SOCIETY FOR SCIENCE AND EDUCATION
UNITED KINGDOM
TMLAI TRANSACTIONS ON MACHINE LEARNING AND ARTIFICIAL INTELLIGENCE
VOLUME 4, ISSUE 1
ISSN 2054-7390

The Adept K-Nearest Neighbour Algorithm - An optimization to the Conventional K-Nearest Neighbour Algorithm

Anjali Jivani[1], Karishma Shah[2], Shireen Koul[3] and Vidhi Naik[4]
Department of Computer Science and Engineering, The M. S. University of Baroda, Vadodara, India.
[1]anjali_jivani@yahoo.com; [2]12.karishma@gmail.com; [3]koulshireen@gmail.com;
[4]vidhinaik2@yahoo.com;

ABSTRACT

This research aims to study the efficiency of a well-known classification algorithm, K-Nearest Neighbour, and suggest a new classification method, an optimised version than one of the existing classification method. The purpose of this research is to reduce the time taken by the existing K- Nearest Neighbour Classification method. The classification algorithm's purpose is to identify the characteristics that indicate the class to which each document belongs. This pattern not only helps in understanding the existing data but also to predict how new instances will behave. Classification algorithms create classification models by examining already classified data (cases) and inductively finding a predictive pattern.

Keywords: Supervised Learning, classification, k Nearest Neighbour, Cosine Similarity

1 Introduction

Classification [1] is a data mining (machine learning) technique used to predict group membership for data instances. For example, you may wish to use classification to predict whether the weather on a particular day will be sunny, rainy or cloudy. Popular classification techniques include decision trees and neural networks.

Following are the examples of cases where the data analysis task is Classification –

- A bank loan officer wants to analyse the data in order to know which customers (loan applicant) are risky or which are safe.
- A marketing manager at a company needs to analyse a customer with a given profile, who will buy a new computer.

Classification is a supervised learning data mining techique.

2 k-Nearest Neighbour Algorithm (kNN)

Simply stated, kNN is an algorithm that classifies the new cases based on similarity measures[2] or distance measures of pair of observations such as euclidean, cosine, etc. kNN algorithm is a lazy learner i.e. it does not learn anything from the training tuples and simply uses the training data itself for classification [5]. It is a non-parametric method used for classification and regression. [3]

DOI: 10.14738/tmlai.41.1876
Publication Date: 27th February, 2016
URL: http://dx.doi.org/10.14738/tmlai.41.1876

The newly arrived document is classified on the basis of the majority occurring class of its neighbours, with the document being assigned to the class most frequent among its *k* nearest neighbours. Figure 1[4] depicts how kNN works.

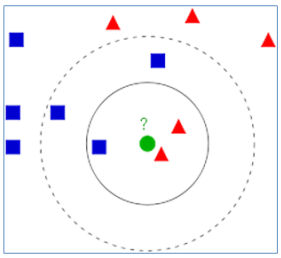

Figure 1: Example of kNN

The test document of green circle needs to be classified; it can either be classified into red triangle or Blue Square depending on the distance between each of them. If k=3 (shown by a solid line), green circle will be classified to red triangle class as out of the three, red triangle occurs two times and blue square occurs one time. But, if k=5(shown by a dotted line), then the green circle will be classified to blue square for similar reasons.

Line up of sub-procedures explains how kNN method is implemented on a broader aspect.

A newly arrived document is fed into an algorithm that removes all the stop words present in the text file and the intermediate result is then sent to porter stemming algorithm. The sub-procedure at that point calculates tf-idf score and hence cosine similarity between the newly arrived document and all the cases which were previously classified into specific classes. An ordered sorted list is prepared with cases arranged in the descending order fashion of cosine similarity. Cosine similarity ranges between 0 and 1 where cosine similarity of 0 means that the two compared cases are not at all similar and a cosine similarity of 1 means that the two compared cases are completely similar. 'k' in k Nearest Neighbour method is a user input. The procedure then selects the top k neighbours from the ordered list. The newly arrived document is then classified in to that class which occurs maximum number of time in the top k-selected neighbours.

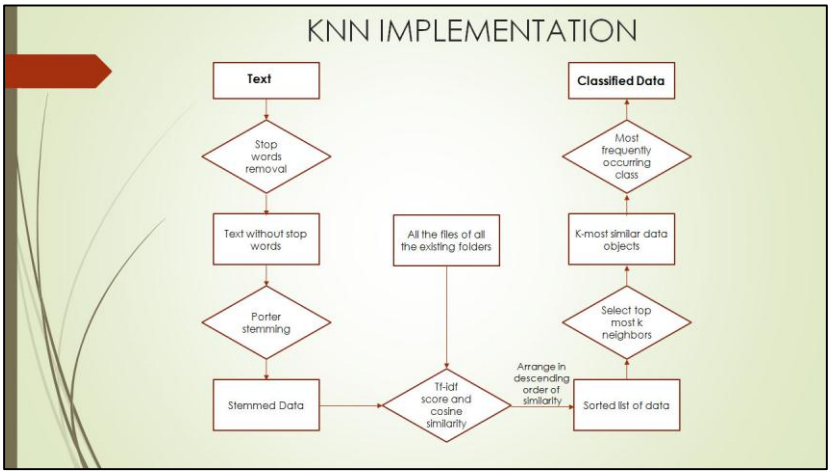

Figure 2: Flowchart showing stepwise implementation of kNN

2.1 Advantages of kNN

The advantages of the kNN method are as follows:

