ITCS 6156 Spring 2017
ASSIGNMENT : 3

SUPERVISED LEARNING

K-NEAREST NEIGHBOUR CLASSIFICATION

Rahul Rachapalli
800968032
rrachapa@uncc.edu

k-Nearest Neighbor Classification

(Supervised Learning)

k-Nearest Neighbors:

k-NN is a type of instance-based learning, or lazy learning algorithm. This algorithm stores all available cases and classifies the new cases based on a similarity measure. Distance functions are the common similarity measures used. This is based on the fact that the like things lie close to each other and can be grouped easily.

The principle behind nearest neighbor methods is to find a predefined number of training samples closest in distance to the new point, and predict the label from these. The number of samples can be a user-defined constant or the prediction of a new data entry is done based on the majority vote of its neighbors i.e., a test set is classified with a class that is most common among its neighbors. Neighbors-based methods are known as non-generalizing machine learning methods, since they simply “remember” all of its training data in a transformed format.

In k-NN regression, the prediction of an entry is done by calculating the average of the k-nearest neighbors.

The classification and regression algorithms can be assigned with weights to emphasize the importance of the nearest neighbors.

Lazy Learning Algorithm:

The algorithms in which the learning method for generalization using the training data set is delayed until a query is made to the system ie., instead of forming a generalized function, the query is compared with the training set and matched to the closest instance in the training set. The training time is almost negligible as all the training data is stored in the memory where as the evaluation of a query takes a very long time.

The main advantage of this type of learning is that the generalization is made locally for a query that is received. The disadvantage of this is that they require large amounts of space to store the entire training data set

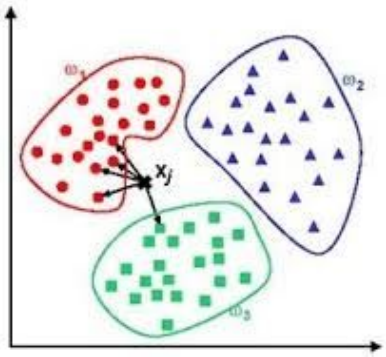

Fig 1: Classifying a new entry based on training set

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*

k-Nearest Neighbors :

A tree is built to classify and store the data in an optimal format for searching when a query is made. For each query, the prediction is made based on the k nearest neighbors..

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 k-NN algorithm that stores

the given matrix after converting it using ball_tree algorithm. The result is the predicted rating of the given review which will be done based on its k nearest neighbors.

Results:

There are 500 columns and all of them are used as attributes to predict the results. The prediction probability for this dataset is **57.9955%** with **8** neighbors, **ball_tree** as the classifier and **50** leaf nodes.

There are 2 algorithms to classify the data and store it for k-NN.

1. k-D tree
2. Ball tree

Its very clear that k-D tree can handle data up to 30 dimensions very well and then the classification for higher dimensions is not done very well. So, I have opted ball tree as the default algorithm for data classification. However, this approach is not suited for data with a large set of dimensions. So, we prefer using k-DD along with a supervised learning algorithm to make a strong. This technique is called Boosting.

Leaf_size is an attribute that represents the number of instances at the leaf nodes. I have conducted experiments to determine the effect of leaf_size on the classifiers. It is observed that the prediction rate did not depend on the number of nodes at the leaf. However, the time taken to construct the tree increases as the number of instances at the leaf decreased.

