A Tweak on K-Nearest Neighbor Decision Rule

Tanmay Basu[1], **C. A. Murthy**[1], **and Himadri Chakrabarty**[2]
[1] Machine Intelligence Unit, Indian Statistical Institute, 203 B. T. Road, Kolkata-700108, India
[2] Department of Computer Scinece, Surendranath College, University of Calcutta, Kolkata - 700009, India

Abstract— *The k-nearest neighbor (knn) decision rule puts a point into a particular class if the class has the maximum representation among the k nearest neighbors of the point in the training set. If the difference between the number of points belonging to the two competing classes (i.e., two classes with maximum representation) among the k nearest neighbors is at least one then the point will be assigned to that class. This article presents a tweak on the knn rule to enhance the confidence of the knn voting process. This method proposes a discrimination criterion on the majority voting of knn by a predefined threshold to enhance the rigidity of the voting process. The method does not require the knowledge of neighborhood parameter k to execute knn. The empirical studies show the utility of the proposed method using some synthetic and well known benchmark data sets. It is observed from the experiments that the accuracy of the proposed method is significantly better than the traditional knn method.*

Keywords: Majority Voting, K Nearest Neighbor, Pattern Recognition

1. Introduction

The k-nearest neighbor (knn) rule is a very popular and simple classification technique. It classifies a test data point to a particular class which is most frequent in the neighborhood of the data point. The size of the neighborhood is determined by the predefined parameter k. Given a test data point x and a training sample set, the task of the k-nearest neighbor rule is to assign x to a particular class. It first finds the k-nearest neighbors from the training sample set using some distance function and assigns x to a particular class by taking a majority vote among the k-nearest neighbors.

In 1951, Fix and Hodges [1] introduced the nearest neighbor rule and in 1967, Cover and Hart [2] found out the classification error for k=1. It was shown that for k=1 and $n \to \infty$ the classification error is bounded by twice the Bayes error rate. A series of research works have been done on knn classification. B. V. Dasarathy has provided a number of a comprehensive collection of around 140 key papers [3] [4]. A rejection approach was proposed by Hellman [5]. In this method a point will be assigned to a particular class, if at least $l > k/2$ nearest neighbors belong to the same class, otherwise the method will refuse to classify the point. The distance weighted knn [6] gives different weights to the k nearest neighbors based on their distances to the test data point with closer neighbors having greater weights. Let $x_1, x_2, ..., x_k$ be the k nearest neighbors of x_0 arranged in increasing order of $\rho(x_j, p)$, $\forall j = 1, 2, ...k$, where ρ is the distance function. Here x_1, x_k are the first and kth nearest neighbor of x_0. The weight of the jth nearest neighbor was defined as $w_j = \frac{\rho(x_k,x_0)-\rho(x_j,x_0)}{\rho(x_k,x_0)-\rho(x_1,x_0)}$, if $\rho(x_k, x_0) \neq \rho(x_1, x_0)$ and $w_j = 1$, if $\rho(x_k, x_0) = \rho(x_1, x_0)$.

The test data point x_0 will then be assigned to the class for which the weights of the representatives of the class among k nearest neighbors sum to the greatest value.

Generally, for knn, a closed disk of radius r_k is taken as the neighborhood, where r_k is the distance between x and its k^{th} nearest neighbor. Then x would be classified to the class which has maximum number of representatives in this neighborhood. So different values of k can change the classification result and hence choice of k is crucial for proper classification. Existing theoretical results suggest that optimally k should tend to ∞ and $k/n \to 0$ hold only for $n \to \infty$ [7]. The usual cross validation technique [8] is used to estimate an optimal value of k. But effectively choosing an optimal k is still a difficult job. The cross-validation method uses the training data to select a single value of k, and then that selected value is used for classifying all observations. However, one should note that in classification problems, in addition to depending on the entire training sample, a good choice of k depends on the specific observation to be classified. A discussion on the bias and the variance of the posterior probability estimates for different k values is available in [9].

Sometimes, one may not be interested in classifying an observation to one of the classes. Suppose, for the chosen k, the same number of observations fall into the two competing classes, and thus no class has a majority of members. We then assign it arbitrarily to one of the competing classes. This assignment may not be intuitively satisfying if the point belongs to the intersection region between classes, where one may not always necessarily be interested classifying every point.

