ELSEVIER

Pattern Recognition Letters 20 (1999) 1353–1359

Pattern Recognition Letters

www.elsevier.nl/locate/patrec

Feature-based decision aggregation in modular neural network classifiers

Nayer Wanas [*], Mohamed S. Kamel, Gasser Auda, Fakhreddine Karray

Pattern Analysis and Machine Intelligence Laboratory, Systems Design Engineering Department, University of Waterloo, Waterloo, Ont., Canada N2L-3G1

Abstract

In several modular neural network (MNN) architectures, the individual decisions at the module level have to be integrated together using a voting scheme. All these voting schemes use the outputs of the individual modules to produce a global output without inferring explicit information from the problem feature space. This makes the choice of the aggregation procedure very subjective. In this work, a new MNN architecture will be presented. This architecture integrates learning into the voting scheme. We will be focusing on making the decision fusion a more *dynamic* process. In this context, dynamic means the aggregation procedure which has the flexibility to adapt to changes in the input. This approach requires the aggregation procedure to gather information about the input to help better understand how to dynamically aggregate decisions. © 1999 Published by Elsevier Science B.V. All rights reserved.

Keywords: Classification; Classifier combination; Dynamic decision fusion; Modular neural networks

1. Introduction

Achieving the best possible classification rate has always been the quest of designing pattern recognition systems. Usually, different classification schemes are developed for the problem in hand and, using experimental assessment, the best classifier is chosen as the final solution to the problem. However, it has been observed that although one design may outperform the others, the patterns that are misclassified by the different classifiers are not necessarily the same (Kittler et al., 1998). This observation suggests that the use of multiple classifiers can complement the decision about the patterns under classification, hence, improves the reliability of the overall classification process. However, the issue of *efficiently* combing the individual decisions to provide an aggregated decision is a very crucial issue in making such systems worthwhile (Auda and Kamel, 1998a).

These observations motivated the relatively recent interest in combining classifiers. The idea is not to rely totally on one single decision making scheme. Instead, all the different classifiers are used in the final decision making by combining their different “opinions”. The main objective behind this is to increase the efficiency and accuracy of the aggregated decision. To increase the efficiency, we can adopt multistage combination rules whereby objects are classified by simple classifiers using a small set of simple features, in combination with a

[*] Corresponding author.
E-mail addresses: nwanas@pami.uwaterloo.ca (N. Wanas), mkamel@pami.uwaterloo.ca (M.S. Kamel), gasser@pami.uwaterloo.ca (G. Auda), karray@pami.uwaterloo.ca (F. Karray)

reject option. For the more difficult objects, more complex procedures, possibly based on different features, are used (El-Shishiny et al., 1989). Classifier combination strategies may reflect the local competence of individual experts or the training process may aim to encourage some experts to achieve local decision making superiority (Auda and Kamel, 1997).

Neural networks have been used extensively in the literature in problems of recognition. Different approaches and techniques were used to approach different domains. Recently, the interest in combining multiple neural network classifiers has been addressed in the literature (Kittler, 1994; Auda and Kamel, 1998a; Drucker et al., 1993; Cho and Kim, 1995; Battiti and Colla, 1994).

Modular neural networks (MNNs) are a new and growing trend in the design of neural-network-based classification systems. The basic idea of MNN is to reduce the classification task into a set of smaller sub-tasks that are easily manageable. Cooperative modular neural networks (CMNNs) are an interesting line of modular designs (Auda and Kamel, 1997). In this case, the modules are decoupled sub-networks that not only learn to accomplish their own task, but try collectively to determine the best module that can provide the correct answer. MNN, generally, proved to be more efficient than the non-modular alternatives.

In any MNN architecture, the individual decisions at the module level can be integrated using a voting scheme (Auda et al., 1995). The individual modules are modeled as multiple voters, electing one candidate in a single ballot election, assuming the availability of votes' preference and intensities. Some of the voting schemes presented in the literature are plurality voting, which is the most popular, Nash voting and Fuzzy voting (Rogova, 1994; Battiti and Colla, 1994). All these voting schemes start from the local outputs of the individual modules to produce a global output, without inferring explicit information from the problem feature space. This makes the choice of the aggregation procedure very subjective. Auda and Kamel (1998b) presented an architecture that integrates the voting scheme in the learning process to help improve the overall performance of the system. In this case, however, the voting scheme is still static and does not adapt with the learning process.

