An Enhancement of k-Nearest Neighbor Classification Using Genetic Algorithm

Anupam Kumar Nath Syed M. Rahman Akram Salah

Department of Computer Science, North Dakota State University
258 IACC Building, Fargo, North Dakota 58105, USA
{Anupam.Nath, Syed.Rahman, Akram.Salah}@ndsu.edu

Abstract

K-Nearest Neighbor Classification (kNNC) makes the classification by getting votes of the k-Nearest Neighbors. Performance of kNNC is depended largely upon the efficient selection of k-Nearest Neighbors. All the attributes describing an instance does not have same importance in selecting the nearest neighbors. In real world, influence of the different attributes on the classification keeps on changing with time. To solve this problem, we have proposed an enhancement of kNNC where Genetic Algorithm (GA) has been applied for effective selection and upgrade of attribute set to find out k-Nearest Neighbors. Our experimental results demonstrate a significant improvement in classification accuracy in comparison with the conventional kNNC.

1 Introduction

Efficiency of kNNC depends largely upon the effective selection of k-Nearest Neighbors [3]. The limitation of conventional kNNC is that once we choose the criteria for k-Nearest Neighbors selection, the criteria remain unchanged. But this characteristic of kNNC is not suitable for many cases if we want to make a correct prediction or classification in real life.

An instance is described in the database by using a number of attributes and the corresponding values of those attributes. So similarity between any two instances is identified by the similarity of attribute values. But in real life data when we are describing two instances and are trying to find out the similarity between those two, similarities in different attributes do not weigh same with respect to a particular classification. Moreover, as with time more training data keeps on coming it may happen that similarity in a particular attribute value carries more or less importance than before. For example, say we are trying to predict the outcome of a soccer game based on the previous results. Now in that prediction, the place and the weather plays a very important role in the outcome of the game. But in future if all the soccer games are played in indoor stadiums then the field weather is no longer going to have same effect on the outcome of the game.

In Genetic Algorithm(GA) during each generation the current population are rated for their effectiveness as solutions, and on the basis of these evaluations, a new population of candidate structures is formed using specific `genetic operators' such as selection crossover, and mutation to find out even more optimized solution. In short GA keeps on upgrading the solution process with time to reach the best result [2].

GA has been applied in kNNC method previously. GA has been used in all most all the previous implementations to set up the voting weight of k Nearest Neighbors. According to our proposed method GA will be used in kNNC with a different approach. In the proposed enhancement, GA has been used to select the attribute set which are going to vote to find out the k_nearest Neighbors and to change the voting weight of the attribute set with time so that kNNC method can make the classification more efficiently.

Our approach provides kNNC more interactivity for changing its nearest neighbor selection criteria to make an efficient classification. We have implemented and compared our proposed method of kNNC with the conventional kNNC on loan approval dataset. We found that our method has significant improvements in the classification accuracy.

2 Proposed Method

In k-Nearest-Neighbor Classification (kNNC), the training dataset is used to classify each member of a "target" dataset. The structure of the data is that there is a classification (categorical) variable of interest ("buyer," or "non-buyer," for example), and a number of additional predictor variables (age, income, location...). Generally speaking, the algorithm is as follows:

1. For each row (case) in the target dataset (the set to be classified), locate the k closest members (the k nearest neighbors) of the training dataset. A Distance measure is used to calculate how close each member of the training set is to the target row that is being examined.
2. Examine the k nearest neighbors - which classification (category) do most of them belong to? Assign this category to the row being examined.
3. Repeat this procedure for the remaining rows (cases) in the target set.

In practical applications, typically, k is in units or tens rather than in hundreds or thousands [1].

In our new approach

- Instead of using all the attributes to find out the k Nearest neighbors we are going to use only q most important attributes and similarity in only in those attributes' value to find out the nearest neighbors.

