UNSW SYDNEY

COMP 9417

Machine Learning and Data Mining

Implementation of Nearest Neighbour

Name: Junchi Wang z3493391

Kai Xiang Yong z5175681

Date: 11/08/2019

Table of Contents

1.0 Introduction

The k nearest neighbour algorithm, also known as the KNN method, is regarded as one of the simplest algorithms for classification and regression. This algorithm could be dated back to the 1960s when scientists named it to be the "minimum distance classifier" [1]. Even though there are serval alternative versions of this method nowadays, the fundamental logic behind this algorithm is intuitive and can be summarised as that samples in the same class must share similar properties. Hence, for a new instance to be classified, the algorithm would look into the existing train sets and compute the distance between this instance and each element in the train set. The top "k" samples will be selected as the focus group. The new instance is then determined to be the majority class within these k samples. In terms of continues value prediction, KNN algorithm may output the average of the labels of the top k samples.

Compared to many other algorithms, there are serval advantage of KNN. Firstly, the training step in KNN methods is simply placing the samples in training set, which has a relatively high efficiency of O(n). Next, the non-parametric characteristic of KNN allows it to make multi-class predictions without extra assumptions. This is beneficial to the flexibility of KNN algorithm and the usage of KNN could be extended to a large amount of problem sets. Finally, the ways for distance computation could be varied depending on the purpose, which further improves the flexibility of KNN algorithm.

However, this method is not robust for most modern applications. The main reason is that the amount of computation increases with the increment in data size and feature number. Some distance evaluation strategies, such as the Earth Mover Distance (EMD), require even more time in determining the top k samples. Though some approaches have been developed to reduce the number of features used to describe the data, some other problems such as the optimal value of "k" and the high sensitivity to outliers also restrict the KNN algorithm to provide high quality prediction [2].

Another problem caused by the basic algorithm is that the voting process treats all the samples in the top "k" equally regardless of the distance. The closer distance samples have no advantage against further samples. A common way to overcome this issue is to apply distance-weighted nearest neighbour (WNN) strategy. By increasing the weighting for the closer samples, the nearest neighbour algorithm can provide more accurate prediction result.

2.0 Aim

This project is aiming to explore the implementation of KNN algorithm without the assistance of existing library in Python. The classification and numeric prediction are tested independently on different datasheets with a range of value of k. Both Manhattan and Euclidean distances are applied to the KNN classifier in order to make comparison. Furthermore, the algorithm is extended to the distance-weighted nearest neighbour along with a difference analysis between KNN and WNN. Leave-one-out cross validation method is used to estimate the error produced by each method. Finally, a new data sheet is generated based on a pre-defined rule. This data sheet is then fitted into the classifiers and the validation error is compared to the Bayes error, which is obtained from the Bayes classifier, to test the accuracy of the classifiers.

3.0 Implementation

This section describes the implementation of the algorithms. The data pre-processing procedure is first outlined followed by the KNN and WNN algorithm discussion. Finally, the data set generator using probabilistic two-class classification target function is introduced. The entire project is coded in Python3. Since the purpose of the study is to explore the implementation of nearest neighbour algorithm, the programme does not include any machine learning library (table 1).

Table 1: Library used in this project

Library Name
numpy
pandas
math
matplotlib

3.1 Data Description and Pre-processing

Since the project investigates the performance of KNN algorithm for both classification and numeric prediction, it is necessary to use appropriate data sets for outstanding analysis. The chosen data sets are all sourced from the UCI Machine Learning Repository. Before any implementation, it is necessary to remove rows containing missing data and columns containing non-numeric data. Normalisation is also important for datasheets to avoid domination of large value data in distance evaluation. However, the first data sheet called ionosphere does not have missing data and all values are element pulse number between 0 and 1. Hence the only pre-processing for this data set is to map the labels from “good” or “bad” to “0” and “1”.

The following table illustrates the information about each datasheet and the pre-processing step.

Table 2: Datasheets information

Type	Dataset Name	Used Features (All features)	Used Size (All samples)	Labels Type	Pre-processing
Classification	Ionosphere	34 (34)	351 (351)	Binary	- Label string to number
Numeric Prediction	Autos	14 (25)	160 (201)	Continuous Integer	- Missing Data Removal - Normalisation

More details on the datasheets are shown in the data description files.

3.2 KNN Implementation

The KNN class utilised the Numpy function and constructed a KNN classifier from scratch. There are serval requirements of the classifier for an optimal design. Firstly, the classifier should be able to accomplish tasks not only for binary prediction but also for multi-label prediction. Secondly, the approach of distance computation should not be limited to only one type. It could be extended to different methods by simply choosing the way of computation when the user is initialising the classifier. Finally, the classifier should be able to make both classification and numeric prediction. This is also achieved by mode initialisation while defining the classifier.

