Implement KNN Classifier for Sequence Prediction

Dr.Muhammad Haris
Information Technology (M.Eng.)
Computer Science and Engineering
Frankfurt University of Applied Sciences
Frankfurt am Main, Germany
Email: here

Zaka Ahmed
Information Technology (M.Eng.)
Computer Science and Engineering
Frankfurt University of Applied Sciences
Frankfurt am Main, Germany
zaka.ahmed@stud.fra-uas.de

***Abstract*—The abstract goes here.**

I. Introduction

A. Background

KNN was first developed by Joseph Hodges and Evelyn Fix in the year 1951[1], in statistics the concept of k-nearest neighbors algorithm(k-NN) is involved in the non-parametric supervised learning method. Further developments in KNN are proceeded by Thomas Cover.[2] KNN is commonly used for regression and classification. Both in regression and classification the input consists of k-closest training examples in the data set. Remember the output depends on whether the use case is either regression or classification of K-NN.

B. Regression

The main difference between classification and regression is that in regression, the output is the property value for the object. The value is the total average of the neighbor's nearest values. If k=1, the output is assigned from that particular single nearest neighbor.

C. Classification

The main variation in the output of classifier and regression is that in classification, the output is the class membership. In classification, the object is classified based on the votes of its nearest neighbors. If k = 1, then the object will be in the class of that single nearest neighbor. The simple function of the kNN model is to predict the target class label. In other words, the class label is often described as a majority voting. The most common terms are technically considered "plurality voting" and "majority vote" The term "majority voting" means the majority needs to be greater the 50% for making decisions. The classification problems with only two classes, like binary predictions, there is always a majority. A majority vote is also automatically a plurality vote. We don't require multi-class settings to make predictions via kNN in multi-class settings.

1) Subsubsection Heading Here: Subsubsection text here.

II. Compute KNN: distance metrics

If we try to recap, the main objective of the k-nearest neighbor algorithm is the identification of the nearest neighbors of a given input point. So, What we can do is assign a class label to that specific point. The first thing is determining the distance matrics. In order to find which class(data point) is nearest to the input data, to do so we will calculate the distance between the data points and query point. We get assistance to decide in which regions the input point belongs. Voronai diagrams are used for the visualization of these decision boundaries.

A. Euclidean distance (p=2)

Euclidean is limited to real-valued vectors and is most commonly used for distance measures. The below expression is used to determine the straight line between the input point and the other point being measured.

$$Euclideandistance = d(x,y) = \sqrt{\sum_{i=1}^{n}(y_i - x_i)^2}$$

B. Manhattan distance (p=1)

In Manhattan distance, the absolute value is measured between two points. Manhattan distance is also one of the popular distance metrics. The most common example is Uber app visualization with a grid and navigation via streets from one address to another illustration.

$$Manhattandistance = d(x,y) = \sqrt{(\sum_{i=1}^{m}|X_i - Y_i)|)}$$

C. Minkowski distance

It is the generalized form of Manhattan and Euclidean distance matrics. Manhattan distance is denoted by p equal to one whereas the Euclidean distance is represented by p equal to two. Parameter p allows the creation of other distance metrics as shown below.

$$Minkowskidistance = \sqrt{(\sum_{i=1}^{n}|X_i - Y_i)|)^1/p}$$

D. Hamming distance

This technique used with string vectors or boolean identifying the points where the vectors do not match. Overlap metrics are also referred as represented below:

$$HammingDistance = D_H = (\sum_{i=1}^{k}|X_i - Y_i)|)$$

III. Implementation Details

Description of the KNNClassifier class. Explanation of methods and functions within the KNNClassifier. Discussion of key functionalities and their significance.

A. Training and Test Dataset

B. KNN Classifier Class

These sections try to discuss the methods defined within the KNN classifier Class, these methods are discussed below:

1) Distance(double[] vector1, double[] vector2): As we have discussed in section 2(A) regarding the Euclidean distance. The distance function calculates the Euclidean distance between two vectors which in our case are vector1 and vector2. As, the formula of Euclidean distance is to go through each element of a vector and compute the squared difference and then find the accumulative sum. And then finally finding the square root of the whole sum, represents the Euclidean distance between two vectors.

2) Vote(IndexAndDistance[] info, double[][] trainData, int numClasses, int k): This method performs the voting mechanism between the k nearest neighbors. The arguments of this method are IndexAndDistance which represents distance and Indices of nearest neighbors in other we can say the information of nearest neighbors, then our training dataset (trainData), the total number of classes (numClasses), and the value of K. The step of Vote method is to initialize an array whose responsibility is to store the total number of votes against each class means every nearest k-neighbour has a vote. In the starting the array is initialized as zero. Then in the second step, it goes through the first k neighbors and retrieves its class label from the training data, and the respective array index is counted as one positive vote. In our case, the training data have either an odd number or an even number respective voting has been counted for each of them. Then the counted voting with the highest vote returns its label as the predicted class.