(here $q = ½$ (number of total attributes)$\leq q <$ number of total attributes)

- Together with the weighting vote of the k neighbors we are proposing the weighting vote of the attributes to find out the k Nearest Neighbors.

- *GA operator* selection will be used for two different purposes.

  i) As the time progresses and more data keep on coming voting weight of the attributes are going to keep on changing as well as the voting weight of the neighbors.

  ii) If necessary, any of the attribute(s) of the attribute set, selected primarily for selection of the k Nearest Neighbors, is going to be replaced by an attribute(s) which was not a set of the initially selected set.

The outline of our proposed method has been described in figure 1. :

m, n: Integer

1. Divide all the attributes into category *X* & *Y***.**

2.

(a) Select *m* attributes from category X;

(b) Select *n* attributes from category Y;

(c) Use the crossover GA operator to build the new attribute set Z where total number of attributes will be *m*+ *n*;

3. Attributes of only *Z* set will vote (based on similarity) *according to the weight* to find out the k nearest neighbors.

4. These k neighbors will vote to predict the outcome.

5. Repetitive Selection algorithm will be applied to *change the attribute set Z* (if necessary) and *reassign the weight* of *the attributes* after p more data come in.

Figure 1.Algorithm of the proposed method

2.1 Overview of the Proposed Method

Step1:

For most of the cases the attributes which describe an instance in dataset can be broadly categorized into two different classes-

- *Class X Attributes:* Value of those attributes are not expected to change frequently i.e. relatively static attributes will be categorized as class X attributes. For example: an attribute namely *SEX*. A person may be Male or female and it is not going to change. So this is an attribute whose values are relatively constant and according to our definition will be categorized as class X.

- *Class Y Attributes:* Value of those attributes are expected to change frequently i.e. relatively dynamic attributes. For example: an attribute namely *JOB SATUS*. There is a chance of person getting or lose a job within a very short time period. So this is relatively dynamic attribute and according to our definition will be categorized as class Y.

Step 2:

Now for the attributes belonging to class X we have to select the m attributes which have the most influence on the classification.

We also have to find out the most influential n attributes *f*or the class Y attributes

We have proposed an algorithm for selection. But before applying the algorithm we have to assign weight to each of the attributes. Broadly there are two sorts of attributes. One are Boolean attributes i.e. those can have only two different values e.g. martial status (married/single), sex (male/female) etc. This sort of attributes we define as group A attributes. The other type of attributes we define as group B attributes can have numeric values of different ranges for example salary, deposit in account, age etc.

This grouping is completely different from previously mentioned basic classes- X and Y. This grouping will be done only for convenience in assigning weights to all the attributes describing an instance.

In order to assign weight to both groups of attributes our proposed algorithms are described in Appendix.

Once we have completed assigning weights to the attributes our proposed algorithms is going to select the most influential attributes with respect to our purpose. Figure 2 and Figure 3 are the algorithms for finding out the most important m and n attributes form class X and class Y respectively.

K=Total number of attributes in category X
L=Total number of attributes category X weighing greater then .5

1. Sort the attributes of category X according to the weight.

2. IF *L>K/2* THEN
   Select the first *K/2+1* attributes of the sorted list of category *X*

   ELSE IF *L<K/2* THEN
   Select the first *K/2-1* attributes of the sorted list of category *X*

   ELSE
   Select the first K/2 attributes of the sorted list of category X

Figure2: Selection algorithm for X category

V=Total number of attributes in category Y
U=Total number of attributes category Y weighing greater then .5

1. Sort the attributes of category X according to the weight.

2. IF U>V/2 THEN
   Select the first U/2+1 attributes of the sorted list of category Y

   ELSE IF U<V/2 THEN
   Select the first V/2-1 attributes of the sorted list of category Y

   ELSE
   Select the first V/2 attributes of the sorted list of category Y

Figure3: Selection algorithm for Y category

Step 3:

After applying these algorithms we create the attribute set consists of total m+n number of attributes. Now we will apply the conventional kNNC method with this chosen attribute set.