The class takes 3 required arguments and 2 default arguments which are listed in the following table:

Table 3: Input of KNN classifier

Name	Description	type
Train	Train set of data	required
labels	Labels of the train set	required
k	The number nearest neighbour to consider	required
dis_mode	Mode of calculating distance: “e”: Euclidean Distance “m”: Manhattan Distance	Default: “e” Optional: “m”
pred_mode	Mode of prediction type: “vote”: classification tasks “avg”: numeric prediction tasks	Default: “vote” Optional: “avg”

The training step in KNN algorithm is not explicit. Some material stated that the aim of the train step is to place the samples into the training set. But in this project, no specific step for train was coded. In the prediction step, the classifier computes the distance and obtain the top k nearest points. The following pseudo code is implemented for the KNN classifier.

Input test cases

For *each element in the train set* **do**:

Compute the distance between the test point and train point

Store the distance as a list

Sort the distance list in ascending distance

Extract the top k elements from the sorted list

Obtain the elements with highest vote as the result *(*)*

For numeric prediction, the step with (*) is replaced by:

Obtain the average of the top k elements

3.3 WNN Implementation

The distance-weighted nearest neighbour algorithm is similar to the KNN algorithm but it weights the closer neighbour more heavily.

Input test cases

For *each element in the train set* **do***:*

Compute the distance between the test point and train point

Compute the weight function for each distance

Store the weighted distance in a list

Sort the distance list in ascending distance

Extract the top k elements from the sorted list

Obtain the elements with highest weighted vote as the result *(*)*

Similarly, the numeric prediction WNN step replaced the (*) step by:

Obtain the weighted average of the top k elements

It should be noticed that the weighted function in this project is:

$$w = \frac{1}{d(x_{test}, x_{train})^2} \tag{1}$$

Concerning the fact that some points may be overlapped and causes infinite value in weight function, an extra parameter called "distance threshold" is introduced to the classifier. When the distance between the 2 points is smaller than this threshold, the return weighted function becomes the square of threshold reciprocal which avoids infinite weighted distance.

3.4 Leave-one-out Cross Validation

The Leave-one-out Cross validation also uses Numpy library only. This algorithm leaves out each sample once and predicts it using the remaining training set. The pseudo code is shown below:

For *each element in train set* **do**:

Temporally Remove this element from the train set and the target set

Predict the result using classifier

Evaluate the error and place in a list

Compute the mean of the error list as error

3.5 Datasheet Generator

A new datasheet is generated from pre-defined rules in order to test the accuracy of the classifiers. In order to be consistent in dimension, the shape of the new data set is the same as the classification data set used in the previous section (i.e. 34×350).

The rules implemented in this project is defined in the following:

$$P(y = 1 | x_1 > 0.40) = 0.90$$

The generating procedure can be described as the following:

Step 1: Randomly generate 350 labels

Step 2: Randomly generate 34 features on each of 350 samples

Step 3: Based on the rule and the label, generate and replace features correspondingly

This datasheet is then tested the KNN and WNN classifier to find the error.

3.6 Bayes Error Computation

Bayes Error for binary classification can be calculated using the formula:

$$P(error) = P(class\ 1|x)\ P(x) \quad \text{If we predict class2}$$
$$+ P(class\ 2|x)\ P(x) \quad \text{If we predict class1} \qquad (2)$$

However, Bayes Error can also be calculated from the error estimated from the Bayes Classifier. In order to obtain a better precision on Bayes Error, we have implemented a Bayes Classifier model for error estimation. The result is then compared with the error rate of both KNN and WNN for further performance measurement. Leave-one-out cross validation is still implemented in this error measurement.

The procedures can be described as following:

Separate samples by its label into 2 distinct groups

For each label:

Calculate the mean $\bar{y}$ and standard deviation s

Calculate the probability of classification of each class using $\bar{y}$ and s

Obtain the overall probability of classifying each class

Determine the largest probability of labels and predict

Compare predictions and actual labels to calculate Bayes Error

Since we are using a large generated dataset, the output error rate should be precise and close to the probability calculated using equation (2).

4.0 Experimentation

4.1 Classifier Cross Validation

The first step of any experiment in KNN algorithm is to determine the optimal k value. Figure 1 shows the change in validation error with respect to the change in k. For KNN classification, k is limited to be odd number otherwise oscillation in error will occur.