Let us consider two classes C_1 and C_2, and the nearest neighbors of a test point x_0 are ordered as follows:

$$C_1, C_2, C_1, C_2, C_2, C_1, C_1, C_2, C_1, ... \qquad (1)$$

For k=5, the data point will be classified to C_2, for k=6 there will be a tie and for k=7, the point will be classified to C_1 and so on. It reveals the fact that simple majority voting may not be appropriate. Basically, when there is more or less same representation from the competing classes among the nearest neighbors, we believe that the test data point should not be put into one of those classes. A way of formalizing more representation for one class is to put a lower bound β on the difference between the cardinalities of the two competing classes for the training set, and selecting the class with maximum representation.

The paper is organized as follows. Section 2 describes the proposed tweak on knn in detail. The experimental evaluation on both synthetic and benchmark data sets is given in section 3. Conclusions and discussion are presented in section 4.

2. A Tweak on KNN Decision Rule

Knn method takes decision on majority voting. For knn rule, a test data point is put in a particular class that has maximum number of representatives among the k nearest neighbors. There are no particular bounds on the discrimination criterion of majority voting. Sometimes, we assign a test data point to an arbitrarily chosen competing class when there is a tie between the competing classes. Choosing one of the competing classes randomly results in less confidence on the selection process. Additionally, suppose, for a given k, the difference between the number of nearest neighbors of competing classes, represented by γ_k, is 3. Then, for that k, intuitively, classifying the test data point to the class with maximum number of nearest neighbors would result in more confidence on our decision compared to the case of γ_k being 1 or 2. On the other hand, putting an arbitrarily large threshold on the value of γ_k (without looking at the size of the training set) may result in a bad classification strategy with many unclassified points.

We propose a discrimination criterion on the voting process of knn. Basically, the proposed method restricts the majority voting of knn with a predefined positive integer threshold, say β, to assign a data point to a predefined class. For the proposed method there is no need to select a fixed k prior to classifying a point. The method will start with an initial value of k as β. If the difference between the number of representatives of the best and the second best competing classes is β, then the point is classified to the best competing class. Otherwise the value of the neighborhood parameter k will be increased by one and the process will continue until the point is classified to a particular class. If the test data point is not classified till the process reaches the last point of the training set, then the test data point will remain unclassified. But this is rare and the experimental results reveal that this kind of situation arises when the number of points of the training set is very small. The initial value of k is taken as β. If for the k under consideration, the test data point can not be classified, then the value of k is increased by one. It may be noted here that β is the only parameter to the proposed method. The proposed method is named *tknn* (i.e., tweak on knn). Algorithm 1 describes the tknn method in detail.

Algorithm 1: TKNN for a Particular Test Data Point

```
Input: a) A set of training data points, X = $\{X_1, X_2, ..., X_n\}$ and n = $|X|$.
b) A set of predefined classes, C = $\{C_1, C_2, ..., C_m\}$ and m = $|C|$
c) Let y is a particular test data point and $\beta$ is an initial threshold.
Steps:
 1: L ← $\beta$
 2: for i ← 1 to n do
 3:     $dist_i$ ← Distance of y from $X_i$
 4: end for
 5: Sort $dist_i^{\prime}s$ in increasing order and the corresponding
    order of points are $Z_1, Z_2, ..., Z_n$
 6: Find the first L data points from $Z_i^{\prime}$ s
 7: $L_i$ ← Total number of data points ∈ $C_i$ and $\sum_{i=1}^{m} L_i$ = L.
 8: $L_{max_1}$ ← max$\{L_1, L_2, ..., L_m\}$
    $L_{max_2}$ ← max$\{\{L_1, L_2, ..., L_m\} - \{L_{max_1}\}\}$
 9: if $(L_{max_1} - L_{max_2})$ == $\beta$ then
10:     y ∈ $C_{max1}$ i.e y is classified to $C_{max1}$
11: else
12:     if L == n then
13:         Classification not possible, stop and exit.
14:     else
15:         L ← L + 1.
16:         if $L^{th}$ data point ∈ $C_i, \forall i \in m$ then
17:             $L_i \leftarrow L_i + 1$
18:         end if
19:         Goto step 8
20:     end if
21: end if
```

Like knn, the proposed algorithm tknn will first find the distances of a test data point from all the training set sample points and will sort it in increasing order of dissimilarity. But unlike knn, initially, it will take only the first β points from the ordered set and performs the majority voting with a criterion that the difference between the total number of data points of the competing two classes (L_{max_1} & L_{max_2}) among the nearest neighbors is equal to β. If it is so then the point will be labeled with that particular class (C_{max1}), possessing maximum number of representatives. Otherwise it will check the next point and performs the same operation. This is continued until it has reached the end of the ordered training sample set.

