Nearest Cluster Classifier

Hamid Parvin, Moslem Mohamadi, Sajad Parvin, Zahra Rezaei, Behrouz Minaei
Nourabad Mamasani Branch
Islamic Azad University
Nourabad Mamasani, Iran
hamidparvin@mamasaniiau.ac.ir, { mohamadi, s.parvin, rezaei, b_minaei }@iust.ac.ir

ABSTRACT: *In this paper, a new classification method that uses a clustering method to reduce the train set of K-Nearest Neighbor (KNN) classifier and also in order to enhance its performance is proposed. The proposed method is called Nearest Cluster Classifier (NCC). Inspiring the traditional K-NN algorithm, the main idea is to classify a test sample according to the tag of its nearest neighbor. First, the train set is clustered into a number of partitions. By obtaining a number of partitions employing several runnings of a simple clustering algorithm, NCC algorithm extracts a large number of clusters out of the partitions. Then, the label of each cluster center produced in the previous step is determined employing the majority vote mechanism between the class labels of the patterns in the cluster. The NCC algorithm iteratively adds a cluster to a pool of the selected clusters that are considered as the train set of the final 1-NN classifier as long as the 1-NN classifier performance over a set of patterns included the train set and the validation set improves. The selected set of the most accurate clusters are considered as the train set of final 1-NN classifier. After that, the class label of a new test sample is determined according to the class label of the nearest cluster center. Computationally, the NCC is about K times faster than KNN. The proposed method is evaluated on some real datasets from UCI repository. Empirical studies show an excellent improvement in terms of both accuracy and time complexity in comparison with KNN classifier.*

Keywords: Nearest Cluster Classifier, K-Nearest Neighbor, Combinational Classification

Received: 11 October 2011, Revised 3 December 2011, Accepted 8 December 2011

1. Introduction

KNN classifier is one of the most fundamental classifiers. It is also the simplest classifier. It could be the first choice for a classification study when there is little or no prior knowledge about the data distribution. The KNN classifies a test sample x by assigning it the label most frequently represented among the K nearest samples; in other words, a decision is made by examining the labels on the K-nearest neighbors and taking a majority vote mechanism. KNN classifier was developed from the need to perform discriminant analysis when reliable parametric estimates of the probability densities are unknown or difficult to determine. In 1951, Fix and Hodges introduced a non-parametric method for pattern classification that has since become known the K-nearest neighbor rule [1] and [15]. Later in 1967, some of the formal properties of the K- nearest neighbor rule have been worked out; for instance it was shown that for $K = 1$ and $n \to \infty$ the KNN classification error is bounded above by twice the Bayes error rate [2]. Once such formal properties of KNN classification were established, a long line of investigation ensued including new rejection approaches [3], refinements with respect to Bayes error rate [4], distance weighted approaches [5-6], soft computing [7] methods and fuzzy methods [8-9].

Some advantages of KNN include: its simplicity to use, its robustness to learn in a noisy training data (especially if it uses the inverse square of weighted distances as the "distance metric"), and finally its effectiveness in learning at a large scale training dataset. Although KNN has the mentioned advantages, it has some disadvantages such as: its high computation cost (because it needs to compute distance of each query instance to all training samples); its need to a large memory (in proportion with the size of training set); its low performance in multidimensional data sets; its sensitivity to well-setting of parameter *K* (number of nearest neighbors); its sensitivity to the used distance metric; and finally there is no solution for the weighting consideration of the features [10].

The computational complexity of the nearest neighbor algorithm, in both space (storage of prototypes) and time (distance computation) has received a great deal of analysis. Suppose we have *N* labeled training samples in *d* dimensions, and seek to find the closest to a test point *x* (*K* = 1). In the most naive approaches we inspect each stored point in turn, calculate its Euclidean distance to x, retaining the identity only of the current closest one. Each distance calculation is O(*d*), and thus this search is O (dN^2) [10].

ITQON et al. in [11] proposed a classifier, TFkNN, aiming at upgrading of distinction performance of KNN classifier and combining plural KNNs using testing characteristics. Their method not only upgrades distinction performance of the KNN but also brings an effect stabilizing variation of recognition ratio; and on recognition time, even when plural KNN classifiers are performed in parallel, by devising its distance calculation it can be done not so as to extremely increase on comparison with that in single KNN classifier.