In this work, a new architecture will be presented. This architecture integrates learning into the voting scheme. The problem feature space is divided into two sub-spaces. The first sub-space is used as an input to the MNN structure used. The other sub-space is used to learn how to determine the best candidate from the different modules. Integrating this approach with a voting scheme will make the final decision-making phase more adaptable to the problem being addressed. The organization of this paper is as follows. Section 2 presents the basic idea of feature based decision aggregation. In Section 3 we will present the different experiments performed. Finally, we present the conclusion of this work in Section 4.

2. Feature based decision aggregation

Most of the existing aggregation schemes can be considered as post-combination of decisions. That is, the decision combining scheme considers each individual classifier as a black box. The cooperative modular neural network (CMNN) (Auda and Kamel, 1997) and the ensembles voting online (EVOL) (Auda and Kamel, 1998b) have presented techniques that make the individual classifiers more involved in the final decision making process. However, even in these setups, the aggregation schemes implemented are totally isolated from the problem being considered. This fact makes the aggregation procedure a predefined scheme, i.e., determined prior to the actual use of the overall system.

This work will focus on making the decision fusion a more *adaptive* process. Local classification decisions are combined in a way similar to the parallel suite in decision fusion models (Dasarthy, 1994). This approach requires the aggregation procedure to gather information about the input beyond what individual classifiers provide. The gathered information (i.e., the extracted additional features) is used to tune the aggregation procedure.

This strategy will automatically guide the modules during the development phase, to adapt

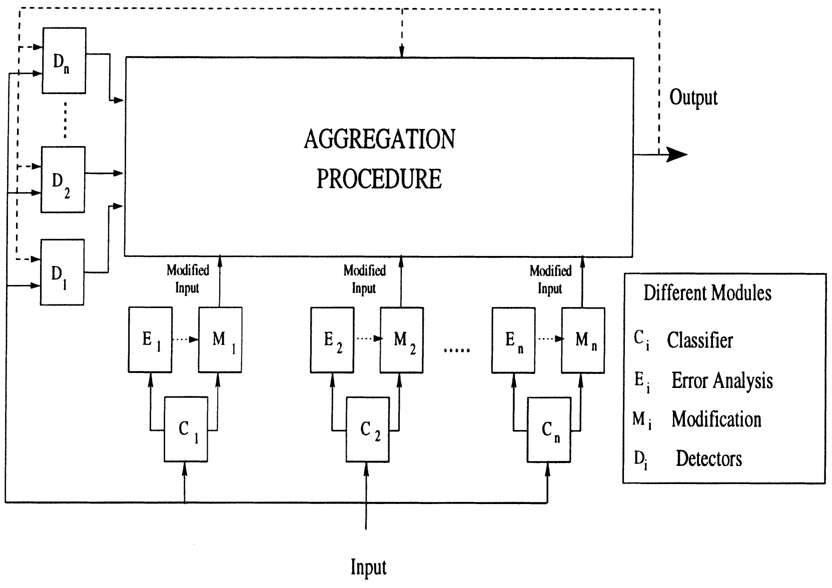

Fig. 1. Block diagram for the proposed architecture.

more to the learning samples that are not correctly classified by the local modules.

Fig. 1 shows the block diagram of a proposed architecture that incorporates these requirements. In the following subsections, we address each component of this architecture in some detail.

2.1. Classifiers

Each individual classifier, C_i, produces some output representing its interpretation of the input. In the context of this paper, we are more interested in using the output of these classifiers to help in the aggregation procedure, rather than the methodology of classification. Another goal is to utilize sub-optimal classifiers in the proposed architecture, to make the development overhead of such a system worthwhile. In this work, we use MNNs as classifiers, however, the scope of this work goes beyond MNNs.

2.2. Error analysis

These modules $\{E_i\}$ use the previously studied error analysis of the individual output and produce means to modify the output so as to overcome possible errors. These modules must be dynamic as well as comprehensive to adequately suit their purpose. Only relevant information is passed to the modifying module.

The accuracy of a pattern recognizer depends upon the intrinsic overlap of the class distributions and the estimation error due to the finite size of the set of learning objects. Errors can be classified into two categories: classification errors and systematic errors. The classification error, defined as the probability of error in classifying new objects, is used as a measure for the accuracy. The classification error depends on the characteristics of the features chosen, the number of features, the size of the learning set, and on the procedure used for the estimation of the discriminant function. A number of these items are influenced or determined by a priori knowledge. Systematic errors may be constant or may vary in a regular way (Topping, 1962). Eliminating those errors would help in achieving better performance. This is the main role of this module. The techniques and approaches used are totally dependent on the nature of the problem and the classification technique implemented in the classification module.