2.2 Repetitive Selection process to make the prediction criteria change with time

As mentioned earlier, in the proposed algorithm Repetitive Selection plays a very important part. This operator of GA which has been used to adjust the importance (i.e. weighting) of attributes with time and if necessary to change the primarily selected attribute set for voting. In Figure 6 our proposed algorithm of Repetitive Selection is described.

2.3 Modification of the proposed method

In the first proposed method weighting vote of the attributes has been used rather than weighting vote of the neighbors. Now to make the proposed method more effective we have also included the weighting vote of the neighbors. So our modified proposed algorithm is given in the Figure 7.

```
While (p<h) DO //   p number of new instances after the last selection & h is a
                    // constant number
        Begin

          FOR each of the attribute DO
          SET Weight_new[i]=Weighting // This will be using our proposed
                                                  // Weighting algorithm
        End

          FOR each of the attribute DO
           Begin

                SET W[i]:=W[i]*Weight_new[i]

            End

  FOR each attribute k that is not an attribute of voting attribute set DO
      FOR each attribute g that is an attribute of voting attribute set DO

       Begin
         IF Weight_new[k]>.7 and W[g]<.4 THEN

         Replace the attribute g   from voting attribute set with the attribute k

         End
```

Figure 4: Repetitive Selection algorithm

The main difference between this algorithm and our proposed algorithm of figure1 is in the step 4. *In the modified proposed algorithm once selected k nearest neighbors are not only going to vote but also they are going to vote according to the weight to predict the outcome.*

m, n:Integer

1. Divide all the attributes into category *X* & *Y***.**

2.

(a) Select *m* attributes from category X;

(b) Select *n* attributes from category Y;

(c) Use the crossover GA operator to build the new attribute set Z where total number of attributes will be *m*+ *n*;

3. Attributes of only *Z* set will vote (based on similarity) *according to the weight* to find out the k nearest neighbors.

4. These k neighbors will vote to predict the outcome.

5. Repetitive Selection algorithm will be applied to *change the attribute set Z* (if necessary) and *reassign the weight* of *the attributes* after *p* more data come in.

Figure 4.Algorithm of the modified proposed method

Now as the weight of the attributes keep on changing the neighbors vote weight are going to change accordingly. We are assigning weight to the neighbors based on the similarity of different instances on the same attribute set and importance of those each individual attribute with respect to the purpose.

3 Implementation of the proposed method

To prove the effectiveness of our proposed method we have implemented the proposed method in the data set which is available in the University of California, Irvine (UCI) Machine learning repository. The Loan Approval Database has the following description:

Good mix of attributes -- continuous, nominal with small numbers of values, and nominal with larger numbers of values 8900 instances, 15 attributes (like age, job status, sex, marital status, deposit amount in account, job experience etc.) some of them are with missing values .

To implement it we have used 500 instances as training data and then have tried to predict the class (i.e. weather a person would be a defaulter or not) for rest of the data. Then we have tried to compare it with the actual outcome.

All these data were collected over three different time periods i.e. a set of all the data were collected at a time then there was a time interval after that the second data set was collected and in the same way the third data set. This characteristic of the collected data set was suitable to test the effectiveness of our proposed modified kNNC methods.

3.1 Processing Before Implementation

For faster computation we have converted the attribute values into the binary form of 1s and 0s. We have used two different algorithms [Appendix] for doing so. One for previously defined group A attributes and other for the group B attributes.

3.2 Results of the implementation

As mentioned earlier the data we have worked on are collected over three time intervals. So we have compared the prediction quality for all three time intervals.

By using conventional kNNC, our proposed kNNC and the modified proposed kNNC method. The outcomes of the implementation are shown in the figure10, 11 and 12.

