A New Method of Learning Weighted Similarity Function to Improve Predictions of Nearest Neighbor Rule

M. Zolghadri Jahromi, E. Parvinnia

***Abstract*— The performance of Nearest Neighbor (NN) classifier is highly dependant on the distance (or similarity) function used to find the NN of an input test pattern. In order to optimize the accuracy of the NN rule, a weighted similarity function is proposed. In this scheme, a weight is assigned to each training instance. The weights of training instances are used in the generalization phase to find the NN of an input test pattern. To specify the weights of training instances, we propose a learning algorithm that attempts to minimize the leave-one-out (LV1) error rate of the classifier on train data. The proposed approach is assessed using a number of data sets from UCI corpora. Simulation results show that the proposed method improves the generalization accuracy of the basic NN and results are comparable to or better than other methods proposed in the past to learn the distance function.**

***Index Terms*— nearest neighbor, weighted metrics, adaptive distance measure.**

I. Introduction

The NN rule is one of the oldest and simplest methods of non-parametric pattern classification. The basic rationale for NN rule is both simple and intuitive: patterns close in feature space are likely to belong to the same class. The NN classifier can be represented by the following simple rule: to classify an unknown pattern, choose the class of the nearest stored training instance. A common extension is to choose the most common class among the K Nearest Neighbors (KNN).

The performance of the NN rule depends crucially on how to choose a suitable distance metric. In the past, many methods have been developed to locally adapt the distance metric. Examples of these include the flexible metric method proposed by Friedman [1], the discriminant adaptive method by Hasti and Tibshirani [2], and the adaptive metric method by Domeniconi et al. [3].

The common idea underlying above methods is that they estimate feature relevance locally at each query pattern. This leads to a weighted metric for computing the similarity between the query patterns and training data. For example, in [4], an adaptive K-NN classification algorithm is proposed that is based on the concept of statistical confidence from hypotheses testing.

In [5], similar to our proposed method, a simple weighted distance measure is proposed that uses a heuristic measure to specify the weight of each training instance.

In [6], a scheme is proposed to learn weighted metrics to improve generalization accuracy of NN. The weights may be specific for each class and feature, for each individual instance, or for both. The learning algorithms derived by approximately minimizing the LV1 classification error rate of the given training set.

Another work in this area is the scheme presented in [7] to learn the distance function for NN rule. The distance function is learned based on maximizing the clustering of objects belonging to the same class. The learned distance function is used to cluster the objects of a data set. Using the learned distance function, a test instance is classified using NN rule (LW1NN)or Nearest Cluster Centroid (NCC).

The weighted metric we use in this paper is based on assigning a weight to each instance in the training set. Our proposed learning algorithm attempts to specify the weight of each training instance such that LV1 classification error of the given training set is minimized. The proposed learning algorithm is basically a hill climbing search algorithm.

The rest of this paper is organized as follows. In section 2, NN classification with weighted similarity metric is described. In section 3, our proposed method of learning the weights of training instances is presented. In section 4, simulation results are given. Section 5 concludes this paper.

II. NN Classification with Weighted Similarity Metric

We briefly describe the NN rule to introduce the notation. For an M-class problem, assume that a set of training examples of the form $\{(X_i, C_i) \mid i = 1, ..., n\}$ is given. Where, X_i is a d-dimensional vector of attributes $X_i = [x_{i1}, x_{i2}, ..., x_{id}]^T$ and $C_i \in [1,2, \ldots, M]$ defines the corresponding class label. To identify the NN of a query pattern, a distance function has to be defined to measure the distance between two patterns. A variety of distance functions has been used for this purpose [8]. Euclidean distance has conventionally been used to measure the distance (i.e., dissimilarity) between two patterns X_i and X_j:

Dr. M. Zolghadri Jahromi is with the Departement of Computer Science and Engineering, Shiraz University, Shiraz, Iran (corresponding author), phone: 0098-711-6271747; fax: 0098-711-6271747;
e-mail: zjahromi@shirazu.ac.ir.
E. Parvinnia, Ph.d. student, Departement of Computer Science and Engineering, Shiraz University, Shiraz, Iran (e-mail: parvinn@shirazu.ac.ir).

$$d(X_i, X_j) = \sqrt{\sum_{k=1}^{d} (X_{ik} - X_{jk})^2} \quad (1)$$

This distance function is appropriate when all the input attributes are numeric and have ranges of approximately equal length. When attribute have substantially different range lengths, the attributes can be normalized by dividing the individual attribute distances by the range or standard deviation of the attributes.

