Combining Multiple Classifiers with Dynamic Weighted Voting

R.M. Valdovinos[1] and J.S. Sánchez[2]

[1] Lab. Reconocimiento de Patrones, Instituto Tecnológico de Toluca
Av. Tecnológico s/n, 52140 Metepec (Mexico)
E-mail: li_rmvr@hotmail.com
[2] Dept. Llenguatges i Sistemes Informàtics, Universitat Jaume I
Av. Sos Baynat s/n, E-12071 Castelló de la Plana (Spain)
E-mail: sanchez@uji.es

Abstract. When a multiple classifier system is employed, one of the most popular methods to accomplish the classifier fusion is the simple majority voting. However, when the performance of the ensemble members is not uniform, the efficiency of this type of voting generally results affected negatively. In the present paper, new functions for dynamic weighting in classifier fusion are introduced. Experimental results with several real-problem data sets from the UCI Machine Learning Database Repository demonstrate the advantages of these novel weighting strategies over the simple voting scheme.

1 Introduction

A multiple classifier system (MCS) is a set of individual classifiers whose decisions are combined when classifying new patterns. There are many different reasons for combining multiple classifiers to solve a given learning problem [6]. First, MCSs try to exploit the local different behavior of the individual classifiers to improve the accuracy of the overall system. Second, in some cases MCS might not be better than the single best classifier but can diminish or eliminate the risk of picking an inadequate single classifier. Another reason for using MCS arises from the limited representational capability of learning algorithms. It is possible that the classifier space considered for the problem does not contain the optimal classifier.

Let $\mathcal{D} = \{D_1, \ldots, D_L\}$ be a set of L classifiers. Each classifier D_i $(i = 1, \ldots, L)$ assigns an input feature vector $\mathbf{x} \in \Re^n$ to one of the possible C problem classes. The output of a MCS is an L-dimensional vector $[D_1(\mathbf{x}), \ldots, D_L(\mathbf{x})]^{\mathrm{T}}$ containing the decisions of each of the L individual classifiers.

In the literature, there are two main strategies for combining classifiers: *selection* and *fusion*. In classifier selection, each individual classifier is supposed to be an expert in a part of the feature space and correspondingly, we select only one classifier to label the input vector $\mathbf{x} \in \Re^n$. On the other hand, in classifier fusion, each component is supposed to have knowledge of the whole feature space and thus, all individual classifiers are taken into account to decide the label of the input vector.

Focusing on the fusion strategy, the combination can be made in many different ways. The simplest one employs the majority rule in a plain voting system [14]. More

elaborated schemes use weighted voting rules, in which each individual component can be associated with a different weight [17]. Then, the final decision can be made by majority, average [12], minority, medium [4], product of votes, or using some other more complex methods [13].

In the present work, several new methods for weighting the individual components of a MCS are proposed. Then the effectiveness of the new approaches is empirically tested over a number of real-problem data sets. All these methods correspond to the so-called dynamic weighting and basically consist of using the distances to the input pattern $\mathbf{x}$ in each individual classifier.

From now on, the paper is organized as follows. Section 2 discusses the classifier fusion technique for combining classifiers and remarks the advantages of using some weighting scheme. In Sect. 3, the different weight functions are introduced. Section 4 provides the experimental results. Finally, the main conclusions of this work and possible lines for future research are commented in Sect. 5.

2 Classifier Fusion

Classifier fusion assumes that all classifiers in the set $\mathcal{D}$ are competitive, instead of complementary. For this reason, each component takes part in the decision of classifying an input vector $\mathbf{x}$. In the simple voting (by majority), the final decision is taken according to the number of votes given by the individual classifiers to each of the C classes, thus assigning $\mathbf{x}$ to the class that has obtained a majority of votes. When working with data sets that contain more than two classes, in the final decision ties among some classes are very frequently obtained. To solve this problem, several criteria can be considered. For instance, to randomly take the decision, or to implement an additional classifier whose ultimate goal is to bias the decision towards a certain class [11].