In this paper a new approach that is based on the idea of KNN classifier is proposed. It can augment the performance of KNN classifier in terms of the accuracy, the time complexity and the memory complexity. This method which is called that named Nearest Cluster Classifier (NCC), applies the clustering techniques to reduce the number of training prototypes. Despite of reducing train samples, the clustering will cause to find the natural groups of data.

The rest of the paper is organized as follows. Section 2 expresses the proposed NCC algorithm. Experimental results are addressed in section 3. Finally, section 4 concludes the paper.

2. NCC: Nearest Cluster Classifier

In all experimentations the 10-fold cross validation is employed to obtain the performance of the NCC algorithm. In 10-fold cross validation the dataset is randomly partitioned into 10 clusters. Considering each partition as test set, *PTeS*, and the other data as train set, *PTrS*, the NCC algorithm reaches 10 experimentations.

Averaging the performances of the NCC algorithm over all 10 test sets, the final performances of the NCC algorithm is obtained. In each experimentation, the train set, *PTrS*, is divided into two sub-partitions, train sub-set, *TS*, and evaluation (validation) sub-set, *VS*. The main idea of the proposed method is assigning the data to the nearest cluster who is naturally consisted the neighbor points. To implement the idea, first, the samples of the train sub-set, *TS*, are clustered into *k* clusters where *k* is a number equal to or greater than the number of real classes, *c*, and equal to or less than 2**c*. The clustering is done using a simple rough clustering algorithm, *clus_alg*. Then, considering the obtained cluster centers as new points, their labels are determined using the simple majority vote mechanism. Indeed the label of a cluster is obtained based on a KNN classifier over train set where *K* is equal to an input parameter, *pq*. The *condition* is computed as equation 1.

$$condition(cvp, pq) = \begin{cases} cvp \geq pq \\ cvp \geq \frac{pq}{2} \\ cvp \leq \frac{pq}{2} \end{cases} \quad (1)$$

For example, if *condition(cvp,pq)* = *cvp* > $\frac{pq}{2}$; it means that each cluster center that is labeled with less than $\frac{pq}{2}$ votes in KNN classifier (using *PTrS* as train set) is immediately omitted; otherwise it is added in a pool of clusters, *PC*. This procedure is

iterated as many as *Maxiteration*. So finally the NCC algorithm has a pool of clusters with $Maxiteration\, c^3(Maxiteration \times k \times k \times \frac{k}{2})$ clusters at most.

The quality of obtained clusters is evaluated employing a 1-NN with considering *PTrS* as test set. Each pattern of the *PTrS* is used as a test sample; determining the nearest cluster, its label is assigned to the sample. After that, in comparison with the ground true labels of data, the accuracy of the obtained classifier is derived. So, after obtaining the pool of clusters, *PC*, the NCC algorithm iteratively selects them into a subset of pool of clusters (*SPC*) if the accuracy of a 1-NN using *SPS* as train set over *PTrS* is improved.

In test phase of the NCC algorithm, a new test sample is assigned to the label of the nearest cluster center. The pseudo code of training phase of the Nearest Cluster Classifier algorithm is shown in Figure 1. Until here, the training of the NCC is finished. After here, any test samples are classified using the trained classifier.

In the rest of this section the proposed method is described in detail, answering the questions, how to cluster the train set, how to determine the class labels of cluster centers and how to find the final classifier to classify a test sample.

Input:
PTrS: patterns of train set
PTeS: patterns of test set
c: the number of classes
clus_alg: clustering algorithm
pq: the threshold for assigning a label to a cluster center
Maxiteration: the maximum of the allowed Iterations
Condition: a condition for decision
Output:
accuracy: accuracy of NCC over *PTeS*
PC = {}
partition *PTrS* into two clusters: *TS* and *VS*
For *i* = 1 to *Maxiteration*
1. *P* = *clus_alg*(*TS*, *k*) where 2 * *c*e ≥ *k*e ≥ *c*; resulting cluster centers P_i
2. For each $p \in P$
if *condition*(*cvr*, *pq*), then *pi* will be added to the set *PC;* where *cvr* is the maximum number of the *pq* nearest patterns of P_i in *PTrS* that have consensus vote to an identical label
SPC={}
cur_acc=0
For each $q \in PC$
1. *TTS* = SPC ∪ q
2. *acc* = 1 - NN(*TTS*, *PTrS*)
3. if(*acc* > *cur_acc*) *SPC* =*TTS*
acc=1-NN(*TTS*, *PTeS*)