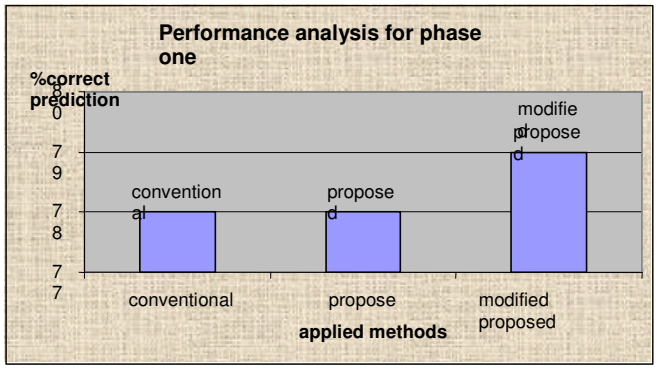


Figure10. Performance of different kNNC for time phase one.

We can notice from the graph of figure10 that in the first time phase the performance of all three methods is quite similar regarding the percentage of correct prediction. Our modified proposed method's performance is just a little bit better but not real significant.

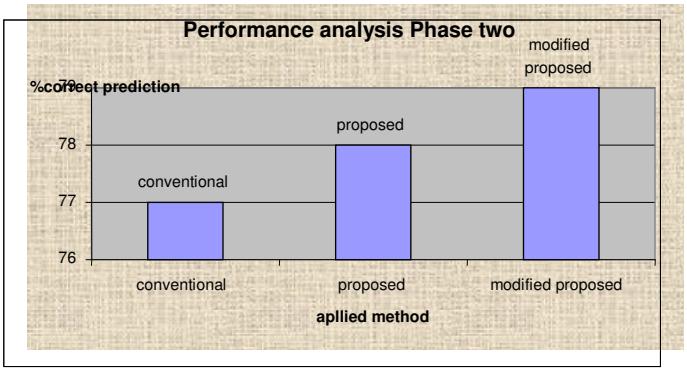


Figure 11.Performance of different kNNC for time phase two.

In the performance of time phase two it is noticeable that the performance of conventional kNNC is not as good as the other two and its percentage of correct prediction has been declined significantly.

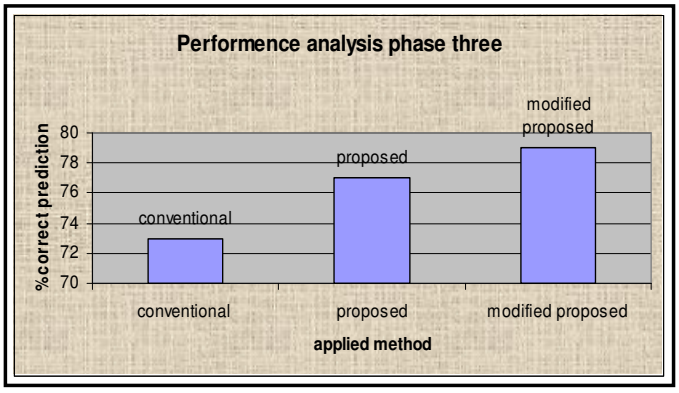


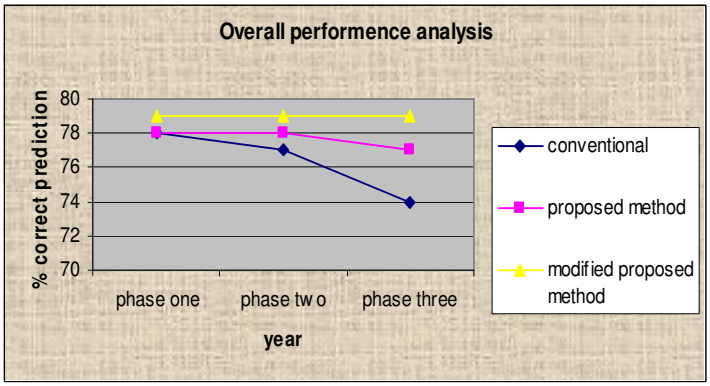
Figure12: Performance of different kNNC for time phase three

A significant improvement of performance can be found in the results of time phase three. Here performance of conventional kNNC method has degraded by almost 5 percentage. But our modified proposed method is still performing quite consistently regarding the percentage of correct prediction.