An important issue that has strongly called the attention of many researchers is the error rate associated to the simple voting method and to the individual components of a MCS. Hansen and Salomon [9] show that if each of the L classifiers being combined has an error rate less than 50%, it may be expected that the accuracy of the ensemble improves when more components are added to the system. However, this assumption not always is fulfilled. For instance, Matan [15] asserts that in some cases, the simple voting might perform even worse than any of the members of the MCS. Thus, the employment of some weighting method has been proposed as a way to partially overcome these difficulties.

A weighted voting method has the potential to make the MCS more robust to the choice of the number of individual classifiers. Two general approaches to weighting can be remarked: *dynamic weighting* and *static weighting* of classifiers. In the dynamic strategy, the weights assigned to the individual classifiers of the MCS can change for each input vector in the operational phase. In the static approach, the weights are computed for each classifier in the training phase, and they do not change during the classification of the input patterns.

3 Dynamic Weighted Voting Schemes

Several weighting functions are introduced in this section. Some of them are taken from the Pattern Recognition literature and are here conveniently adapted for classifier fusion in a MCS, while others are now proposed for the first time.

A voting rule for the k-NN rule [5] in which the votes of different neighbors are weighted by a function of their distance to the input pattern was first proposed by Dudani [7]. A neighbor with smaller distance is weighted more heavily than one with a greater distance: the nearest neighbor gets a weight of 1, the furthest neighbor a weight of 0, and the other weights are scaled linearly to the interval in between. From this, the Dudani's weight can be computed as:

$$w_j = \begin{cases} \frac{d_k - d_j}{d_k - d_1} & \text{if} \quad d_k \neq d_1 \\ 1 & \text{if} \quad d_k = d_1 \end{cases} \tag{1}$$

where d_j denotes the distance of the j'th nearest neighbor, d_1 is the distance of the nearest neighbor, and d_k indicates the distance of the furthest (k'th) neighbor.

In order to utilize this weighting function in the classifier fusion, the value of k (i.e., the number of neighbors in Dudani's rule) can be here replaced by the number of classifiers L that constitute the MCS. Moreover, the L distances of $\mathbf{x}$ to its nearest neighbor in each individual classifier have to be sorted in increasing order $(d_1, d_2, \ldots, d_L)$. Thus, the original Dudani's weight (Eq. 1) can be now rewritten as follows:

$$w(D_j) = \begin{cases} \frac{d_L - d_j}{d_L - d_1} & \text{if} \quad d_L \neq d_1 \\ 1 & \text{if} \quad d_L = d_1 \end{cases} \tag{2}$$

where d_1 denotes the shortest of the L distances of $\mathbf{x}$ to the nearest neighbor, and correspondingly d_L is the longest of those distances.

Dudani further proposed the *inverse distance weight* [7], which can be expressed as follows:

$$w(D_j) = \frac{1}{d_j} \quad \text{if} \quad d_j \neq 0 \tag{3}$$

Another weighting function proposed here is based on the work of Shepard [16], who argues for a universal perceptual law which states that the relevance of a previous stimulus for the generalization to a new stimulus is an exponentially decreasing function of its distance in psychological space. This gives the weighted voting function of Eq. 4, where α and β are constants and determine the slope and the power of the exponential decay function.

$$w(D_j) = e^{-\alpha d_j^{\beta}} \tag{4}$$

A modification to Shepard's weight function consists of using a different value of α for each input pattern. Firstly, the L distances of $\mathbf{x}$ to its nearest neighbor in each individual classifier have to be sorted in decreasing order. Then, the value of α for each input pattern is computed according to $\alpha = L - j + 1$. By this, the higher the distance

given by a classifier, the higher the value of α and thereby, the lower the weight assigned to such a classifier.

Finally, we propose another weighting function, which corresponds to the *average distance weight*. In summary, the aim of this new dynamic weighting procedure is to reward (by assigning the highest weight) the individual classifier with the nearest neighbor to the input pattern.

$$w(D_j) = \frac{\sum_{i=1}^{L} d_i}{d_j} \tag{5}$$

The rationale behind this weight is that the classifier with the nearest neighbor to $\mathbf{x}$ probably corresponds to that with the highest accuracy in the classification of the given input pattern.