1. Analytically tractable
2. Simple implementation
3. Nearly optimal in the large sample limit ($N \rightarrow \infty$)
4. Uses local information, which can yield highly adaptive behaviour
5. Lends itself very easily to parallel implementations

2.2 Drawbacks of kNN

The drawbacks of the kNN method are as follows:

1. kNN algorithm is that it is a *lazy learner*, i.e. it simply uses the training data itself for classification.
2. Result of this is that the method does not learn anything from the training data, which can result in the algorithm not generalizing well. Further, changing K can change the resulting predicted class label.
3. Also, algorithm may not be robust to noisy data.
4. To predict the label of a new instance the kNN algorithm will find the K closest neighbours to the new instance from the training data, the predicted class label will then be set as the most common label among the K closest neighbouring points.
5. The algorithm needs to compute the distance and sort all the cases at each prediction, which can be slow if there are a large number of training examples.
6. Large storage requirements.
7. Computationally intensive recall.
8. Highly susceptible to the curse of dimensionality.

3 The Adept k Nearest Neighbour Algorithm

Here we describe the proposed algorithm. We have considered our input data as text documents. These documents belong to three classes ARTS, SCIENCE and FINANCE with a number of documents already been classified into one of these three classes. In the document of the conventional kNN as we have seen above, after the removal of stop words and obtaining the stemmed output, the intermediate result will be compared with the existing classes of documents, in order to find the cosine similarity between all of them.

The newly suggested method, Adept kNN, emphasises on the fact that most of the comparisons or similarity checks go in vain. It is suggested that in contrast of comparing a document with each existing or previously classified document, it will be better to compare the document to be classified with three cases only (or files as in our example) each document being an intermediate representation of corresponding class. This intermediate representative document or text file is subjected to change over time.

Adept kNN is just an optimisation of kNN. The flowchart of the Adept kNN is as shown in Figure 3.

The intermediate file is designed in such a way that it contains only the important or frequently occurring words/tags. To determine the frequently occurring words, a counter for each word is maintained which is

incremented by one each time the word occurs in the text. Once, the counter exceeds a value X (which is a user defined value), the counting procedure for that particular word stops as the word has already been put in the file or list of frequently occurring words or tags. A newly arrived document is then compared with only 3 such files/cases instead of being compared to 30 files as we were doing it previously. Also with arrival of each new document, the intermediate representative file is updated with concatenation of new frequently occurring tag, if any.

The line-up of sub-procedures in Adept kNN is similar to the one in conventional kNN.

Firstly, the document is fed into procedure for stop word removal and then into porter stemming method. Then the cosine similarity between the intermediate stemmed output and representative files is determined. The document to be classified is classified into the class which is most similar to it, that is the one which has highest cosine similarity. This can also be inferred that there is no question of k (user input) in this method.

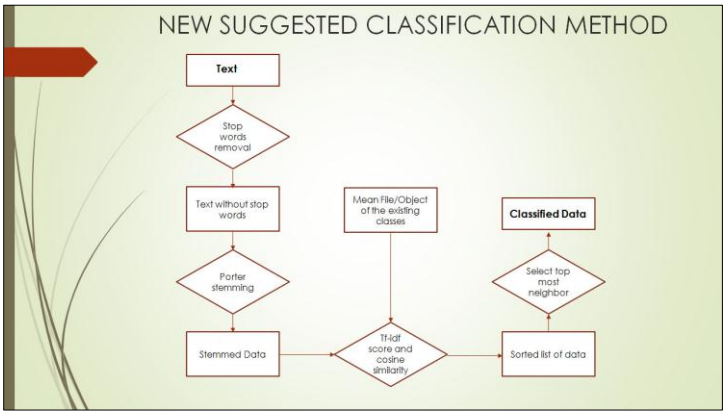

Figure 3: Flowchart showing stepwise implementation of Adept kNN

3.1 Advantages of Adept- kNN over kNN

1. The algorithm needs to compute the distance with only the representative files and hence reduce the time required to perform this task significantly.
2. Accuracy of this method is almost comparable to that of the original kNN.
3. Lesser number of comparisons or similarity checks are needed to be performed as compared to that of kNN.
4. There is no need of a user input K as the document is directly classified to that class whose representative file is most similar to the newly arrived document i.e. where cosine similarity is highest between the new document and the representative file.

