768.545268	0.99744107

Table 3: Variation of Validation Accuracy and Prediction Time with K

From the table above we can notice that the prediction accuracy are relatively low for K values 1 and 2 and then peak and post a certain value of K start decreasing. This is primarily due to over-fitting when the value of K is small and under-fitting when the value of K is very large. We therefore search for the point of inflection which shall be for the optimal value k; which is K=4 as per the data that we have gathered. We also find that the time taken is significantly lesser than the time by ANN.

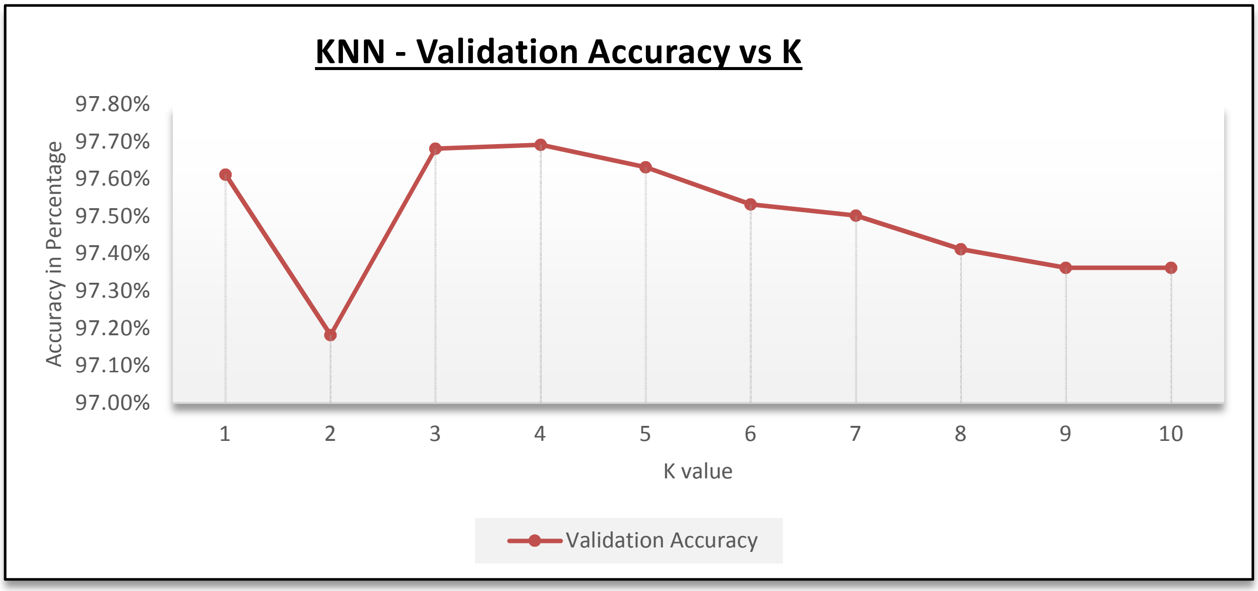

Figure 5: Plot of Validation Accuracy against K

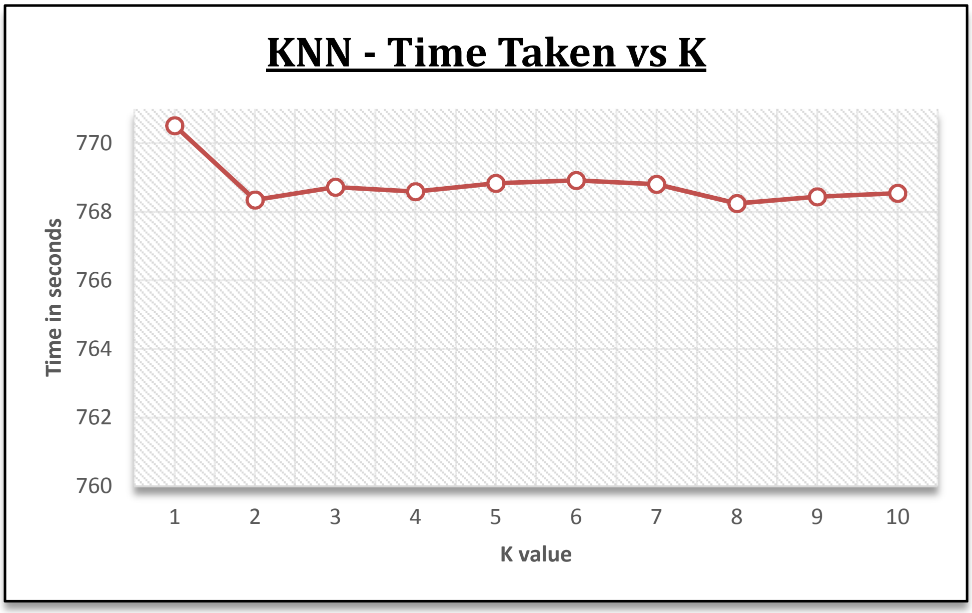

Figure 6: Plot of Actual Time Taken against K

Experiment 3b: Find the optimal value of k for K-NN (WITH TIE-BREAKING)