Instead of working with distance function, we can equivalently work with the following similarity measure, which normalizes the similarity between two instances X_i and X_j by a real number in the interval [0,1].

$$\mu(X_i, X_j) = 1 - \frac{d(X_i, X_j)}{d_{\max}} \quad (2)$$

In above relation, $d_{\max}$ is the maximum distance that a pair of instances can have. This is calculated by considering two virtual instances having maximum difference in each attribute value. That is:

$$d_{\max} = \sqrt{\sum_{i=1}^{n} (\Delta_i)^2} \quad (3)$$

Where, Δ_i represents the difference between maximum and minimum values of attribute i. Using (2), the most similar pattern X_p to a query pattern Q can be formally stated as:

$$P = \arg\max_{1 \le j \le n} \left\{ \mu(Q, X_j) \right\} \quad (4)$$

The NN rule assumes that all classifiers (i.e., stored instances) are equally reliable and uses equation (4) to find the NN of a query pattern. This paper is based on the idea that some of the stored instances are more reliable classifiers than others. We accomplish this by assigning a weight w_k to each stored instance X_k. The weights of the stored instances are used in the test phase to find the most similar pattern X_w to a query pattern Q.

$$W = \arg\max_{1 \le j \le n} \left\{ w_j . \mu(Q, X_j) \right\} \quad (5)$$

III. Learning Weights of Training Instances Math

In this section, we present a greedy algorithm that attempts to maximize LV1 classification rate on the given training set (using NN classifier) by specifying the weights of training instances. For an M-class problem, assume that N labeled instances $\{X_i, i=1, 2, \ldots, n\}$ from different classes are available.

Note that, assigning a large weight to a training instance increases its influence and can be used to classify many instances in LV1 test. In contrast, setting the weight of a training instance to zero will reduce the similarity of that instance with any test instance to zero (i.e., appears to be far away from any input test pattern).

The basic component of our proposed learning scheme is an algorithm that can specify the optimal weight of a training instance assuming that the weights of all other instances are given and fixed. The weight of an instance specified using this algorithm is optimal in the sense that it maximizes the LV1 classification rate (under the mentioned conditions). Starting with an initial weight vector (i.e., $\{w_i = 1.0, i=1,2,\ldots,n\}$), the overall learning scheme is to use this algorithm to adjust the weight of each training instance in turn. After a single pass over all instances, it is still possible to improve LV1 classification rate by a new pass over the data. The search for globally optimum solution can be stopped after a certain number of passes or when no improvement was observed during the last pass over the training set. Note that, the learning scheme is a greedy optimization technique since LV1 classification rate never decreases during this learning scheme.

Without loss of generality, in the following, we present the algorithm that specifies the optimal weight w_k (a number in the interval $[0, \infty]$) of a training instance $X_k \in$ Class T. To find the optimal value of w_k, in the first step, w_k is set to zero. This effectively removes the influence of X_k and the instance is not used to classify any test instance in LV1 test. Using w_k=0., instances from class T that are classified correctly in LV1 test are marked. Note that, these instances will be classified correctly regardless of the value w_k. Instances of $\overline{Class\,T}$ that are misclassified are also marked. These instances will be misclassified regardless of the value of w_k. At this stage, the true classification of unmarked instances depends on the value of w_k. The score S of any unmarked instance (i.e., X_t) in the training set is calculated using the following definition:

$$S(X_t) = \frac{\max_{1 \le j \le n} \left\{ w_j . \mu(X_t, X_j),\ j \ne t \right\}}{\mu(X_t, X_k)} \quad (6)$$