3) Classifier(double[] unknownSDR, double[][] Sdrdata, int numofclass, int k): This method performs the KNN classification for a given unknown SDR which we are also saying as test-data. The arguments of this method include the testdata (unknownSDR) and the training data (Sdrdata) and the total number of classes which is 3 in our case. And the value of k as an input parameter. In this method, we have called all the methods including IndexandDistance objects to store indices and distances of all data points from the unknown SDR. The next, we have called the distance method, which will calculate the distances between the test data ad training data points that are stored in the array. Then we sorted an array in ascending order based on distances. Then in the final, we have called the vote method to determine the predicted class label based on k nearest neighbors. Then, the class with the highest vote is where the test data belongs to that specific class. And then this method returns the result of a specific class label.

4) IndexAndDistance Class: This class used to represent the index of a training item and its distance from the input (test data). It implements the IComparable¡IndexAndDistance¿ Interface to enable sorting based on distance. The CompareTo method is overridden to compare distances between instances of IndexAndDistance class.

C. Defining k

The k value indicates how many neighbors will be compared or we can say checked to determine the resultant class in KNN Algorithm. For-example: By changing the value of k the classification can lead to underfitting or overfitting. If k=1, the instance will be assigned to the same class because we have a single neighbor. By using lower value of k can low bias, but high variance. As well as, larger value of k may lead to lower variance and high bias. So, we can define k as the balancing act as different values impact the variance on underfitting or overfitting. For avoiding ties in classification, k is recommended to be a odd number. The best approach to get a optimal k for you dataset is cross-validation tactics.

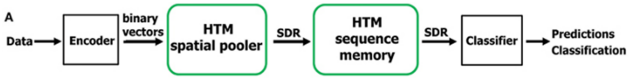
IV. SDR Dataset Handling

A. Overview of HTM

As the objective behind the HTM CLA is to make a progress towards making a program that functions cognitive tasks as like a simple human being brain. The prediction is done by making a system which can memorize as well as learn from the information executions which are fed before. HTM to anticipate and memorize, it requires user Input.

As the overall HTM have multiple sections which includes data, Encoder, HTM spatial Pooler, HTM temporal Memory, and HTM Classifier. The data or which is also know as input is a scalar value, data or time, or a picture. Then the next element is encoder which is responsible for changing the data into SDR which can further be used with HTM classifier. SDR is in the cluster of binary input either '0' or '1'. As discussed earlier, input of encoder can be anything a scalar value. It includes locations, weeks, months, time or days in a week etc.

The next part of the HTM is a spatial pooler it, is an algorithm which learns spatial patterns. The spatial pooler gets an input of bits cluster and converts it into SDR. Next Parts is Temporal memory, it a part which learns the arrangements of SDRs shaped by the spatial pooler algorithm. [5]



B. Spatial Pooler

Spatial Poolar is the second phase of HTM learning. It uses the output of the encoder to learn the SDR of the given input binary array. The idea for the spatial poolar SDR is to generate SDRs of the input which is the output of the encoder. Once the input SDR are learned, if the same input is given again, it tries to match already learned SDRs and then generates a similar matching SDR.In this method, it will disgunish, is it a same input which is already memorized or a different one.

Explanation of how SDR datasets are handled and utilized. Description of methods for loading SDR data from files. Discussion of data structures and formats used for SDR datasets.

V. Unit Testing and Exception Handling

Overview of unit testing methodologies used. Description of UnitTest1 class and its purpose. Explanation of how test datasets are generated and utilized for testing. Discussion of exception handling strategies implemented. Explanation of how errors and exceptions are handled during file reading and data parsing.

A. Applications of k-NN in machine learning

KNN Algorithm is utilized in different applications across different sectors, mostly in classification. The common cases includes:

1) Pattern Recognition: KNN is used for the identification of specific patterns, it can be in a text. Like it predict the missing fill in the blanks. This also help in solving cache which are basically handwritten numbers. So, KNN can also to identify these patterns.

2) Healthcare: The common use of KNN is health department is prediction of chances of cancer as well as heart attacks risks. The algorithm try to learn from most likely gene expressions.

3) Recommendation Engines: When we surf on internet, the KNN algorithm can be used by the website to recommend us other additional content. This recommendation is based on user behaviour. But for larger datasets this approach is not optimal.

4) Data preprocessing: Mostly we have a missing values in our dataset, KNN algorithm can help to determine those values. Those estimated missing values are also known as missing data imputation.

5) Finance: The banks uses the credit card spending to predict the risk associated with the specific individual loan. As well as knn is used to determine the credit card worthiness of a loan application. So, KNN is used in variety of economic and finance departments. As well as the other common use case is currency exchange forecasting, stock market as well as trading futures etc.

Acknowledgment

The authors would like to thank...

References

[1] Cover, T., & Hart, P. (1967). Nearest neighbor pattern classification. IEEE transactions on information theory, 13(1), 21-27.

[2] Cover, T., & Hart, P. (1967). Nearest neighbor pattern classification. IEEE transactions on information theory, 13(1), 21-27.