A Probabilistic Model for the Cooperative Modular Neural Network

Luís A. Alexandre[1,2] *, Aurélio Campilho[2,3], Mohamed Kamel[4]

[1] Departamento de Inform´atica, Universidade da Beira Interior, Covilhã, Portugal
lfbaa@di.ubi.pt
[2] INEB - Instituto de Engenharia Biom´edica, Laborat´orio de Sinal e Imagem Biom´edica,
Campus da FEUP, Rua Roberto Frias, s/n, 4200-465 Porto, Portugal
[3] Universidade do Porto, Faculdade de Engenharia, Departamento de Engenharia Electrot´ecnica
e Computadores, Porto, Portugal campilho@fe.up.pt
[4] Department of Systems Design Engineering, University of Waterloo, Ontario, Canada
mkamel@uwaterloo.ca

Abstract. This paper presents a model for the probability of correct classification for the Cooperative Modular Neural Network (CMNN). The model enables the estimation of the performance of the CMNN using parameters obtained from the data set. The performance estimates for the experiments presented are quite accurate (less than 1% relative difference). We compare the CMNN with a multi-layer perceptron with equal number of weights and conclude that the CMNN is preferred for complex problems. We also investigate the error introduced by one of the CMNN voting strategies.

1 Introduction

The basic idea behind a modular neural network (MNN) architecture [1–5] is the combination of several small networks that are trained to solve a specific part of the full problem. The output of these networks can be combined using, amongst others, rules such as the simple and weighted averages or the product [6–8] or alternatively, one of the outputs can be selected as the correct result.

Intuitively, a MNN architecture should perform better than a single network for problems that can be separated into several subproblems. In this case, there is a decoupling of the neurons (and weights) used for learning each subproblem when compared to the case of using a single network to solve the entire problem.

This paper introduces a model for the probability of correct classification for the cooperative MNN (CMNN) [1, 9, 10]. This model enables a better understanding of the way this MNN works. It also enables the estimation of the performance of the CMNN using parameters estimated from the data set. We show empirically that these estimates are accurate. We compare the CMNN with a multi-layer perceptron (MLP) with equal number of weights and conclude that the CMNN is preferred for complex problems. We also investigate the error introduced by one of the voting strategies.

* We acknowledge the support of project number POSI/35590/SRI/2000 approved by the Portuguese 'Fundação para a Ciência e Tecnologia'(FCT) and POSI and partially financed by FEDER.

Section 2 introduces the CMNN architecture and the model for the probability of correct classification (PCC). Section 3 includes the several voting strategies that can be associated with the CMNN. Section 4 contains experiments, illustrating the ideas presented in the previous sections and confirming the validity of the developed model. In the last section, the results are discussed and the conclusions posted.

2 CMNN Architecture

In this section we describe the CMNN architecture. Consider a classification problem with L classes. C_n represents class n. The input feature vector is X. The CMNN consists of k expert NNs, $g_i(X),\ i = 1, \ldots, k$, that are trained to solve a particular subproblem of the total problem, and also to recognize when the input data does not belong to its own subproblem.

A classifier g_i outputs a vector of estimates of the posterior probabilities, $p_i(X \in C_n|X)$,

$$g_i(X) = (p_i(X \in C_n|X), \ldots, p_i(X \in C_{n-1+\#I_i}|X)), n, \ldots, n-1+\#I_i \in I_i \quad (1)$$

with I_i being the set of indexes that correspond to the classes that classifier g_i can deal with and $\#I_i$ the number of corresponding classes.

We define the set containing the indexes of all the experts as

$$H = \{1, \ldots, k\} \quad (2)$$

and also

$$H_j = H \backslash \{j\},\ j \in H \quad (3)$$

Each expert g_i has also a set of $k-1$ outputs, $o_{i,j},\ j \in H_i$, corresponding to the other experts in the architecture. These outputs have values in $[0, 1]$. A higher value represents more confidence on the fact that the classifier g_j should be selected to produce the final decision.

For each input X, each expert NN produces a vector of posterior probabilities on the I_i outputs corresponding to the classes of its own subproblem, and tries to guess which classifier should be used to classify this pattern, using the remaining $k-1$ outputs.