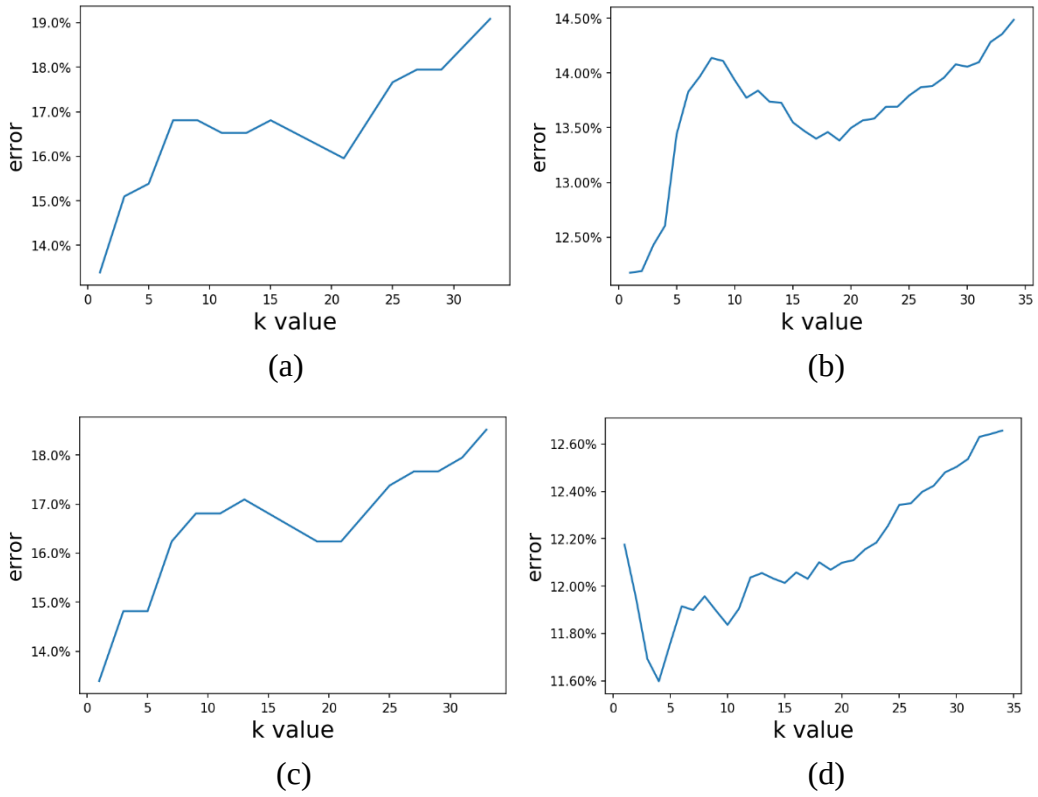

Figure 1: k value versus error plot for: (a) knn classification (b) knn numeric prediction (c) wnn classification (d) wnn numeric prediction

A general trend could be observed from these diagrams is that the error increases with the increment of k value. The optimal k is usually below 5 and the minimum error obtained from WNN classifier is lower (table 4).

Table 4: minimum error of classifiers

	Classification	**Numeric Prediction**
KNN	13.4%	12.1%
WNN	13.4%	11.5%

However, the low difference in error indicates that weighted-distance strategy does not have much influence on the result for this datasheet. Further investigation should be made on more data sheet to explore the accuracy difference between KNN and WNN.

4.2 Data Generation and Bayes Error

The hypothesis for this part was that the error rates of KNN and WNN is greater than the Bayes Error. This is because the new datasheet has very limited informative features and KNN does not perform well on such dataset. On the other hand, Bayes Classifier is a Probabilistic Classifier hence it would treat uninformative features equally and develops strong assumptions of independence among features [3]. KNN and WNN were ran with different k-values and distance mode to determine the optimal strategy (figure 2).