The interesting property of the score of a pattern (i.e., $S(X_t)$) is that using any value $w_k > S(X_t)$, pattern X_t will be classified as class T in LV1 test. This is due to the fact that using any value of the $w_k > S(X_t)$, the following relation can be easily derived from (6).

$$w_k . \mu(X_t, X_k) > \max_{1 \le j \le n} \left\{ w_j . \mu(X_t, X_j),\ j \ne t \right\} \quad (7)$$

This in turn means that using any value $w_k > S(X_t)$, pattern X_k will be the nearest neighbor of X_t. To find the optimal value of w_k, unmarked instances are ranked in ascending order of their scores. Assuming that L instances $\{X_1, X_2, \ldots X_L\}$ are in the list, choosing any value of w_k between two successive scores (i.e., $S(X_P) < w_k < S(X_{P+1})$), all instances that their scores are smaller than w_k will be classified as class T. To find optimal value of w_k, for a list of L instances, L+1 thresholds must be examined. The corresponding LV1 accuracy for any chosen value of w_k can be easily calculated as we know the true class of each instance. The best value of w_k is the one maximizing LV1 classification rate. In our implementation, we choose the thresholds in the middle of two successive scores. The above steps for finding optimal weight of an instance are summarized in Table I.

IV. 4. Simulation Results

We used six data sets available from the UCI repository to evaluate the prediction accuracy of the proposed scheme. All these data sets only contain numerical attributes. Table II gives a short summary of the data sets we used in the experiments.

Table I. Algorithm for finding optimal weight w_k of training instance $X_k \in$ Class T

1. Set $w_k = 0$.
2. Mark instances of Class T that are classified correctly.
3. Mark instances of $\overline{ClassT}$ that are misclassified.
4. Use (6) to rank unmarked training instances in ascending order of their scores.
5. Find the best vale of w_k resulting in maximum LV1 classification rate.

Table II. Statistics of the data sets used in experiments

Data Sets	# of Instances	# of Features	# of Classes
Diabet	768	8	2
Glass	214	9	6
Heart_h	294	13	5
Heart_s	270	13	2
Ionosphere	351	34	2
Vehicle	846	18	4

Attributes of each data set were first normalized to the interval [0,1]. In data sets having missing values (i.e., Heart-h), missing values were replaced by the average value of that attribute in the data set. Using Euclidean distance function for a normalized data set having n attributes, the maximum possible distance between two instances is . The similarity of each pair of instances in the training set was then calculated using (2).

Ten-fold cross validation (10-CV) was used to evaluate the scheme proposed in this paper. In this, the data set is divided into ten disjoint groups of approximately equal size. Using nine partitions (i.e., using 90% of data) as training data, the proposed learning method was used to specify the weights of training instances. Ten passes over training data was used to specify the weights. Using the weighted training instances, classification accuracy on the test partition was measured. This process was repeated until all partitions were uses in the test phase. The 10-CV test was repeated ten times and average classification rate on test data was calculated as the performance of the scheme.

In Fig. 1, we have plotted the LV1 error-rate of the classifier during the learning process for one fold in 10-CV experiment. Using nine portions of the Diabetes data, Fig. 1 shows the LV1 error-rate as the weight of each instance is updated using the proposed method. The initial LV1 error-rate was 19.41% at the beginning of the run. After 10 passes of the proposed method, LV1 error-rate was reduced to 6.85%, which proves the effectiveness of the scheme. It can also be seen that LV1 error-rate never increases during this process.

Table III gives the 10-CV classification rate on various data sets. For comparison, we have also reported the classification rate of basic NN. As seen, the proposed method could improve the generalization accuracy of the basic NN classifier for 4 out of 6 data sets. All improvements are