Figure 1. Pseudo-code of training phase of the nearest cluster classifier algorithm

2.1 Determining the Label of Cluster Centers

In this section, we explain how the class labels are used to specify the labels of the cluster centers which are explanatory points of the clusters. There are some combining methods to aggregate the class labels of the cluster members. When the individualvotes of classifiers are crisp (not soft/fuzzy), the Simple Majority Vote approach is the common logical one that votes a pattern x to a class j if a little more than n/c of classifiers (here cluster members) assigns to class j [12], where n and c stand for the number of cluster members and the number of classes, respectively. In the paper, the majority vote mechanism is used to assign a class label to cluster centers. It means that pq nearest neighbors of a cluster center in the *PTrS* make a consensus decision about its label.

2.2 Evaluation of the Cluster Centers

There are many methods to evaluate the clustering result. They may use whether external indices, whether internal indices or relative indices [13]. External index needs further information to evaluate the clusters. In the paper, the *PTrS* is used to measure the performance of the different clusterings. It is a kind of external index usage. First, the NCC algorithm is trained on the *TS*. Then, by executing the trained classifier on the *PTrS*, the accuracy of this method is obtained using the ground true class labels of the *PTrS*.

2.3 Final Classifier

As it is shown in Figure. 1, the steps 1, 2 and 3 are repeated *Maxiteration* times. In this method there is a procedure to select a set of satisfactory good cluster centers from several times of performing clustering techniques; however, the cluster centers obtained from any iteration can be considered as the solution. The method enhances both the accuracy and robustness of the KNN classifier algorithm, significantly; however, it needs less time and memory in testing phase. Based on empirical study, it can be induced that, usually the best results may be obtained when the *SCP* size is chosen near to the value of number of classes, c. Since each cluster center has d dimensions, examining each test sample needs to $O(cd)$. In the worst case the time complexity is $O(c^3d)$. It shows that the proposed combinational method can be employed with less order than the KNN classifier method which is $O(dkN)$.

	Dataset Name	*# of Class*	*# of Features*	*# of Samples*
1	Breast-Cancer*	2	9	683
2	Iris*	3	4	150
3	Bupa*	2	6	345
4	SAHeart*	2	9	462
5	Ionosphere	2	34	351
6	Glass*	6	9	214
7	Halfrings	2	2	400
8	Galaxy*	7	4	323
9	Yeast*	10	8	1484
10	Wine	3	13	178

Table 1. Brief information about the used datasets

3. Experimental Study

This section discuses the experimental results and compares the performance of the NCC algorithm with original KNN methods.

3.1 Datasets

The proposed method is examined over 9 different standard datasets and one artificial dataset. It is tried for the used datasets to be diverse in their number of true classes, features and samples. A large variety in used datasets can more validate the obtained results. Brief information about the used datasets is available in Table 1. More information is available in [14]

Note that datasets which are marked with star (*) in paper are normalized. The experiments are done over the normalized features in the stared dataset. It means each feature is normalized with the mean of 0 and the variance of 1, N(0, 1). The artificial HalfRing

dataset is depicted in Figure 2. The HalfRing dataset is considered as one of the most challenging dataset for the proposed NCC algorithms.

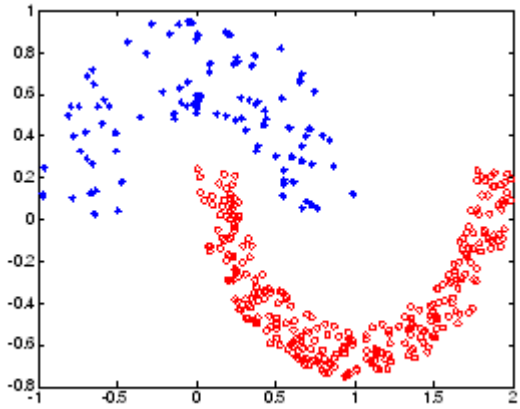

Figure 2. Half Ring dataset

3.2 Experimental Settings

All experiments are reported on 10-fold cross validation. Parameter *pq* is set to 5 through all experimentations. In all experiments Parameter *Maxiteration* is 10. Parameter *clus_alg* is k-means. It means that the k-means clustering algorithm is considered as clustering algorithm. The maximum allowed iterations in the k-means clustering algorithm is equal to 2 in order to obtain the rough and unlike partitions out of data. The number of partitions that is requested from k-means is a random value between *c* and 2**c*. Validation set, *VS*, is 22.22% of train set, *PTrS*, through all experimentations.