The final decision consists on the class with the largest posterior probability from the classifier that is selected by the votes of the $o_{i,j}$ outputs of all classifiers. Several voting strategies can be considered.

This architecture is represented in figure 1.

2.1 General case

We extend the operator '$\max$' to work with vectors: it outputs the largest component of the vector. The set of points in which the event 'class n has the largest posterior probability for classifier g_i ' occurs will be represented as $B_{n,i}$:

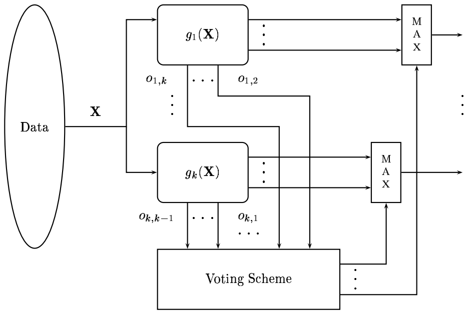

Fig. 1. The CMNN architecture.

$$B_{n,i} = \{X : p_i(X \in C_n|X) = \max g_i(X)\} \tag{4}$$

The set of points in which the event 'classifier g_i makes a correct classification' occurs will be represented by D_i:

$$D_i = \bigcup_{n \in I_i} (B_{n,i} \cap \{X : X \in C_n\}) \tag{5}$$

To simplify, will call $B_{n,i}$ an event and not the set of points where this event takes place. This will also be done for the set D_i and others to be defined below.

The event 'classifier g_i is elected as the one which will output the final decision' will be represented by F_i.

This way, the probability of correct classification for this architecture comes as

$$PCC = P\left(\bigcup_{i=1}^{k} \bigcup_{n \in I_i} (B_{n,i} \cap \{X \in C_n\} \cap F_i)\right) \tag{6}$$

Using expression 5 results

$$PCC = P\left(\bigcup_{i=1}^{k} (D_i \cap F_i)\right) \tag{7}$$

Since the events D_i are disjoint, so is the intersection $(D_i \cap F_i)$, and expression 7 can be written as

$$PCC = \sum_{i=1}^{k} P(D_i \cap F_i) \tag{8}$$

To simplify the last expression we will assume that the events D_i and F_i are independent. This leads to the following expression for PCC

$$PCC = \sum_{i=1}^{k} P(D_i)P(F_i) \tag{9}$$

This assumption can be justified since the fact that classifier g_i is the chosen one for classifying the input (event F_i) is dependent of the majority of the classifiers, thus not particularly dependent of classifier g_i (the dependence that may exist, since classifier g_i also votes, is decreased as the total number of experts increases). Since the event D_i depends exclusively of classifier g_i, it is not a strong assumption to consider its independence from F_i.

The different voting strategies will now be considered.

3 Different voting strategies

These are the voting strategies proposed by the original author of the CMNN architecture [9]. We present them in a formal manner using the events defined above and also defining new ones.

3.1 Plurality vote

In this case, each expert g_i votes only for one (other) expert: the one with the highest value of $o_{i,j}$. The expert with more votes wins.

The number of votes that classifier g_i receives is T_i:

$$T_i = \sum_{j \in H_i} \mathbb{I}_{\{\max_{n \in H_j} o_{j,n} = o_{j,i}\}} \tag{10}$$

where $\mathbb{I}_{\{A\}}$ denotes the indicator function, which gives one if the event A is true and zero otherwise.

Using this definition, we can write $F_i = \{T_i = \max_{j \in H} T_j\}$.

3.2 Borda count

The $o_{j,i}$ are ranked and a value of $k-2$ is assigned to the largest output of classifier g_j, $k-3$ to the second largest and so on, such that the smallest output receives a value of zero.

The values are summed for each classifier and the one with the largest sum is elected.

We define the function $r(o_{j,i}) : H \times H \mapsto \{1, \ldots, k-1\}$ that gives the rank of $o_{j,i}$.

The total value assigned to classifier g_i is

$$BC_i = \sum_{j \in H_i} (k - 1 - r(o_{j,n})) \tag{11}$$

The event F_i is thus $F_i = \{BC_i = \max_{j \in H} BC_j\}$.

3.3 Fuzzy vote

In this case, the elected classifier is the one with the largest summation over all values of the votes $o_{j,i}$.