Consider sequence 1 and assume $\beta = 3$. Let there be 9 data points in the training set and the order of those points according to their dissimilarity with x_0 in increasing order are shown in sequence 1. Initially $L_{max_1} = 2$ and $L_{max_2} = 1$ and $(L_{max_1} - L_{max_2} < 3)$. So L (total number of nearest neighbor traversed by tknn so far) has to be increased by one to execute the same operation and so on. Thus, at the end $L_{max_1} = 5$ and $L_{max_2} = 4$, which concludes that the test data point will remain unclassified.

Tknn will refuse to classify a point to a particular class, if the majority voting is not discriminating i.e., if $(L_{max_1} - L_{max_2}) < \beta$ and it reaches the last point of the ordered training sample set. These points will remain unclassified. The decision should not be taken on the difference of only one vote. The difference should not be less than the threshold (β). Among the nearest neighbors of the test data point there should be sufficiently many representatives of the class to which the point will be assigned by the classifier than the other classes. Intuitively, the likelihood of correct classification is more for the proposed scheme than the knn. Consider the situation of the points which lie on the boundary of two adjacent classes. Knn method can misclassify those points with the weak voting criterion. But for tknn, either it would not be able to classify the point or it is likely to correctly classify it. If it is able to classify the point then it has sufficiently many representatives among the nearest neighbors in support of the corresponding class, otherwise the point will remain unclassified.

Selection of k is a very difficult problem for knn. Wrong selection of k for knn may degrade the quality of knn. Tknn has no requirement of a fixed value of k to execute the method, though the value for β is to be fixed by the user. In the experimental evaluation we have shown that one of the value of β among $\beta = 2, 3, 4$ is sufficient. It is to mention that a particular β will be used for every point in the test set. The algorithm will stop only when it satisfies the said voting criterion.

In the proposed procedure, we are interested in ensuring significant difference between the number of representative points of majority class and its competing classes. Intuitively, such a restriction may not result in good performance of the proposed method if the size of the training data set is small since smaller training data sets may not have enough sample points to ensure such a restriction. Larger training data sets are likely to produce better results for the proposed method.

3. Experimental Evaluation

In this section we present the experimental results on synthetic and benchmark data sets to compare the performance of the proposed tknn method with the traditional knn method.

3.1 Parameter Selection

The knn classifier classifies a point on the basis of k nearest neighbors. Note that the value of k can vary from one to the total number of points in the training set. But k must be fixed for all test data points. If k is very high then the computational complexity of the classifier will be very high which is not desirable. So k should not be as high as possible. $\beta = 1$ is actually the one nearest neighbor (1-NN) method. So in the experimental evaluation 1-NN is not used for comparison. Here 10 fold cross validation is performed on the training set varying k from 2 to 15. The k value which provides best accuracy among these 14 values is used in the experiments of the test data points for knn and wt-knn. The knn and wt-knn method were executed using this k value and the error rate along with their standard errors are reported in Table 1 and Table 2 for comparison and the particular k value is reported in a separate column.

Tknn is dependent on the β values. A high β value can produce a large number of unclassified points if the training set size is not very high. So here we have restricted the values of β to 2, 3, 4 for comparison with the other methods. Here also 10 fold cross validation is performed on the training sets using $\beta = 2, 3, 4$. The β value for which tknn gives best accuracy is used in the experiment of the test set. It is to be mentioned that euclidean distance is used in all the experiments.