3.2 Disadvantages of Adept kNN

1. There is an overhead of maintaining an additional file in addition to maintain the already residing files or cases.
2. When the count of previously classified files will increase to millions, the size of intermediate representative file will also become huge.
3. In addition to maintain the representative file, Adept kNN also needs to consistently maintain a counter of each word or tag. If at any moment it fails to do so, the algorithm may result in an

inconsistent classification as it no longer maintains the perfect list of frequently occurring words or tags.

The algorithm for our proposed Adept kNN method is as shown is Figure 4. The details about each procedure are described as per the pre-processing done on the documents and the classification process itself is also described in detail.

The documents are first tokenized and converted to a bag of words. After that the stop words have been removed by comparing the tokens from the documents with the list of stop words available on the net. The Porter's stemming algorithm has been used to perform stemming on the remaining words. The comparison between the bag of words of documents has been done using the cosine similarity.

Algorithm: (Implement the Adept K-NN) ADEPT_KNN(F_NAME)
F_NAME is the name of the file to be classified. START_TIME is the value of time at which algorithm starts. OUT_STEMFILE is the file which holds the stemmed output. COS_SIM is the value of cosine similarity between two files. END_TIME is the value of time at which algorithm ends. DURATION is time taken by algorithm to complete. HM is the HashMap<Double, String> which holds the cosine similarity and the corresponding file name. AL is the ArrayList<Double> which holds the cosine similarity.

1. SET START_TIME := CURRENT_TIME.
2. Call REMOVE_STOP_WORDS (F_NAME).
3. Call PORTER_STEMMING ().
   Read OUT_STEMFILE and write it into TEMP_FILE1.
4. Call GET_FILES1 () OR ST:=GET_FILES().
5. Call FIND_COS_SIM(TEMP_FILE1).
6. /*Select first value of file name and corresponding folder from the descending ordered list of cosine similarity*/
   6.1 COS_SIM := AL.GET(0).
   6.2 PATH:=HM.GET(COS_SIM).
7. /*Steps 7 and 8 check which folder comes maximum no.of times*/
   SET I1 :=0, J1 :=0, COUNT:=0.
8. /*Returns the class in which it is classified*/
   Call GET_FOLDER(PATH) OR FOLDER=GET_FOLDER(PATH).
9. Call UPDATE_IFILE(FOLDER).
10. SET END_TIME:=CURRENT_TIME.
11. SET DURATION:= END_TIME-START_TIME.

Algorithm: FIND_COS_SIM(TEMP_FILE1)
Find Cosine Similarity between TEMP_FILE1 and TEMP_FILE2. This algorithm finds the cosine similarity between two files. HM is the HashMap<Double,String> which holds the cosine similarity and the corresponding file name. AL is the ArrayList<Double> which holds the cosine similarity.

1. SET LEN:=0, MAX:=Maximum_no_of_files_in_Database, FILE[] F1:= NEW FILE[MAX].
2. REPEAT WHILE LEN<=MAX
   FILE[LEN]:=NEW FILE(ST[LEN]).
3. FOR FILE FF IN F1
   3.1 SET A_PATH:=AbsolutePath(FF).
   3.2 REMOVE_STOP_WORDS(A_PATH).
   3.3 PORTER_STEMMING().
   3.4 Read OUT_STEMFILE and write it to TEMP_FILE2.
   3.5 SET A:=COSINE_SIMILARITY(TEMP_FILE1,TEMP_FILE2).
   3.6 SET S2:=GET_NAME(FF). /*Returns the name/absolute path of that file*/
   3.7 HM.PUT(A,S2).
   3.8 A1.ADD(A).
4. /*To set in descending order*/
   SET LEND:= SIZE(A1).
   FOR I=0 to 5, REPEAT
   4.1 FOR K=I+1 to LEND, REPEAT
   4.1.1 IF AL.GET(I)<AL.GET(K)
   SET TMP:= AL.GET(I).
   AL.SET(I,AL.GET(K)).
   AL.SET(K,TMP).

Figure 4: Algorithm of Adept kNN

4 Future Scope

The area of application of Adept kNN is almost similar to that of the original kNN

1. Text Mining
2. Agricultural Area
3. Financial Departments
4. Medicinal Field
5. Large Organizations

REFERENCES

[1] Based on study from http://www.academia.edu/4607757/Application_of_K-Nearest_Neighbor_kNN_Approach_for_Predicting_Economic_Events_Theoretical_Background

[2] Biju Issac, Nauman Israr. " Case Studies in Intelligent Computing: Achievements and Trends"

[3] Altman, N. S. (1992). "An introduction to kernel and nearest-neighbor nonparametric regression".

[4] Antti Ajanki AnAj. http://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm

[5] Based on the study from Jiawei Han, Micheline Kamber. "Data Mining Concepts and Techniques"

[6] http://research.cs.tamu.edu/prism/lectures/pr/pr_l8.pdf

[7] http://www.nickgillian.com/wiki/pmwiki.php?n=GRT.kNN