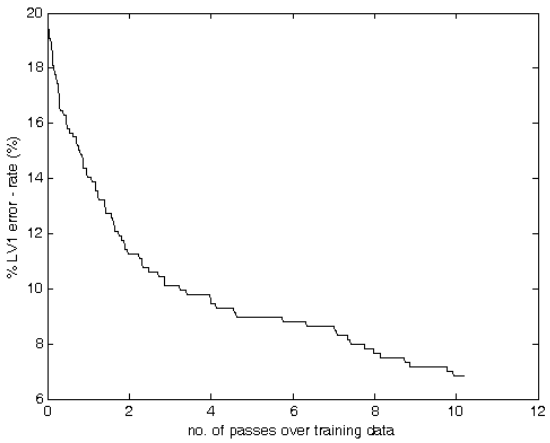

Fig. 1. LV1 classification error rate during the weight learning algorithm

Table III. 10-CV classification rate on various data sets

Data sets	1NN	Proposed
Diabetics	70.36	**75.13**
Glass	69.38	67.62
Heart_h	78.31	**80.69**
Heart_s	75.41	**81.11**
Ionosphere	86.49	**92.86**
Vehicle	69.52	68.10

Statistical significance was determined by a paired t-test on the accuracy of the 10 runs of 10 fold cross validation. The improvement in accuracy is more than 3.3% for these data sets. In case of Glass and Vehicle data sets, our proposed scheme has caused a small drop in classification rate.

In Table IV, we compare the performance of our proposed method in comparison with the methods (i.e., LW1NN and NCC) presented in [7] that uses clustering to learn the distance function for NN rule. The performance of C4.5 decision tree algorithm that was run with its default parameter settings is also reported for these data sets. The best result on each data set is indicated in bold. As seen, the performance of our proposed method is comparable or better than these algorithms.

Table IV. 10-CV classification rate of various methods

Data sets	Proposed	NCC	LW1NN	C4.5
Diabetics	**75.13**	73.07	68.89	74.49
Glass	67.62	66.41	**73.50**	67.71
Heart_h	80.69	**81.54**	77.55	80.22
Heart_s	81.11	**81.70**	77.52	78.15
Ionosphere	**92.86**	86.73	91.73	89.74
Vehicle	68.17	65.94	69.86	**72.28**

V. Conclusions

In this paper, we introduced a new technique for adapting the distance function in NN classification method. This was achieved by assigning a weight to each training instance. We proposed a greedy algorithm for learning the weights of

training instances. The learning algorithm attempts to maximize LV1 classification rate, which is the true estimate of the generalization ability of the classifier. Simulation results proved that the scheme could improve the generalization ability of basic NN rule.

Although not addressed in this paper, the proposed scheme can be easily extended to improve generalization speed of NN rule by reducing the number of stored instances. This is very important case of high dimensional problems having large number of training data.

REFERENCES

[1] J. H. Friedman, Technical Report 113, Stanford University Statistics Department, Flexible metric nearest-neighbor classification, 1994.

[2] T. Hastie, R. Tibshirani, "Discriminant adaptive nearest neighbor classification", *IEEE Transaction on Pattern Analysis and Machine Intelligence*, vol. 18 no. 6, 1996, pp. 607-615.

[3] C. Domeniconi, J. Peng, and D. Gunopulos , "Locally adaptive metric nearest neighbor classification", *IEEE Transaction on Pattern Analysis and Machine Intelligence* vol. 24, 2002, pp. 1281-1285.

[4] J. Wang, P. Neskovic, and L. N. Cooper, "Neighborhood selection in the k-nearest neighbor rule using statistical confidence", *Pattern Recognition*, vol. 39, 2006, pp. 417–423.

[5] J. Wang, P. Neskovic, and L. N. Cooper, "Improving nearest neighbor rule with a simple adaptive distance measure", *Pattern Recognition Letters*, vol. 28, 2007, pp. 207-213.

[6] R. Paredes, and E. vidal, "Learning weighted metrics to minimize nearest-neighbor classification error", *IEEE Transaction on Pattern Analysis and Machine Intelligece*, vol. 28 no. 7, July 2006, pp. 1100-1110.

[7] C. F. Eick, A. Rouhana, A. Bagherjeiran, and R. Vilalta, "Using clustering to learning distance functions for supervised similarity assessment" *Engineering Applications of Artificial Intelligence*, Vol.19, no. 4, June 2006, pp. 395-401.

[8] D. R.Wilson, T. R. Martinez, "Reduction Techniques for Exemplar-Based Learning Algorithms", *Machine Learning*, vol. 38 no. 3, 2000, pp. 257-286.