4 Experiments and Results

The results here reported correspond to the experiments over ten real-problem data sets taken from the UCI Machine Learning Database Repository (`http://www.ics.uci.edu/~mlearn`). For each data set, the 5-fold cross-validation method was employed to estimate the classification error: 80% of the available patterns were used for training purposes and 20% for the test set. On the other hand, it has to be noted that in the present work, all the base classifiers correspond to the 1-NN (Nearest Neighbor) rule [5].

The experiments basically consist of computing the classification accuracy when using different voting schemes in a MCS. The weight functions proposed in the present paper (the average distance weight, the Shepard's and modified Shepard's functions, the inverse distance weight, and Dudani's weight) are compared to the simple majority voting. In the experiments here carried out, we have set $\alpha = \beta = 1.0$ for the computation of the original Shepard's weight function (Eq. 4).

The MCSs have been integrated by using four well-known resampling methods: *random selection with no replacement* [1], *bagging* [2], *boosting* [8], and *Arc-x4* [3]. Only the result of the best technique on each database has been presented in Table 1. Analogously, for each database, related to the number of subsamples to induce the individual classifiers, that is, the number of classifiers in the system, we have experimented with 3, 5, 7, 9, and 15, and the best results have been finally included in Table 1. Moreover, the 1-NN classification accuracy for each original training set (i.e., with no combination) is also reported as the baseline classifier. Note that values in bold type indicate the highest accuracy for each database.

From the results given in Table 1, some comments can be drawn. First, it is clear that in all databases the employment of a MCS leads to better performance than the individual 1-NN classifier. Second, the application of some weight function generally outperforms the combination of classifiers by means of the simple majority voting. In fact, we can find some weighting scheme with higher (or equal) classification accuracy than that of the simple voting on 8 out of 10 databases. Even, in the case of Pima database, differences between the simple voting and the average distance weight are not significant (0.13%).

Table 1. Averaged accuracy of different voting procedures

	Individual 1-NN	Simple voting	Average distance	Shepard's function	Modified Shepard's	Inverse distance	Dudani's weight
Cancer	95.62	**96.35**	96.20	**96.35**	**96.35**	96.20	95.89
Heart	58.15	62.96	**64.81**	61.11	61.85	**64.81**	58.52
Liver	65.22	65.80	**66.09**	65.80	65.80	64.93	60.87
Pima	65.88	**72.81**	72.68	68.37	68.24	71.90	67.58
Iris	96.00	**98.00**	97.33	97.33	96.67	97.33	96.67
Vehicle	64.24	62.34	64.48	**65.56**	65.19	64.48	64.24
Wine	72.35	75.88	**77.65**	73.53	74.12	**77.65**	75.95
German	65.21	70.21	**70.81**	68.11	66.91	**70.81**	67.34
Phoneme	76.08	75.93	76.51	75.97	**76.56**	76.51	76.02
Waveform	77.96	83.20	83.20	83.06	78.20	**83.54**	83.22

When comparing the dynamic weighting schemes, one can observe that no technique is clearly the best. Nevertheless, the average distance and the inverse distance weights seem to generally behave better than any other weighting function. In fact, each one reaches the highest classification accuracy on 4 out of 10 databases. On the other hand, when these two methods do not obtain the best results, their classification accuracies are still very close to that of the winner. For instance, in the case of Phoneme domain, while the modified Shepard's function obtains the highest accuracy rate (76.56%), the classification performance of both the average distance and the inverse distance weights are 76.51%. Similarly, for the Cancer database, differences in accuracy with respect to the "best" weighting schemes (Shepard's and modified Shepard's functions) are not significant (only 0.15%).

5 Concluding Remarks

When a MCS is employed in a classifier fusion scenario, one has to implement some procedure for combining the individual decisions of the base classifiers. Although the plain majority voting rule constitutes a very appealing method due to its conceptual and implementational simplicity, its efficiency can become too poor when the performance of the ensemble members is not uniform. Under this practical situation, more complex voting techniques, mainly in the direction of assigning different weights to each base classifier, should be applied to derive the final decision of the MCS.