KNN - ACCURACY & TIME TAKEN

K	Validation Accuracy	Test Accuracy	Time Taken	Scaled
1	97.46%	97.09%	871.37741	0.948459
2	97.46%	97.09%	878.151074	0.955832
3	97.56%	97.30%	918.729368	1
4	97.67%	97.37%	917.785007	0.998972
5	97.68%	97.27%	917.626729	0.9988
6	97.71%	97.20%	917.597596	0.998768
7	97.50%	97.15%	917.832789	0.999024
8	97.65%	97.24%	917.600584	0.998771
9	97.51%	97.02%	917.789408	0.998977
10	97.49%	97.13%	917.775534	0.998962

Table 4: Variation of Validation and Test Accuracy, and Prediction Time against K

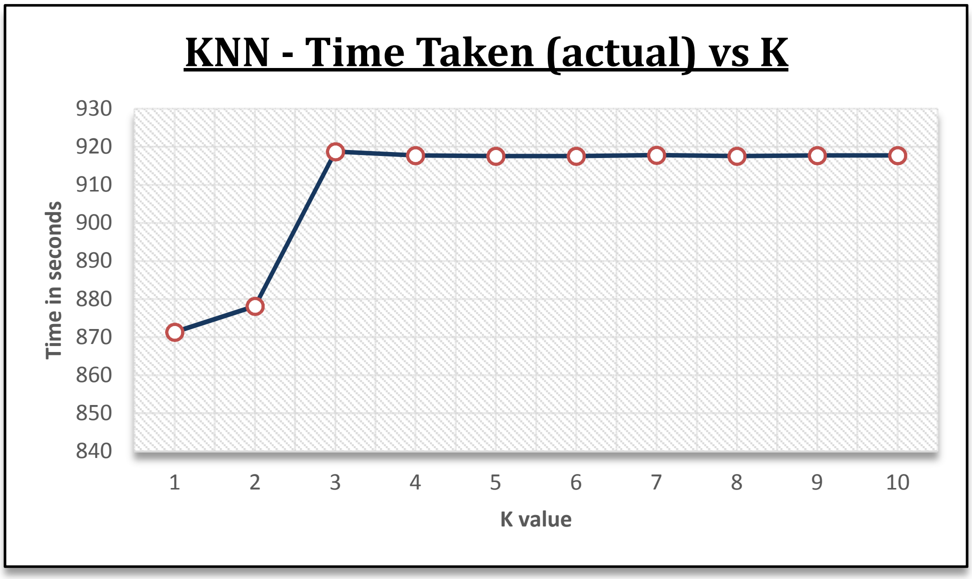

Figure 7: Plot of Actual Prediction Time vs K

From the graph and table above we can see that the time taken for each value of K is greater than the time taken when using KNN without tie-breaking rule, and this is so because of the added logic used for tie-breaking.