Number of leaf nodes	Prediction Percentage
50	57.9955
30	57.9955
20	57.9955
10	57.9955
5	57.9955
1	57.9955

Table 1 : Dependency of Prediction rate on number of leaf nodes

The effect of the number of neighbors on various classifiers are as follows :

Number of Neighbors	Prediction Percentage
2	18.2413
3	18.2413
5	57.9955
8	57.9955

10	57.9955
20	57.9955
25 and above	57.9955

Table 2 : Dependency of number of neighbors on Prediction rate

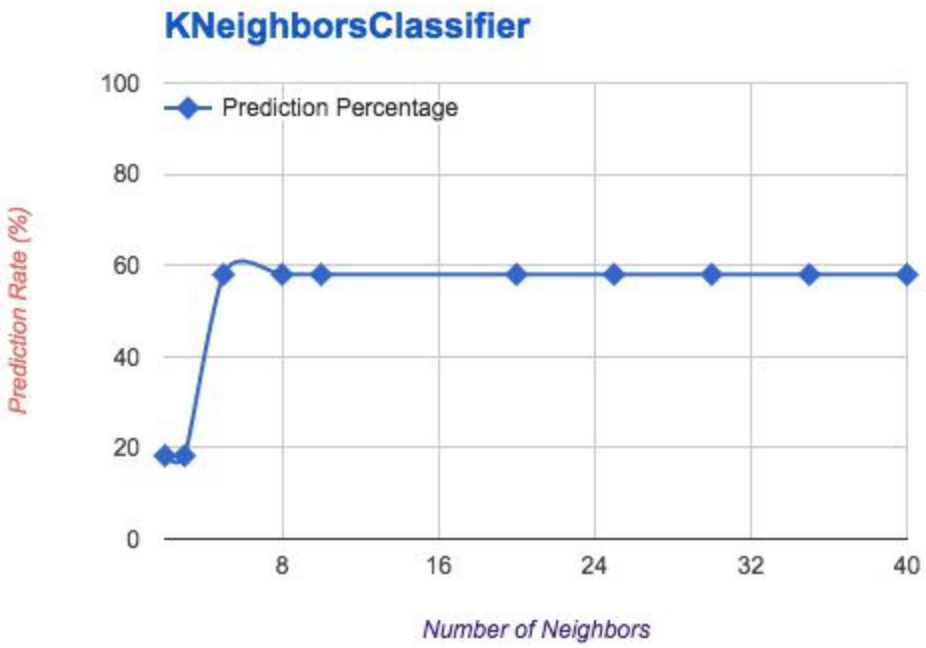

Fig 2 : Dependency of number of neighbors on Prediction rate

From the above experiments, I have observed that the optimal neighbors is **8** with 50 leaf_size.

It is also observed that the effect of weights

- uniform - assigning equal weights to all its neighbors
- Distance - emphasize importance of neighbors that are close by

Is negligible in our case.

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.

k-Nearest Neighbors :

A tree is built to classify and store the data in an optimal format for searching when a query is made. For each query, the prediction is made based on the k nearest neighbors.

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 **97.9967%** with **8** neighbors, **ball_tree** as the classifier and **50** leaf nodes that is used by k-Neighbor Classifier.

There are 2 algorithms to classify the data and store it for k-NN.

3. k-D tree
4. Ball tree

Its very clear that k-D tree can handle data up to 30 dimensions very well and then the classification for higher dimensions is not done very well. So, I have opted ball tree as the default algorithm for data classification. However, this approach is not suited for data with a large set of dimensions. So, we prefer using k-DD along with a supervised learning algorithm to make a strong. This technique is called Boosting.

Leaf_size is an attribute that represents the number of instances at the leaf nodes. I have conducted experiments to determine the effect of leaf_size on the classifiers. It is observed that the prediction rate did not depend on the number of nodes at the leaf. However, the time taken to construct the tree increases as the number of instances at the leaf decreased.

Number of leaf nodes	Classifier	Prediction Percentage
50	KNeighborsClassifier	97.9967

	KNeighborsRegressor	95.0036
30	KNeighborsClassifier	97.9967
	KNeighborsRegressor	95.0036
20	KNeighborsClassifier	97.9967
	KNeighborsRegressor	95.0036
10	KNeighborsClassifier	97.9967
	KNeighborsRegressor	95.0036
5	KNeighborsClassifier	97.9967
	KNeighborsRegressor	95.0036
1	KNeighborsClassifier	97.9967
	KNeighborsRegressor	95.0036