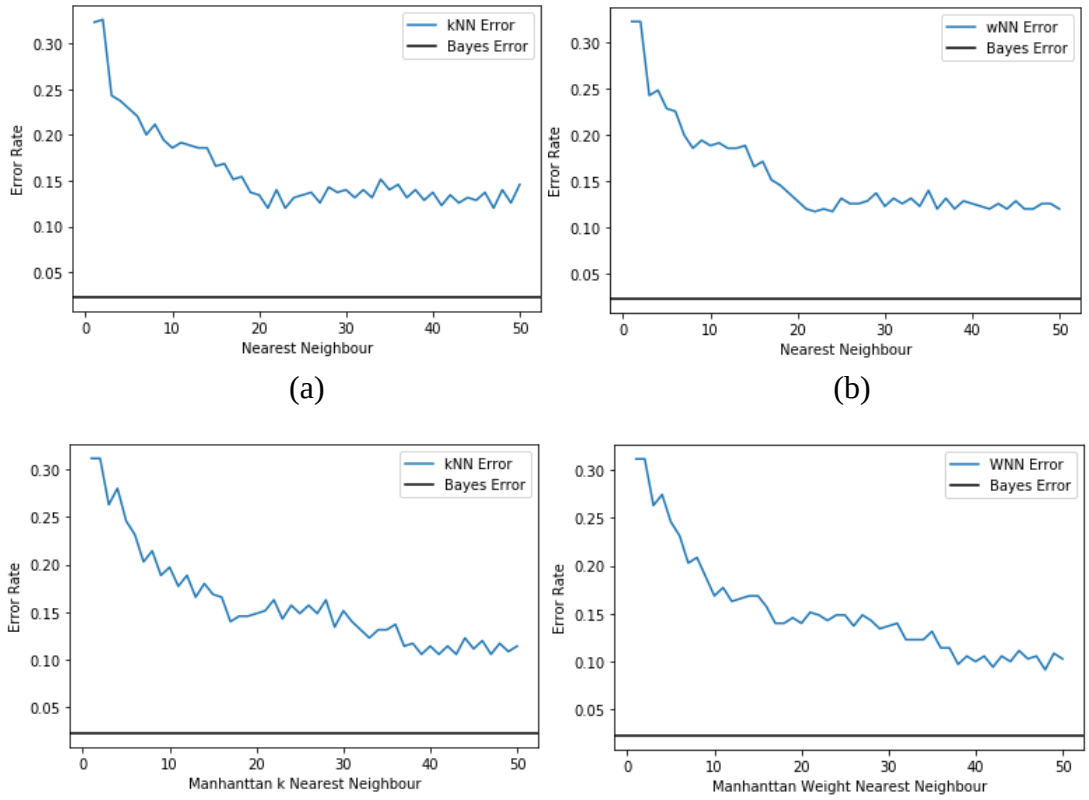

Figure 2: Comparisons between Bayes Error and error rate with (a) Euclidean distance KNN (b) Euclidean distance WNN (c) Manhattan distance KNN (d) Manhattan distance WNN

Table 5: Comparisons between Bayes Error and minimum error rate of KNN and WNN under different distance measures

	Euclidean Distance		**Manhattan Distance**		**Bayes Error**
	k value	**Error Rate**	**k value**	**Error Rate**	
KNN	21	0.12	39	0.1057	0.0257
WNN	22	0.1171	48	0.0914	

As shown in table 5, Bayes classifier outperforms KNN and WNN classifer significantly. This is because KNN and WNN algorithm make an assumption that similar points share similar labels. Nevertheless, only one feature among 34 features in the dataset shows informative characteristic and hence KNN and WNN could not provide high performance.

In addition, the curse of dimensionality indicates that KNN and WNN degrades with higher dimensional data because there is little difference between the nearest and farthest neighbours. But considering the large number of features in this dataset, the NN and WNN classifiers perform better than the initial expectation. As the k-value get larger, the error converges to approximately 0.13 which is close to the target function we set for this dataset.

5.0 Conclusion

In conclusion, this project has explored the implementation of KNN and the distance weighted KNN algorithm. Error estimation is based on the leave-one-out cross validation method and it was repeated for different k value in order to obtain the optimal classifier. It was found that the error usually increases when k is increasing. Even though the error difference between KNN and WNN classifiers is small, it could be seen that the distance-weighted strategy has a positive impact on the classification.

Furthermore, a new set of data with the same shape as the "ionosphere" data sheet was generated according to a predefined rule. The Bayes error of this data set was calculated to be 0.0257 which is much higher than the errors obtained from the KNN and WNN classifier. This result is simply because Bayes error is calculated based on the rule of data generation. Besides, nearest distance algorithm assumes that all features are informative, which is not true in this data set. Finally, the problem of curse of dimensionality of nearest neighbour classifier also reduces the accuracy in this dataset. This project can be extended further by testing the KNN and WNN efficiency on data with more samples and more complex characteristic.

6.0 Bibliography

[1] N. Nilsson "Minimum-distance classifiers", In *Learning Machines: Foundations of Trainable Pattern-Classifying Systems*, 1st ed., California, US, McGraw-Hill, 1965, pp. 16.

[2] P. Cunningham and S. Delany, "k-Nearest neighbour classifiers", 2007.

[3] N. Friedman, D. Geiger, and M. Goldszmidt, "Bayesian Network Classifiers", In *Machine Learning*, vol.29, no. 2 – 3, 1997, pp. 131 – 163.

[4] M. Verleysen and D. François, "The Curse of Dimensionality in Data Mining and Time Series Prediction". In *Computational Intelligence and Bioinspired Systems*, vol 3512, Springer, Berlin, Heidelberg, 2005.