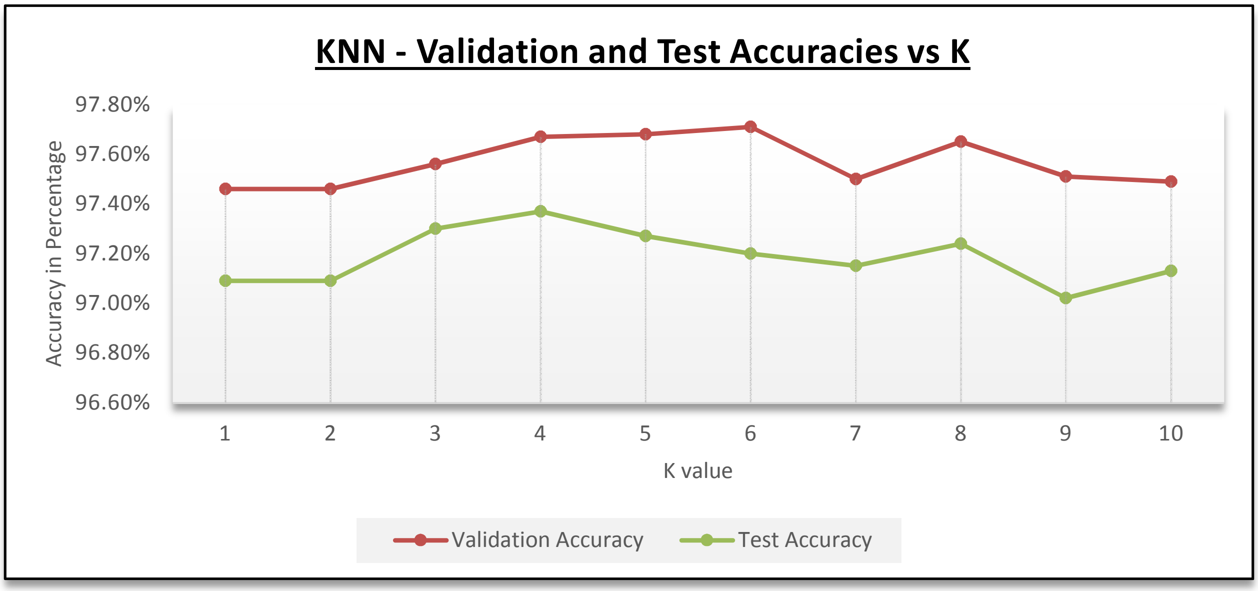

Figure 8: Plot of Validation and Test Accuracies vs K

In the above chart the prediction accuracy on the validation dataset is highest for value of k = 6. However; we find that the prediction accuracy on the test dataset is highest at K=4 instead. This could probably be because of higher K value resulting in an under-fitting (overlapping decision boundaries) leading to misclassifications and lower prediction accuracies.