2.3. Modification modules

These modules $\{M_i\}$ use the information from the error analysis modules and operate on the classifier input to produce a modified output that is normalized to be within a common representation with all other inputs fed to the aggregation procedure. Hence, the comparison between the different inputs then is meaningful. Similar to the error analysis modules, these modules are dependent on the nature of the problem and the classifier used. In these modules, a confidence index is associated to each classifier. This index can be interpreted as the conditional probability that the classifier experienced success given a certain input, or $P(x_i \text{ correct} \mid \text{Input})$, where x_i is the output vector of classifier *i*. The confidence in the output can be interpreted in different ways. In this work, the difference between the two top output values of each classifier is used as the confidence in the output of that classifier.

2.4. Detectors

Each detector, D_i, takes the input space and tries to extract useful information for the aggregation procedure, rather than aiming to solve the classification problem. In other words, it tries to understand and collect information that might be helpful in the aggregation procedure. For instance, in a character recognition problem, the goal is to identify a given character. While the individual classifiers try to determine the character, the detectors try to identify the category of the character. This helps the aggregation scheme in determining how to combine the different classification outputs to achieve a better performance.

2.5. The aggregation procedure

The aggregation procedure represents the fusion layer of all the different outputs to generate a more competent output. The aggregation procedure uses detectors' outputs to guide the means of combining different classification results. The aggregation scheme can be divided into two phases: a learning phase and a decision making phase. The learning phase assigns a weight factor to each input to support each decision maker. This weighting factor represents a confidence in the output of each classifier. These confidences are then aggregated using standard classifier-combining methods.

A neural network approach is selected to perform the weighting of the individual classifier. The neural network would take the inputs from both individual classifiers and detectors and presents a newly modified probability of success of each classifier. Implementation details are given in Section 3.

3. Test problems and results

3.1. Data sets

3.1.1. Gaussian 20-class problem

This problem is basically a two dimensional recognition problem introduced by Auda and Kamel (1997), in which the samples are randomly generated into 20 different classes. Each class has a total number of 100 samples. The samples where randomly generated to follow a Gaussian distribution around a random class mean. The complexity, and in this case what makes the presented results more general, lies in the "non-homogeneous" regions in the feature space and the overlapping boundaries between them. Fig. 2 shows the distribution of the data set.

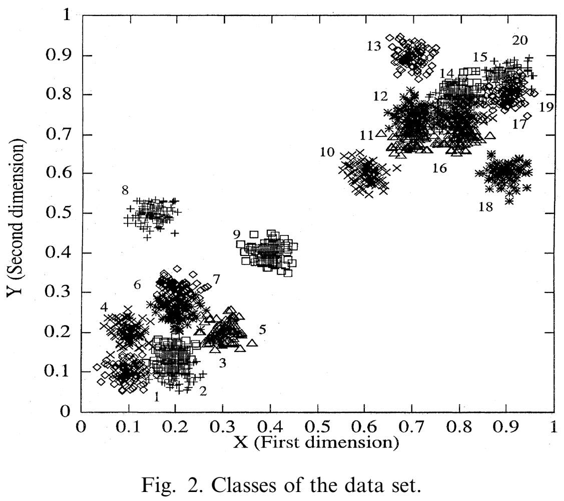

Fig. 2. Classes of the data set.

3.1.2. Arabic caps problem

Handwritten Arabic character recognition data is a reasonably complex classification problem due to the similarities among many groups of its characters (Drawish and Auda, 1994). This introduces different levels of overlap among the different characters in the input-space, and hence complicates the job of the classifier due to the resulting internal interference. Also, note that solving the complete Arabic handwritten recognition problem is not the main focus of this work since that has many other dimensions of complexity (Dasarthy, 1994).

The database used consists of 50 images for each character collected from 5 writers, each wrote the whole 28-character alphabet ten times. At first the characters are normalized with respect to scaling, translation and rotation, using the geometric moments of the image. Translation invariance is achieved by transforming the image into a new one whose two first-order moments are equal to zero (translate the centroid of the images to the origin). Scale invariance is achieved by enlarging each object such that its zero-order moment, i.e. the number of object pixels in the image, is set equal to some predetermined value. Rotation invariance is achieved by setting the "major axis angle" to zero.

A 46-element feature vector is calculated for every character. This means that the job of the classifier to ultimately map the characters to 28 regions in the 46-dimension feature space. The utilized features are simple topological attributes, yet they carry enough information to discriminate between the different characters.