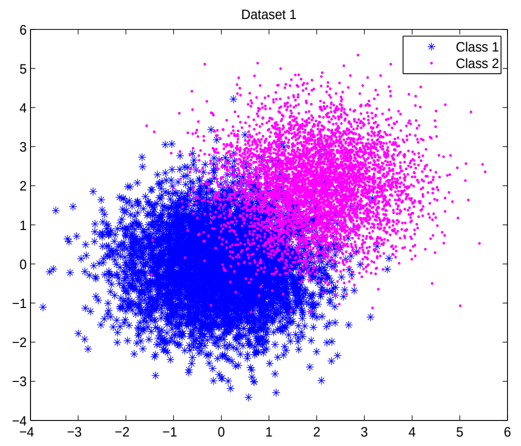
3.2 Description of Simulated Data Sets

This section presents the performance of the proposed method on some randomly simulated data sets. Here, different sizes and shapes of classes are considered for performance evaluation. Suppose the number of classes is c, and the conditional probability density function for the i-th class is $p_i(x)$, and the prior probability of i-th class is π_i. Then the mixture probability density function is $p(x) = \sum \pi_i p_i(x)$. Except Dataset 6, the artificial data sets are generated by considering different p_is. Different sizes are achieved by considering different prior probabilities of classes. For each of the data sets (except for Dataset 6), we have generated 10 independent sets of points and reported the average misclassification percentage along with the standard error for misclassification rates. For the first three data sets each of the two populations is normally distributed and has the same prior probability 0.5. For each of the first five data sets the size of the test set is 10000 and there are five training sets of size 100, 500, 1000, 1500, 2000 respectively. The description of all the data sets is given below.

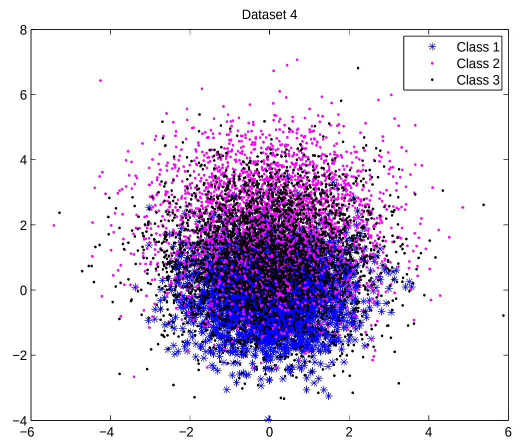
Dataset 1 : The mean and the scatter matrix for the first population are μ_1 = (0, 0) and $\Sigma_1 = I_2$ (identity matrix) respectively and those for the second population are μ_2 = (2, 2) and $\Sigma_2 = I_2$ respectively. The data set is shown in Figure 1(a).

Dataset 2 : The mean and the scatter matrix for the first population are μ_1 = (0, 0, 0) and $\Sigma_1 = I_3$ (identity matrix) respectively and those for the second population are μ_2 = (2, 2, 2) and $\Sigma_2 = I_3$ respectively.

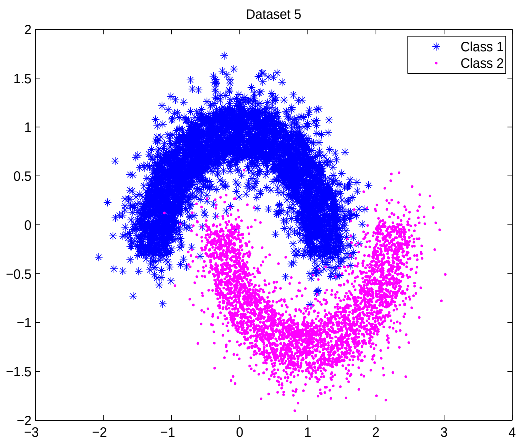
Dataset 3 : The mean and the scatter matrix for the first population are μ_1 = (0, 0, 0, 0) and $\Sigma_1 = I_4$ (identity matrix)



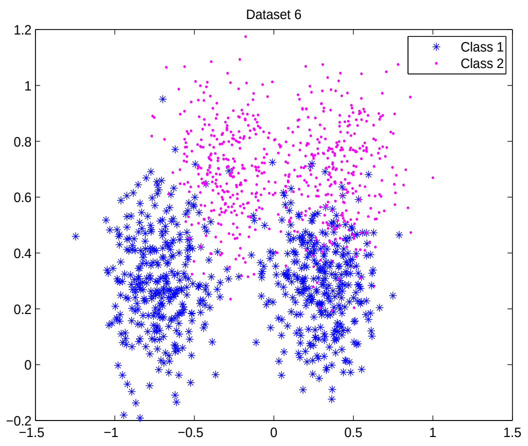
(a)



(b)



(c)



(d)

Fig. 1: Simulated Data Sets

respectively and those for the second population are $\mu_2 = (2, 2, 2, 2)$ and $\Sigma_2 = I_4$ respectively.