In this paper, new methods for dynamic weighting in the framework of MCS have been introduced. More specifically, several weighting functions present in the literature have been adapted to be used in a voting system for classifier fusion. In particular, we have explored the reformulated Dudani's distance, the inverse distance weight [7], the average distance weighting, and also the Shepard's function [16] and a modification to it (which is based on a rank of distances).

Experimental results with several real-problem data sets have shown the benefits of using some dynamic weighting strategies over the simple majority voting scheme. On the other hand, the average distance and the inverse distance weights have appeared to

be as the best weighting functions in terms of highest classification accuracy: each one has exhibited the best performance on 4 out of 10 databases. Results also corroborate that in general, a MCS clearly outperforms the individual 1-NN classifier.

Future work is primarily addressed to investigate other weighting functions applied to classifier fusion in a MCS. Within this context, the use of several well-known data complexity measures [10] could be of interest to conveniently adjust the classifier weights. On the other hand, the results reported in this paper should be viewed as a first step towards a more complete understanding of the behavior of the weighted voting procedures and consequently, it is still necessary to perform a more exhaustive analysis of the dynamic and static weighting strategies over a larger number of synthetic and real databases.

Acknowledgements

This work has been partially supported by the Spanish CICYT (Ministry of Science and Technology) under grant TIC2003-08496.

References

1. R. Barandela, R.M. Valdovinos, J.S. Sánchez: New applications of ensembles of classifiers, *Pattern Analysis and Applications*, 6:245–256, 2003.
2. L. Breiman: Bagging predictors, *Machine Learning*, 24:123–140, 1996.
3. L. Breiman: Arcing classifiers, *Annals of Statistics*, 26:801–823, 1998.
4. D. Chen, X. Cheng: An asymptotic analysis of some expert fusion methods, *Pattern Recognition Letters*, 22:901–904, 2001.
5. B.V. Dasarathy: *Nearest Neighbor Norms: NN Pattern Classification Techniques*. IEEE Computer Society Press, Los Alamos, CA, 1991.
6. G.T. Dietterich: Machine learning research: four current directions, *AI Magazine*, 18:97–136, 1997.
7. S.A. Dudani: The distance weighted k-nearest neighbor rule, *IEEE Trans. on Systems, Man and Cybernetics*, 6:325–327, 1976.
8. Y. Freund, R.E. Schapire: Experiments with a new boosting algorithm, In: *Proc. of the 13th Intl. Conference on Machine Learning*, 148–156, 1996.
9. L.K. Hansen, P. Salomon: Neural network ensembles, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 12:993–1001, 1990.
10. T.-K. Ho, M. Basu: Complexity measures of supervised classification problems, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 24:289–300, 2002.
11. M. Kubat, M. Cooperson, *Jr.*: Voting nearest neighbor subclassifiers, In: *Proc. of the 17th Intl. Conference on Machine Learning*, 503–510, 2000.
12. L.I. Kuncheva: Using measures of similarity and inclusion for multiple classifier fusion by decision templates, *Fuzzy Sets and Systems*, 122:401–407, 2001.
13. L.I. Kuncheva, J.C. Bezdek, R.P.W. Duin: Decision templates for multiple classifier fusion, *Pattern Recognition*, 34:299–314, 2001.
14. L.I. Kuncheva, R.K. Kountchev: Generating classifier outputs of fixed accuracy and diversity, *Pattern Recognition Letters*, 23:593–600, 2002.
15. O. Matan: On voting ensembles of classifiers, In: *Proc. of the 13th Natl. Conference on Artificial Intelligence*, 84–88, 1996.

16. R.N. Shepard: Toward a universal law of generalization for psychological science, *Science*, 237:1317–1323, 1987.
17. K. Woods, W.P. Kegelmeyer, *Jr.*, K. Bowyer: Combination of multiple classifiers using local accuracy estimates, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 19:405–410, 1997.