3.2. Results

3.2.1. Twenty class problem

The individual classifiers in this case were six backpropagation neural networks. For consistency, the training procedure, though not the training data, as well as the neural network structure was the same for all the classifiers. The data were initially split into two sets, a training set and an evaluation set. The evaluation set was further split into a verification set and a testing set. The training was done separately on different subsets from the training set. Six different training subsets were used, each to train one classifier. Table 1 shows the different training subsets. The classifiers had 2 input, twelve hidden and 20 output nodes. The learning algorithm was the Extended DBD algorithm with momentum 0.4 and learning coefficient 0.5. The individual decision was the highest output of each classifier.

Table 1
Classes within each training subset

Subset	Classes included
1	1,2,3,4,5,6,7,8,9
2	8,9,10,11,12,13,14,15,16,17,18,19,20
3	1,2,4,5,6,7,8,9,10
4	2,4,7,10,11,12,13,14,15,16,17,18,19,20
5	Data from all classes, with emphasis from classes 1 to 7
6	Data from all classes, with emphasis from classes 10 to 20

The subsets were chosen due to the nature of the data distribution. Two detectors were used, both were self-organizing maps. One split down the clusters to twelve groups, which was the decomposition using the hierarchical task-decomposition technique (Auda, 1996). The other split down the clusters into three groups. The aggregation procedure was yet another backpropagation neural network, taking the confidence output of each classifier and the output of the detectors. The confidence was evaluated using the confusion matrix of each classifier. If classifier k produces an output j, the confidence C_k of that classifier can be given by

$$C_k = \frac{c^k_{jj}}{\sum_{i=1}^{N} c^k_{ij}}, \tag{1}$$

where c^k_{ij} represents the number of data belonging to class i whereas the classifier recognized them as being class j. These represent the diagonal elements in the confusion matrix generated from the training data for each classifier.

Table 2 shows the performance of the different techniques implemented. The performance was measured using different data sets. The verification

Table 2
Performance of the Gaussian 20-class problem

Data set	Verification (%)	Testing (%)	Group 1 (%)	Group 3 (%)
Best NN	70.33	72.29	78	67.82
Majority vote	78	79.86	87.23	75.27
Average vote	76	77.71	85.71	66.55
Feature based	81.67	82	86.57	82.55

and testing were the split up of the second half of the data set (700 entries for testing and 300 for verification). Groups 1, 2 and 3 were generated using data from clusters {1,2,3,4,5,6,7}, {8,9} and {10,11,12,13,14,15,16,17,18,19,20}, respectively.

All the different classifier combination schemes did succeed in recognizing data from classes 8 and 9 (group 2) perfectly; therefore it is not included in the table. It can be noted that the static classifier combination procedures did improve on the single classifier case; however, the feature based procedure proposed improved the performance further. Furthermore, it was consistent with all the data subsets in producing high classification results and did not suffer from high fluctuations in the reported output. This robustness is due to its adaptability to the input data through the detectors.

3.2.2. *Arabic caps problem*

The structure used in this problem was an ensemble MNN. Two classifiers, one a backpropagation and the other a learning vector quantizer. These classifiers were trained using the same set of data which represents half of the available data. The other half was divided between a verification and a testing set. A self-organizing map dividing the characters into groups was used in the detector blocks to assist the aggregation. The final aggregation procedure was a back-propagation neural network. The training procedure is performed by running the training data through all the individual classifiers and then using the outputs as the training data for the aggregating neural network.

The proposed architecture shows a higher performance than any of the individual neural networks (C_n); it successfully reduces the number of classification errors by at least 11%. Moreover, it also produces better results than other architectures proposed in the literature to solve this problem (Auda and Kamel, 1997). The results were also compared to a static combination scheme, namely, the maximum output criteria. The static combiner did not perform well due to the lack of consistency between both classifiers. The aggregation NN overcomes this problem. Table 3 shows the results of applying this approach to solving this classification problem. Due to the fact that only two individual classifiers were implemented, we used only the maximum vote. Other voting techniques may not be meaningful (e.g. majority vote, average vote, etc.) in this case.