Dataset 4 : Here we considered three populations which are normally distributed and having prior probabilities 0.6, 0.1, 0.3 for population1, population2 and population3 respectively. The mean and the scatter matrix for the first population are $\mu_1 = (0,\ 0)$ and $\Sigma_1 = I_2$ (identity matrix) respectively. The same for the second population are $\mu_2 = (0,\ 2)$ and $\Sigma_2 = 2I_2$, and for the third population are $\mu_3 = (0,\ 1)$ and $\Sigma_3 = 2I_2$ respectively. The data set is shown in Figure 1(b).

Dataset 5 : This data set has two classes with equal prior probabilities. For obtaining an unusual shape for a class, we initially considered a curve which has that basic shape, and added a normal distribution with mean at the origin for every point on the curve. The data set is shown in Figure 1(c).

Dataset 6 : This data set was developed by Ripley[1] [8] and used by Holmes [4]. The data set is a two class classification problem where each population is an equal mixture of two bi-variate normal distributions. A total of 1250 points are given. A training set of 250 points is used and the model is tested on a set of 1000 points. The training set is given by the authors, and thus there is only one test set here. The data set is shown in Figure 1(d).

3.3 Analysis on Simulated Data Sets

Table 1 shows the comparative performances of the proposed method with weighted voting knn method and the usual knn method on the simulated data sets stated above. For Dataset 6, which has separate training and test sets, we have reported the test set misclassification rates for different classifiers in the following way. If a classifier leads to a test set error rate r, the corresponding standard error is taken as $\sqrt{r(1-r)/n}$, where n is the size of the test set [10]. The misclassification rate (MR) of tknn is determined as follows:

[1]www.stats.ox.ac.uk/pub/PRNN/

Table 1: Misclassification rates (in %) of different classification methods and their standard errors on various simulated data sets