We define

$$S_i = \sum_{j \in H_i} o_{j,i} \tag{12}$$

In this case, the event F_i comes as $F_i = \{S_i = \max_{j \in H} S_j\}$.

3.4 Nash vote

Nash vote is similar to fuzzy vote but instead of having a sum of the $o_{j,i}$ we have the product.

We define

$$Pd_i = \prod_{j \in H_i} o_{j,i} \tag{13}$$

In this case, we have $F_i = \{Pd_i = \max_{j \in H} Pd_j\}$.

4 Experiments

4.1 A 17 class artificial problem

An artificial problem with 2 features and 17 classes that are roughly clustered in 5 groups was produced. The classes were generated using Gaussian distributions. The data is plotted in figure 2. Each class has 150 data points, hence, the data set has 2550 data points.

The CMNN architecture consists of 5 MLPs with topologies [2:22:7] for the 3 groups with 3 classes (the other 4 outputs are for the voting scheme) and [2:20:8] for the 2 groups with 4 classes (again using 4 outputs for the voting scheme). The voting strategy used was the plurality vote. We trained a single multi-layer perceptron (MLP) with the same number of weights as the CMNN architecture (topology [2:56:17]) to give an idea of the improvement that can be obtained with the CMNN over a single MLP. Since both the CMNN and the MLP use the same number of weights, the differences of performance are related to the way the weights are connected and not to their number. All networks were trained using resilient back-propagation for 100 epochs.

Table 1 presents the average classification error and standard deviation, both in percentage, for the 10 repetitions of the leave-k-out cross-validation, with $k = 255$.

Notice that there is a third line in the table for an CMNN-IV. This is the same as the CMNN but assuming that the voting was ideal, i.e., that the experts always made the correct choice of the expert that should made the final decision. It has slight better performance than the CMNN giving an idea of the error introduced by the voting scheme, which is about 0.75%.

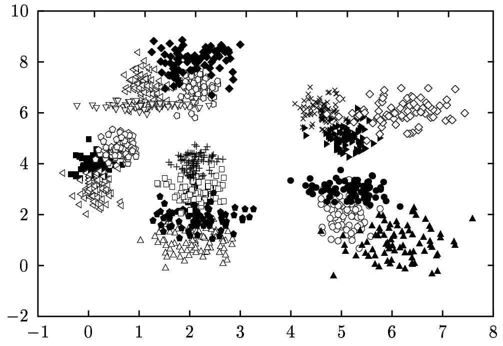

Fig. 2. The data set for the artificial problem.

Table 1. Average classification errors and corresponding standard deviations, for the artificial problem.

Architecture	*Error [%]*	*St. Dev. [%]*
MLP	17.61	2.95
CMNN	14.55	3.26
CMNN-IV	13.80	3.22

During testing, the values of $P(D_i)$ and $P(F_i)$ were estimated. These values were then used with the model for the PCC, yielding the value of 86.44%. This is equivalent to an error of 100-86.44=13.56% . This is in good agreement with the obtained value of 14.55% error for the CMNN (the difference is 0.89% out of 14.55%), thus asserting that the model developed is accurate.

4.2 A 2 group, 4 class real problem

To test the prediction capabilities of our bounds on real problems we used a data set for a vowel discrimination problem. The data consists of the first and second formants of the vowels ‘i’,‘I’,‘a’ and ‘A’ produced by 76 speakers (33 males, 28 females and 15 children). Each vowel was repeated twice by each speaker, giving a total number of 608 data points. It is a subset of the Peterson and Barney data set referred in [3] and is represented in figure 3. Both features were linearly scaled by dividing by 1000.

The CMNN architecture consists of 2 multi-layer perceptrons (MLPs) with topologies [2:15:3] - 2 outputs for each class in each group and the other for the output used for the voting strategy. The voting strategy used was again the plurality vote. We trained a single multi-layer perceptron (MLP) with the same number of weights as the CMNN architecture (topology [2:26:4]) to give an idea of the improvement that can be obtained with the CMNN over a single MLP. The networks were again trained using resilient back-propagation for 100 epochs. Table 2 presents the average classification error and standard deviation, both in percentage, for the 8 repetitions of the leave-k-out cross-validation, with $k = 76$.