3.3 Experimental Results

Table 2 shows final accuracies of the NCC algorithm using three different conditions. In each column, the accuracies obtained by the NCC algorithm employing one condition are shown. Next to the accuracy of each NCC algorithm over each dataset, the averaged number of cluster centers in the final 1-NN classifier is presented. It is obvious that the condition $cvp \geq \frac{pq}{2}$ is the best condition among the three used conditions. But there is a hidden rule among the results obtained by employing the condition $cvp \geq \frac{pq}{2}$ in the NCC algorithm and the results obtained by employing the condition $cvp \geq pq$ in the NCC algorithm. First we define a new column in Table 2. It is the ratio of column 3 to column 5. We present this defined column in last column of Table 2. By a detailed considering of the values in the last column of Table 2, it is inferred when the last column for a dataset is lower than the averaged last column over all datasets (depicted at last column and last row in Table 2, it is equal to 1.58), the NCC algorithm with the condition $cvp \geq \frac{pq}{2}$ is superior to the others; otherwise the NCC algorithm with the condition $cvp \geq \frac{pq}{2}$ is superior to the others.

The hidden rule says when the number of prototypes in the NCC with the condition $cvp \geq pq$ is less than the number of prototypes in the NCC with the condition $cvp \geq \frac{pq}{2}$ by large margin, the dataset is a hard one. So we must turn to the NCC with the best prototypes. It means that the NCC with the condition is the best option. It also says when the number of prototypes in the NCC with the condition $cvp \geq pq$ is less than the number of prototypes in the NCC with the condition by small margin, the dataset is an easy one. So we can turn to the NCC with the more prototypes to cover total feature space. It means that the NCC with the condition is the $cvp \geq \frac{pq}{2}$ best option.

By using the hidden rule, we can present a combinatorial selective classifier that contains both NCCs and uses each of them dependent on the defined ratio for the two classifiers.

Table 3 shows the performances of different KNN classification and the proposed combinatorial selective classifier comparatively. NCC is compared with original versions of KNN. The NCC method outperforms some of KNN classifiers in terms of average accuracy. In addition, because of the lower number of stored prototypes, the results of the proposed combinatorial selective classifier are gained while the testing phase of the NCC method has less computational burden in both cases of time and memory rather than the KNN classifiers.

	$cvp \geq \frac{pq}{2}$		$cvp \geq pq$		$cvp \leq \frac{pq}{2}$			
	NCC	Average # of prototype	NCC	Average # of prototype	NCC	Average # of prototype	All prototype	Ratio of column 3 to 5
1	**97.25**	6.90	96.2	6.20	96.25	9.00	280	1.11
2	95.33	8.60	**96.00**	7.30	**96.00**	7.00	105	1.18
3	96.47	6.20	**97.06**	5.10	95.88	5.00	479	1.22
4	**60.29**	10.40	52.53	2.50	58.24	10.00	242	4.16
5	**71.56**	52.00	67.50	28.90	69.69	141.00	227	1.80
6	**66.67**	28.90	52.86	15.00	65.71	98.00	151	1.93
7	82.86	9.90	**85.7**	6.90	84.00	8.00	246	1.43
8	**69.78**	10.10	65.65	1.60	68.26	9.00	324	6.31
9	95.29	9.20	95.29	7.40	**95.88**	8.00	126	1.24
10	**57.70**	92.40	57.53	6 7.31	54.53	480.00	1040	1.37
average	**79.32**	23.46	76.64	14.82	78.44	77.50	322	1.58