Data Set Description	Training Set Size	β	L (avg)	UP[1]	TKNN	TKNN (including UP)[2]	K (knn)	KNN	K (wt-knn)	Wt-KNN
Dataset 1 (No. of Attributes = 2, No. of Classes = 2, Test Set Size = 10000)	100	4	6	3	**8.16** (0.003)	8.17 (0.003)	14	**8.16** (0.002)	15	8.59 (0.005)
	500	4	6	0	**8.13** (0.001)	**8.13** (0.001)	15	8.25 (0.002)	15	8.50 (0.003)
	1000	4	6	0	**8.18** (0.001)	**8.18** (0.001)	14	8.22 (0.002)	15	8.48 (0.002)
	1500	4	6	0	**8.13** (0.002)	**8.13** (0.002)	15	8.19 (0.002)	15	8.43 (0.002)
	2000	4	6	0	**8.12** (0.0009)	**8.12** (0.0009)	15	8.19 (0.001)	15	8.41 (0.001)
Dataset 2 (No. of Attributes = 3, No. of Classes = 2, Test Set Size = 10000)	100	4	6	0	4.82 (0.005)	4.82 (0.005)	14	**4.74** (0.004)	15	5.18 (0.006)
	500	4	6	0	**4.45** (0.001)	**4.45** (0.001)	14	4.46 (0.001)	15	4.95 (0.003)
	1000	4	5	0	**4.33** (0.001)	**4.33** (0.001)	15	4.36 (0.001)	15	4.56 (0.001)
	1500	4	5	0	**4.32** (0.001)	**4.32** (0.001)	15	4.38 (0.001)	15	4.54 (0.001)
	2000	4	5	0	**4.39** (0.001)	**4.39** (0.001)	15	4.43 (0.001)	15	4.58 (0.001)
Dataset 3 (No. of Attributes = 4, No. of Classes = 2, Test Set Size = 10000)	100	4	5	1	**2.47** (0.003)	**2.47** (0.003)	14	2.50 (0.002)	15	2.72 (0.004)
	500	4	5	0	**2.26** (0.0007)	**2.26** (0.0007)	15	2.32 (0.0009)	15	2.47 (0.001)
	1000	4	5	0	**2.20** (0.0005)	**2.20** (0.0005)	14	2.24 (0.0007)	15	2.37 (0.001)
	1500	4	5	0	**2.19** (0.0008)	**2.19** (0.0008)	14	2.22 (0.0008)	15	2.35 (0.001)
	2000	4	5	0	**2.23** (0.0005)	**2.23** (0.0005)	15	2.26 (0.0004)	15	2.37 (0.001)
Dataset 4 (No. of Attributes = 2, No. of Classes = 3, Test Set Size = 10000)	100	4	16	191	**46.67** (0.01)	**46.98** (0.01)	14	47.42 (0.009)	15	48.83 (0.01)
	500	4	17	0	**46.41** (0.006)	**46.41** (0.006)	15	47.00 (0.005)	15	48.82 (0.007)
	1000	4	16	0	**46.48** (0.01)	**46.48** (0.01)	15	47.25 (0.01)	15	48.66 (0.01)
	1500	4	17	0	**46.47** (0.006)	**46.47** (0.006)	14	47.29 (0.007)	15	48.57 (0.007)
	2000	4	17	0	**46.25** (0.005)	**46.25** (0.005)	14	47.19 (0.006)	15	48.60 (0.008)
Dataset 5 (No. of Attributes = 2, No. of Classes = 2, Test Set Size = 10000)	100	2	3	0	4.26 (0.006)	4.26 (0.006)	8	4.23 (0.006)	12	**4.17** (0.005)
	500	4	5	0	**3.72** (0.001)	**3.72** (0.001)	13	3.73 (0.001)	15	3.83 (0.001)
	1000	4	5	0	**3.67** (0.001)	**3.67** (0.001)	15	3.71 (0.001)	15	3.77 (0.001)
	1500	4	5	0	**3.69** (0.001)	**3.69** (0.001)	13	3.71 (0.0009)	15	3.79 (0.001)
	2000	4	5	0	**3.76** (0.001)	**3.76** (0.001)	15	3.80 (0.0009)	15	3.87 (0.002)
Dataset 6 (Test Set Size = 1000)	250	4	7	0	**9.00** (0.009)	**9.00** (0.009)	15	9.50 (0.009)	15	10.80 (0.009)

[1] UP stands for unclassified points. [2] Here the UPs are treated as misclassified points and the misclassification rates are measured as usual.

$$MR = \frac{MP}{TP-UP},$$

where MP is the number of misclassified points, TP is the size of the test set, and UP is the number of points remaining unclassified. Basically we are discarding the unclassified points from the total set of points to measure the accuracy. In the experimental results the performance of tknn are measured using two separate misclassification rates. The first one is using the above mentioned misclassification rate and second one is using the usual misclassification rate (here we treat every unclassified point as misclassified point).

Table 1 shows the misclassification rates of tknn, knn and wt-knn methods along with their standard errors (in parentheses). The k values are noted in separate columns for each of the knn and wt-knn methods which is used in the experiments of the test set points. Tknn observes first L nearest neighbors of point to determine its class. The average of the L values ($\lceil L \rceil$ is used here) of all the test data points (excluding the unclassified points) is noted in a separate column for each data set in this table. The β value which is used for each of the data sets is noted in another column.

It is observed from Table 1 that tknn performs better than the other methods in all cases except for one case of dataset 2 when training set size is 100 and one case of dataset 5 when training set size is 100. Let us consider the case of dataset 2 when training set size is 100. The misclassification rate of knn (4.74) has an edge over tknn (4.82). It needs to be checked whether these such differences are significant i.e., whether 4.74 is significantly different from 4.82. For all other cases when tknn performs better than the other methods, it needs to be judged whether tknn performs significantly better than the other methods.

A generalized version of paired *t-test* is suitable for testing the equality of means when the variances are unknown. This problem is the classical Behrens-Fisher problem in hypothesis testing and a suitable test statistic[2] is described and tabled in [11] and [12], respectively. It has been found that whenever a method has an edge over other method, the test showed significant difference in the values for the level of significance 0.05. No testing is done on Dataset 6.