Table 3 : Dependency of number of leaf nodes on Prediction rate

The effect of the number of neighbors on various classifiers are as follows :

Number of Neighbors	Classifier	Prediction Percentage
2	KNeighborsClassifier	97.9967
	KNeighborsRegressor	95.39
3	KNeighborsClassifier	97.8854
	KNeighborsRegressor	95.3493
5	KNeighborsClassifier	97.8854
	KNeighborsRegressor	95.3368
8	KNeighborsClassifier	97.9967
	KNeighborsRegressor	95.1023
10	KNeighborsClassifier	97.914
	KNeighborsRegressor	95.2066
15	KNeighborsClassifier	97.4402
	KNeighborsRegressor	94.4007

20	KNeighborsClassifier	97.2167
	KNeighborsRegressor	93.9847
25	KNeighborsClassifier	97.2167
	KNeighborsRegressor	93.5526
35	KNeighborsClassifier	96.7726
	KNeighborsRegressor	92.6237
50	KNeighborsClassifier	96.4942
	KNeighborsRegressor	91.5504

Table 4 : Dependency of number of neighbors on Prediction rate

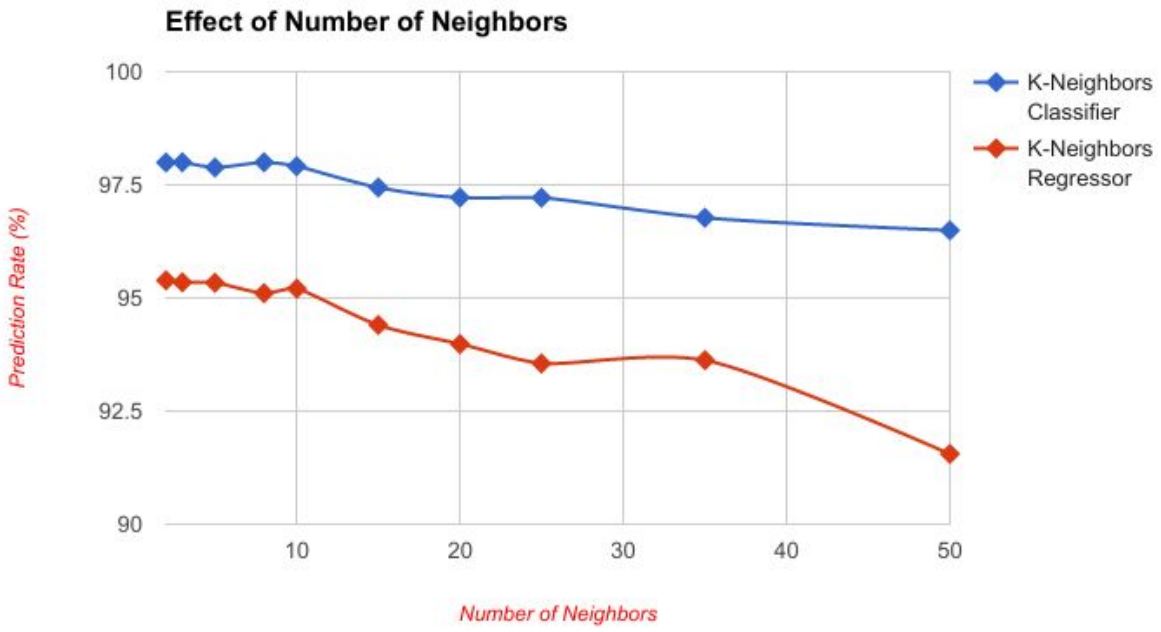

Fig 3 : Dependency of number of neighbors on Prediction rate

From the above experiments, I have observed that the optimal neighbors is **8** with 50 leaf_size.

It is also observed that the effect of weights

- uniform - assigning equal weights to all its neighbors
- Distance - emphasize importance of neighbors that are close by

Is negligible in our case.

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/K-nearest_neighbors_algorithm
7. http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html