Table 2. Final results of the NCC algorithm using different conditions

	Different Method								
	1-NN	2-NN	3-NN	4-NN	5-NN	6-NN	7-NN	NCC	Average # of Prototypes
1	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	96.25	6.20
2	96.00	95.33	96.67	97.33	97.33	**98.00**	**98.00**	96.00	7.30
3	96.91	94.71	97.21	96.76	**97.50**	96.18	96.62	97.06	5.10
4	63.53	60.00	**63.82**	62.06	60.59	55.59	58.53	60.29	10.40
5	81.88	**82.81**	80.63	77.51	75.94	71.25	70.94	71.56	52.00
6	69.05	68.57	**69.52**	63.80	64.29	63.33	62.86	66.67	28.90
7	86.57	**89.43**	84.00	86.57	84.86	85.71	83.43	85.71	6.90
8	65.65	66.30	68.70	69.13	65.43	67.17	66.52	**69.78**	10.10
9	95.29	94.12	94.71	94.71	**96.47**	94.71	96.47	95.29	7.40
10	52.77	51.55	55.47	55.61	**57.91**	57.57	57.03	57.53	67.31
average	80.77	80.28	**81.07**	80.35	80.03	78.95	79.04	79.61	20.16

Table 3. Final accuracies of the NCC comparing with the results of different KNN classifiers

4. Conclusion and Future Works

In this paper, a new method is proposed to improve the performance of KNN classifier. The proposed method which is called NCC, standing for Nearest Cluster Classifier, improves the KNN classifier in terms of both time and memory order. The NCC algorithm employs clustering technique to find the same groups of data in multi-dimensional feature space.

Despite of reducing training prototypes, the clustering technique can cause to find the natural groups of data. On the other

hands, the natural neighborhoods can be successfully recognized by clustering technique. Moreover, unlike the KNN method which classifies any sample without considering the data distribution, only based on exactly *K* nearest neighbor, in the NCC algorithm, the data is grouped into *k* clusters unequally, according to the data distribution and the position of data samples in feature space.

The NCC method is examined over nine benchmarks from UCI repository and one hand made dataset, HalfRing. Regarding to the obtained results, it can be concluded that the proposed algorithm is comparatively not worse than the KNN classifier. The NCC method is even more accurate than the KNN classifier in some cases.

References

[1] Fix, E., Hodges, J. L. (1951). Discriminatory analysis, nonparametric discrimination: Consistency properties. Technical Report 4, USAF School of Aviation Medicine, Randolph Field, Texas.
[2] Cover, T. M., Hart, P. E. (1967). Nearest neighbor pattern classification, *IEEE Trans. Inform. Theory,* IT-13, 21–27.
[3] Hellman, M. E. (1970). The nearest neighbor classification rule with a reject option, *IEEE Trans. Syst. Man Cybern.*, 3, 179–185.
[4] Fukunaga, K., Hostetler, L. (1975). k-nearest-neighbor bayes-risk estimation, I*EEE Trans. Information Theory,* 21(3) 285-293.
[5] Dudani, S. A. (1976). The distance-weighted k-nearest-neighbor rule. *IEEE Trans. Syst. Man Cybern.*, SMC-6. 325–327.
[6] Bailey, T., Jain, A. (1978). A note on distance-weighted k-nearest neighbor rules. I*EEE Trans. Systems, Man, Cybernetics*, 1 (8) 311-313.
[7] Bermejo, S., Cabestany, J. (2000). Adaptive soft k-nearest-neighbour classifiers. *Pattern Recognition*, 33.
[8] Jozwik, A. (1983). A learning scheme for a fuzzy k-nn rule. *Pattern Recognition Letters*, 1,287–289.
[9] Keller, J. M., Gray, M. R., Givens, J. A. (1985). A fuzzy k-nn neighbor algorithm. *IEEE Trans. Syst. Man Cybern*., SMC-15(4) 580–585.
[10] Duda, R. O., Hart, P. E., Stork, D.G. (2000). Pattern Classification, John Wiley & Sons.
[11] ITQON, Shunichi, K., Satoru, I. (2001). Improving Performance of k-Nearest Neighbor Classifier by Test Features, Springer Transactions of the Institute of Electronics, Information and Communication Engineers.
[12] Lam, L., Suen, C. Y. (1997). Application of majority voting to pattern recognition: An analysis of its behavior and performance, *IEEE Transactions on Systems, Man, and Cybernetics*, 27(5) 553–568.
[13] Jain A.,K.,Dubes, R. C. (1988). Algorithms for Clustering Data. Englewood Cliffs, NJ: Prentice-Hall.
[14] Newman, C. B. D. J., Hettich, S., Merz, C. (1998). UCI repository of machine learning databases, http://www.ics.uci.edu/~mlearn/MLSummary.html.
[15] Wu, X. (2007). Top 10 algorithms in data mining, Knowledge information Springer-Verlag London Limited: 22-24.