Remarks : It is observed from the experiments that tknn performs better than knn and wt-knn if the training set size is large ($\geq$ 500). Tknn is useful when the training set size is large. For large training set, neighborhood selection may be difficult for knn and wt-knn. But from the experiment it is seen that one of the β values from $\beta = 2, 3, 4$ is sufficient for tknn. On the other hand it is also true that the essence of tknn can not be reflected if the training set size is very small.

[2]The test statistic is of the form $t = \frac{\bar{x}_1 - \bar{x}_2}{\sqrt{s_1^2/n_1 + s_2^2/n_2}}$, where $\bar{x}_1, \bar{x}_2$ are the means, s_1, s_2 are the standard deviations and n_1, n_2 are the number of observations

3.4 Analysis on Benchmark Data Sets

The proposed algorithm has been evaluated using 8 representative data sets from UCI [13] machine learning repository. The data sets used here are well known and the descriptions of the data sets can be obtained from the online UCI repository[3]. The sizes of the data sets range from 150 to 4601 with the dimensionality between 4 and 57 including both two class and multiclass data. The data sets are evaluated (i.e., finding optimal value of k and finding average misclassification rate and standard error for knn and wt-knn,) using 10 fold cross validation. Note that the results of cross validation depend on the partition of the data set. Thus, to make the results partition independent, the experiments have been executed 100 times for every 10 fold cross validation. Out of these 8 data sets Monks2 and Spect Heart data sets have specific training and test sets, and thus no cross validation is performed on these data sets. Table 2 shows the average misclassification rates of the 100 independent runs along with their standard errors of the data sets for tknn and the other methods. Here also individual values of k for knn and wt-knn and the values of β for tknn for each of the data sets has been noted in separate columns in this table.

From Table 2, it can be seen that tknn performs better than knn and wt-knn method for Abalone, Balance Scale, Heart Disease (Hungarian), Monks2, Pima Indian Diabetes and Spambase data sets. It has been found using paired *t-test* (discussed above) that tknn performed significantly better than the other methods for the level of significance 0.05 for these data sets (except for Monks2, Spect Heart data sets where the t-test could not be performed since the training set is not changed). It may be noted that the sizes of the training sets for the other data sets (i.e., Iris, Spect Heart) for which tknn has not provided better results, are less than 150.

Remarks : The performance of knn was slightly better than the proposed method for the *Iris* data set. The paired t-test shows that this difference was significant. The following points are the nearest neighbors of a particular data point of the iris data set.

1 3 2 2 1 1 3 1 1 1 1 1 3 2 1 1 1 1 1 1 1 1 1 2 2 2 2 3 3
2 2 3 1 2 1 . . .

For k = 3, 4 the point will be classified by knn to class 2, and actually it belongs to class 2. But tknn will classify it to class 1 for β =2, 3, 4. Note that the third and the fourth nearest neighbors belong to class 2. Then, the next point belonging to class 2 is the 14th nearest neighbor. This is an extreme example, and these examples change the pattern of results significantly when the sizes of the training data sets are small.

[3]http://www.ics.uci.edu/~mlearn/MLRepository.html

Table 2: Misclassification rates (in %) of different classification methods and their standard errors on various benchmark data sets

Data Set Name	No of Data Points	No of Classes	No of Attributes	β	L (avg)	TKNN	TKNN (including UP)#	k (knn)	KNN	k (wt-knn)	Wt-KNN
Abalone	4177	28	8	4	38	**73.44** (0.003)	**73.44** (0.003)	15	73.91 (0.003)	15	75.40 (0.003)
Balance Scale	625	3	4	4	8	**10.53** (0.002)	**10.53** (0.002)	10	10.71 (0.005)	15	12.62 (0.005)
Heart Disease (Hungarian)	294	5	14	4	9	**36.06** (0.0005)	**36.06** (0.0005)	13	36.28 (0.003)	13	39.05 (0.007)
Iris	150	3	4	4	5	3.56 (0.006)	3.56 (0.006)	15	**2.96** (0.007)	15	3.58 (0.006)
Monks2*	601 (169)	2	6	3	5	**32.87** (0.02)	**32.87** (0.02)	12	**32.87** (0.02)	3	32.87 (0.02)
Pima Indian Diabetes	768	2	8	4	11	**28.52** (0.01)	**28.52** (0.01)	12	28.57 (0.01)	14	29.46 (0.01)
Spambase	4601	2	57	2	3	**18.14** (0.002)	**18.14** (0.002)	3	18.87 (0.003)	7	19.97 (0.001)
Spect Heart*	267 (80)	2	22	2	3	43.85 (0.03)	43.85 (0.03)	14	**39.57** (0.03)	5	41.17 (0.03)