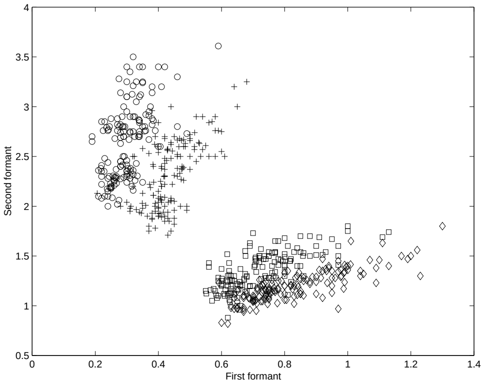

Fig. 3. Data set for a 4 class, 2 group problem.

Table 2. Average classification errors and corresponding standard deviations, for the real problem.

Architecture	*Error [%]*	*St. Dev. [%]*
MLP	6.41	2.04
CMNN	8.39	3.22
CMNN-IV	8.22	2.88

The CMNN-IV has again, and as expected, a slight better performance than the CMNN. In this case, the error introduced by the voting scheme against the CMNN with the ideal voting scheme is 0.17%.

With the estimated values of $P(D_i)$ and $P(F_i)$ replaced in the model, we obtain an estimate for the PCC of 91.64%. This is equivalent to an error of 100-91.64=8.36% . This is again in good agreement with the obtained value of 8.39% error for the CMNN. Once again the model for the PCC yields a good estimate: the difference of the estimate to the true value is only 0.03%.

In this case the MLP outperformed the CMNN. We believe that this happened because the problem was too simple for the CMNN. Some of the weights used in the voting scheme were better used by the MLP in approximating the problem as a whole.

5 Conclusions

This paper presents a model for the probability of correct classification for the cooperative modular neural network (CMNN) architecture. The validity of the presented model was confirmed by experiments using both artificial and real data sets. Its predictions of the CMNN error rates, using some estimated parameters from the data sets, were in good accordance with the empirical errors.

The error introduced by one of the voting strategies, the plurality vote, as compared with the ideal vote was also investigated. We concluded that the error the voting scheme introduces is small when compared with the error of the experts in their subproblems.

Finally, a multilayer perceptron (MLP) with equal number of weights as the CMNN was used. This makes the differences in accuracy of these two classifiers to be only due to the way the weights are connected and not to their number. The results suggest that the CMNN produces better results with problems involving several groups, i.e., if the problem is simple, a simple architecture should be used.

References

1. Auda, G., Kamel, M.: Modular neural network classifiers: A comparative study. J. Intel. Robotic Systems (1998) 117–129
2. De Bollivier, M., Gallinari, P., Thiria, S.: Cooperation of neural nets and task decomposition. In: Int. Joint Conf. on Neural Networks. Volume 2., Seattle, USA (1991) 573–576
3. Jacobs, R., Jordan, M., Nowlan, S., Hinton, G.: Adaptive mixtures of local experts. Neural Computation (1991) 79–87
4. Jacobs, R., Peng, F., Tanner, M.: A bayesian approach to model selection in hierarchical mixtures-of-experts architectures. Neural Networks **10** (1997) 231–241
5. Wanas, N., Kamel, M., Auda, G., Karray, F.: Feature-based decision aggregation in modular neural network classifiers. Pattern Recognition Letters **20** (1999) 1353–1359
6. Alexandre, L., Campilho, A., Kamel, M.: Combining independent and unbiased classifiers using weighted average. In: Proceedings of the 15th International Conference on Pattern Recognition. Volume 2., Barcelona, Spain, IEEE Press (2000) 495–498
7. Alexandre, L., Campilho, A., Kamel, M.: On combining classifiers using sum and product rules. Pattern Recognition Letters **22** (2001) 1283–1289
8. Kittler, J., Hatef, M., Duin, R., Matas, J.: On combining classifiers. IEEE Trans. PAMI **20** (1998) 226–239
9. Auda, G., Kamel, M.: CMNN: Cooperative modular neural networks for pattern recognition. Pattern Recognition Letters **18** (1997) 1391–1398
10. Auda, G., Kamel, M., Raafat, H.: Voting schemes for cooperative neural network classifiers. In: IEEE Int. Conference on Neural Networks. Volume 3., Australia (1995) 1240–1243