Table 3
Comparison of recognition rate between the new approach and non-modular neural network for Arabic caps problem

Feature based	94.96%
Maximum vote	92.85%
Best NN	90.3%

4. Conclusion

A new architecture was proposed to allow for dynamic decision fusion of classifiers. In this architecture, the aggregation procedure has the flexibility to adapt to changes in the input and output in order to improve on the final output. The main idea behind this architecture is that it tries to understand changes in the input, by means of extracting features using the detectors, to direct the way it performs the aggregation. The aggregation learns how to combine the different decisions in order to improve the overall performance of classification. This approach also aims at reducing the cost and time of designing individual classifiers by allowing collaborate work. The empirical results were satisfactory. Both of the test problems showed improvement in the performance over static combiners. The architecture proposed here provides a robust and adaptive scheme for combining classifiers.

Discussion

Kuncheva: You propose a rather complex architecture. Where do you think does the credit go? Is the improvement due to the detectors, to modifiers of the classifier's output, or to the aggregation that you choose?

Kamel: Actually, we think it is due to all of those. We feel that the detectors have more credit than the others. Many of the existing schemes perform error analysis and modifications in terms of confidence levels. So, the credit here goes mainly to the detectors.

Ghosh: My question is a more specific version of the previous one. When you have an adaptive decision fusion strategy, as opposed to a fixed one like averaging or voting, it has been widely observed that such a system tends to over-train. And that is the reason, for example, why Wolpert has to use a complicated leave-one-out procedure for stacked generalisation. (*Note of the editors*: *see, e.g., D.H. Wolpert, Stacked Generalization, Neural Comp. 5, 241–159, 1992.*) But in your case, the validation and testing results are very close, which indicates no over-training. Do you have an explanation for this?

Kamel: I don't really have an explanation, now, but I would like to study this architecture and other combining schemes and how to fit them in the same architecture, to be able to make better comparisons between all of them. I am aware of some other work on combining classifiers, which are data-dependent as well. However, I have not seen a general architecture like the one proposed here.

Ghosh: Perhaps your detectors make the classifiers very decoupled. This makes it more like a mixture of experts, where you don't have this over-training problem.

Kamel: This is possible, however, it needs to be proven. Well, I feel this will keep me going for another five years, to fully study the properties of this architecture.

Acknowledgements

This work was supported by the Natural Science and Engineering Research Council of Canada (NSERC).

References

Auda, G., 1996. Cooperative modular neural network classifiers. Ph.D. thesis, University of Waterloo, Canada.

Auda, G., Kamel, M., 1997. CMNN: Cooperative Modular Neural Networks for pattern recognition. Pattern Recognition Letters 18, 1391–1398.

Auda, G., Kamel, M., 1998a. Modular neural network classifiers: a compartive study. Journal of Intelligent and Robotic Systems 21, 117–129.

Auda, G., Kamel, M., 1998b. EVOL: ensemble voting on-line. In: Proceedings of the 1998 International Conference on Neural Networks, pp. 1356–1360.

Auda, G., Kamel, M., Raafat, H., 1995. Voting schemes for cooperative neural network classifiers. In: Proceedings of the 1995 International Conference on Neural Networks, Perth, Australia, pp. 1240–1243.

Battiti, R., Colla, M., 1994. Democracy in neural nets: voting schemes for classification. Neural Networks 7 (4), 691–707.

Cho, S., Kim, J., 1995. Combining multiple neural networks by fuzzy integral for robust classification. IEEE Transactions on Systems, Man and Cybernetics 25 (2), 380–384.

Dasarthy, B., 1994. Decision Fusion. IEEE Computer Society Press, Silver Spring, MD.

Drawish, A., Auda, G., 1994. New composite feature vector for Arabic handwritten signature recognition. In: Proceedings of the 1994 International Conference on Acoustics, Speech and Signal Processing, Australia.

Drucker, H., Schapire, R., Simard, P., 1993. Improving performance in neural networks using a boosting algorithm. Neural Information Processing Systems 5, 42–49.

El-Shishiny, H., Abdel-Mottaleb, M., El-Raey, M., Shoukry, A., 1989. A multistage algorithm for fast classification of patterns. Pattern Recognition Letters 10, 211–215.

Kittler, J., 1994. Cooperative decision making processes and their neural net implementation. In: Cherkassky, V., Friedman, J., Wechsler, H. (Eds), From Statistics to Neural Networks: Theory and Pattern Recognition Applications. NATO ASI Series, Springer, Berlin, pp. 263–281.

Kittler, J., Hatef, M., Robert, D., Matas, J., 1998. On combining classifiers. IEEE Transactions on Pattern Analysis and Machine Intelligence 20 (3), 226–239.

Rogova, G., 1994. Combining the results of several neural network classifiers. Neural Networks 7 (5), 777–781.

Topping, J., 1962. Errors of Observation and Their Treatment. Chapman & Hall, London.