# Here misclassification rate = Total number of misclassified points/Total data points. The UPs are treated as misclassified points.

* The data sets have specific training set and test set. The size of the training set is given below the total instance (in ( )).

Note that, for each increase in the value of k, the distance from the test point increases, thereby, the confidence on either the majority voting of knn or the difference in cardinalities of competing classes for tknn would decrease.

4. Conclusions

The knn method is one of the most fundamental and simple classification methods for statistical pattern recognition. The majority voting of knn groups a point to a class if the best competing class wins over the second best competing class by at least one vote. For a tie we can arbitrarily assign a point to a class. But for high dimensional data set with two or more number of classes this weak voting criterion may not be suitable. Another major issue of knn classification rule is to select an optimal value for neighborhood parameter k. Generally cross validation or some resampling methods are used to select a value of k from the labeled data.

We proposed a discrimination criterion on the voting process of knn which applies a threshold (β) on the winning method of the voting process. The advantages of the proposed method are as follows: a) tknn would not require a prior knowledge of neighborhood parameter k, b) it will refuse to classify a data point if its margin of winning vote is not sufficient. Thus tknn may reduce the misclassification rate.

Tknn is dependent on β. The experimental results show that one of the values 2, 3, 4 of β would provide better classification results if the size of the training set is not small. Thus, instead of checking several values of k, one can look for the best among β=1, 2, 3, 4. For smaller sample sizes, other classification methods may perform better than the proposed one.

References

[1] E. Fix and J. Hodges, "Discriminatory analysis, nonparametric discrimination: Consistency properties," Technical Report 4, USAF School of Aviation Medicine, Randolph Field, pp. 261-279, Texas, USA, Tech. Rep., 1951.

[2] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Transactions on Information Theory*, vol. 13, no. 1, p. 21âĂŞ27, 1967.

[3] B. Dasarathy, *Nearest Neighbor NN Norms: NN Pattern Classification Techniques*. McGraw-Hill Computer Science Series. IEEE CS Press, 1991.

[4] C. C. Holmes and N. M. Adams, "A probabilistic nearest neighbor method for statistical pattern recognition," *Journal of Royal Statistical Society*, vol. 2, no. 64, p. 295âĂŞ306, 2002.

[5] M. Hellman, "The nearest neighbor classification rule with a reject option," *IEEE Transactions on Systems, Man, Cybernetics*, vol. 3, p. 179âĂŞ185, 1970.

[6] S. A. Dudani, "The distance weighted k nearest neighbor rule," *IEEE Transactions on Systems, Man, Cybernetics*, vol. SMC-6, p. 325âĂŞ327, 1976.

[7] K. Fukunaga and L. Hostetler, "Optimization of k-nearest neighbor density estimates," *IEEE Transactions on Information Theory*, vol. 19, pp. 320–326, 1973.

[8] B. D. Ripley, *Pattern Recognition and Neural Networks*. Cambridge University Press, Cambridge, 1996.

[9] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, 2nd ed. John Wiley and Sons, 2000.

[10] A. K. Ghosh, "On nearest neighbor classification using adaptive choice of k," *Journal of Computational and Graphical Statistics*, vol. 16, no. 2, p. 482âĂŞ502, 2007.

[11] E. Lehmann, *Testing of Statistical Hypotheses*. New York: John Wiley, 1976.

[12] C. R. Rao, S. K. Mitra, A. Matthai, and K. Ramamurthy, Eds., *Formulae and Tables for Statistical Work*. Calcutta: Statistical Publishing Society, 1966.

[13] C. L. Blake and C. J. Merz, "Uci repository of machine learning databases," *Department of Information and Computer Science, University of California, Irvine*, 1998.