KNN vs. ANN: Comparison of the classification techniques

CLASSIFICATION TECHNIQUES			
Metric		**Artificial Neural Networks**	**K - Nearest Neighbours**
Training Time & Accuracy	*Lot of Training Examples*	Takes a long time to learn the weights for each feature at each layer of the neural network	Does take time but lesser than ANN does; to learn the value of k that shall help KNN to rightly classify examples
	Not many Training Examples	Takes longer than KNN to learn as it needs to learn the weights for each feature; but does take lesser time than when it has to learn from a dataset that has lot more examples. Cannot guarantee good prediction accuracy due to lack of examples.	Does not take as much time as ANN to get the value of K; takes lesser time than when it has to learn from a dataset that has lot more examples. Cannot guarantee good prediction accuracy due to lack of examples.
	Small number of features	Learning shall happen faster but suffers from lack of features	Suffers from lack of features
	Large number of features	Does not suffer from curse of dimensionality. If there are many features and amongst them are features that do not necessarily contribute in helping rightly classify an example then those features are turned off, by adjusting the weights.	Suffers from curse of dimensionality; does not have a means of identifying the features that do not have any significance and therefore ends up treating all features as equals which need not necessarily be the right thing to do
Prediction Time		Irrespective of the dataset size the prediction time shall be lesser than KNN	Time taken to predict shall be longer than ANN irrespective of dataset size
Complexity		**Fairly Complex:** ANN learns a model whose parameters are the Weights, at each layer of the neural network, for each feature and the hyper-parameters namely; lambda and hidden nodes. The complexity makes ANN more trustworthy in terms of the predictions that it makes on testing data	**Low Complexity:** KNN does not learn much from the examples in the training set, other than finding the optimal value of K for which the classification accuracy is highest. In that way one can look at it as one parameter model with no other parameters to act as checks and balances. This could result in low prediction accuracy w.r.t testing data
Sophistication of Classification method		ANN method is fairly sophisticated as it does learning from classification mistakes and penalizes certain features by adjusting the weights. Therefore ANN does actual learning and therefore is a much more robust classifier than KNN	Naïve classification method; does not learn from classification mistakes therefore lacks sophistication

Ideal For		Using ANN would be ideal in scenarios where a learnt model needs to be constantly applied to thousands of test instances, for example, in a bot.	**Lazy classification:** where the classification method does not try to learn the underlying nature of the data and the nature and structure of the features that were used in finding optimal 'K'; which is used in classifying an example as belonging to one class or otherwise

PREDICTION RESULTS & PARAMETER CHOICE

Our decision to randomly assign examples to training and validation datasets from the master dataset for each experiment did seem to have an impact on the prediction accuracies, but this impact was not a significant one (as is made evident from our accuracy graphs).

Following are the final result of our experiment:

- **Artificial Neural Networks:**
  - Regularization Parameters (Lambda) – 20 **(Experiment 1)**
  - # Hidden nodes – 34 **(Experiment 2)**

We find that the predictions accuracies on validation datasets are the highest for hidden node values of:

- 34 : 94.38 % (Time taken: 21.51 minutes)
- 48 : 94.67 % (Time taken: 42.10 minutes)
- 54 : 94.71 % (Time taken: 45.92 minutes)

But since the miniscule increase in prediction accuracies do not justify the increase in training time; we choose to go with a value of 34 for Hidden nodes.

The prediction accuracy on the test dataset for Lambda = 20 and # Hidden nodes = 34 is:

93.94%

- **K-Nearest Neighbors:**
  - One can observe from the results of experiment 3 that the optimal value of K could be either 4 or 6 as they both have reasonably close prediction accuracies over the validation datasets. We initially went with the value of 6 for K in the spirit of choosing the parameter which gives the best prediction accuracies over the validation dataset; however we see observe an dues to under-fitting the value of 6 underperforms when it comes to prediction accuracies w.r.t to the test dataset. This logic led us to choose a value of K=4. The corresponding prediction accuracy on the test data is:

97.37 %

KNN vs. ANN (Advantages and Disadvantages)

KNN

- Prediction time for KNN is longer
- Training time for KNN is lesser than that of ANN
- Susceptible to curse of dimensionality
- Gives better prediction accuracies
- May suffer from noise in dataset as is the case from real world scenario; the handwritten digit dataset that we worked with is devoid of these issues hence the higher accuracy relative to ANN

ANN

- Prediction time is less
- Training time is longer for ANN
- Does not suffer from curse of dimensionality
- Gives reasonably good prediction results but not as good as KNN
- Will perform better than KNN when the training dataset has noise (as is the case in real world scenario) as ANN has the tools to deal with noise in the dataset

REFERENCE

[1]Determining Optimum Structure for Artificial Neural Networks:
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.32.9720&rep=rep1&type=pdf