3.3 Overall performance analysis

The performance of three methods over all the three time phases is summarized in the figure 13. Here it can be noticed that as the time progresses performance of the

conventional kNNC is degrading significantly. On the other hand, both our proposed method outperforms the conventional method.



Figures 13: Overall Performance analysis.

Among our two proposed methods the later one's performance is even better than our first proposed one. The possible reason behind the better performance of our proposed methods is the up gradation of prediction/classification criteria by using the mutation algorithm.

So these results show the better performance of our proposed version of kNNC regarding the effective correct classification. It can be inferred that the proposed version of kNNC is especially suitable for the application domains where the scenario changes frequently with time.

4 Conclusions

In this paper we have proposed an enhancement k Nearest Neighbor classification method by using Genetic Algorithm. Our new approach is to provide kNNC more interactivity so that it can keep on changing its nearest neighbor selection criteria to make better prediction. We have implemented our newly proposed methods of kNNC together with the conventional one on real life data set. Considering the percentage of correct prediction our proposed methods have outperformed the conventional one.

But both of our proposed methods have some overheads like preprocessing of data before implementation and running the selection algorithm repetitively to change the selection as well as prediction criteria with time.

In future we would like provide more interactivity to the selection algorithm. In this paper the algorithm we have proposed to change the voting attribute set is a rigid one as the value used in the algorithm to repeat the repetitive selection algorithm is a

preset one. But to handle all sorts of real life data this value should be an adaptive one based on the dataset.

References

[1]http://www.resample.com/xlminer/help/k-NN/knn_intro.htm, web retrieve on January 10, 2005

[2]Patterson, Dan, W. “Artificial Intelligence and Expert systems”, Prentice Hall, 1990

[3] Maleq Khan, Qin Dong,William Perrizo . "K-Nearest Neighbor Classification on Spatial Data Streams Using P-Trees",, Pacific-Asian Knowledge and Data Mining Conf., May, 2002

Appendix

Preprocessing algorithm for group A Attributes:

```
For each attribute of Group A DO
        Begin

            IF A[i] [ j]= T && ACTUAL _OUTCOME=Positive THEN
                        SET C:=C+1

            ELSE IF A[i] [ j]= F && ACTUAL _OUTCOME=Negative THEN

                        SET C:=C+1
            ELSE
                        SET C:=C-1


            IF A[i] [ j]= F && ACTUAL _OUTCOME=Positive THEN
                                SET D:=D+1

            ELSE IF A[i] [ j]= T && ACTUAL _OUTCOME=Negative THEN
                                SET D:=D+1
            ELSE

                                SET D:=D-1


            IF C>D THEN DO

                        For i=1 TO n DO
```

```
                Begin

                    IF A[i] [j]=T  THEN

                    Set A[i] [j]:=1
                    ELSE
                            Set A[i] [j]:= 0
                 End

ELSE DO

            For i=1 TO n DO


             Begin

                  IF A[i] [j]=F  THEN
                     Set A[i] [j]:=1

                   ELSE
                   Set A[i] [j]:= 0
                End
          End
```

Preprocessing algorithm for group B Attributes:

```
For each of the group B attributes Do

 Begin
        Set C: =0;
        Set SUM: =0;

        For i=1 TO n (total number of training data) DO

          Begin

            ACTUAL OUTPUT=POSITIVE
                    SUM: =SUM+T[i][j] // j th attribute
                    C: =C+1

              T_AVG[j]:=SUM / C

             End

         For i=1 TO n DO

          Begin

              IF A[i] [j]>T_AVG[j] THEN
                            Set A[i] [j]:=1
             ELSE
                            Set A[i] [j]:= 0
           End

End
```