Cogn Comput (2010) 2:285–290
DOI 10.1007/s12559-010-9051-6

Multiclass Pattern Recognition Extension for the New C-Mantec Constructive Neural Network Algorithm

José L. Subirats · José M. Jerez · Iván Gómez · Leonardo Franco

Received: 9 March 2010 / Accepted: 18 May 2010 / Published online: 3 June 2010

Abstract The new C-Mantec algorithm constructs compact neural network architectures for classsification problems, incorporating new features like competition between neurons and a built-in filtering stage of noisy examples. It was originally designed for tackling two class problems and in this work the extension of the algorithm to multi-class problems is analyzed. Three different approaches are investigated for the extension of the algorithm to multi-category pattern classification tasks: One-Against-All (OAA), One-Against-One (OAO), and P-against-Q (PAQ). A set of different sizes benchmark problems is used in order to analyze the prediction accuracy of the three multi-class implemented schemes and to compare the results to those obtained using other three standard classification algorithms.

Keywords Supervised learning · Neural networks · Multiclass pattern recognition

Introduction

A main issue at the time of implementing feed-forward neural networks in classification or prediction problems is the selection of an adequate architecture [1, 2, 3]. Several constructive methods [4] and pruning techniques [1] have been proposed as an alternative for the architecture selection process, but it is a research issue whether these methods can achieve the same level of prediction accuracy than standard feed-forward neural networks trained by back-propagation [5, 6]. Constructive algorithms start with a very small network, normally comprising a single neuron, and work by adding extra units until some convergence condition is met [7–10], obtaining an architecture together with the set of weights that implement the available training examples. Most prediction systems are designed for separating two classes of inputs, and while there exists trivial extensions to multi-class problems, this process is not straightforward as particular features of the algorithms have to be analyzed. Multi-class pattern recognition is relevant because it has a wide range of applications, including for example, handwritten digit recognition, object classification, speech tagging, speech recognition, and text categorization. Multi-class problem classification has been relatively less investigated than its binary counterpart and in this work we analyze the extension to multi-class problems for the C-Mantec (Competitive Majority Network Trained by Error Correcting) neural network constructive algorithm.

A multi-class, denoted a K-class, neural network classification can be described formally as follows. For a given feature space, Ω , and a set of class labels, $CL = \{cl_1, cl_2, ..., cl_K\}$, where $cl_j \neq cl_h$ for all $h \neq j$ and $K > 2$, we need to develop a system F such that for any given feature vector $x \in \Omega, F(x) : CL$.

We analyze three different known schemes in which a K-class pattern classification problem using a neural network system can be modelled [11]: One-Against-All (OAA), One-Against-One (OAO), and P-Against-Q (PAQ), where P and Q are greater than 1. The following sections describe the C-Mantec algorithm, the K-classes schemes recently mentioned, the application to a set of benchmark functions and the results obtained.

J. L. Subirats · J. M. Jerez · I. Gómez · L. Franco (✉)
Departamento de Lenguajes y Ciencias de la Computación, Universidad de Málaga, Campus de Teatinos, 29071 Málaga, Spain
e-mail: lfranco@lcc.uma.es

The C-Mantec Algorithm

C-Mantec is a constructive neural network algorithm that creates architectures containing a single layer of hidden nodes with threshold activation functions [12]. For binary classification tasks, the constructed networks have a single output neuron computing the majority function of the responses of the hidden nodes. The learning procedure starts with an architecture comprising a single neuron in the hidden layer and continues by adding a neuron every time the present ones are not able to learn the whole set of training examples. The neurons learn according to the thermal perceptron learning rule proposed by Frean [8], for which the synaptic weights are modified according to Eq. 1.

$$\delta w_i = (t - o)\, \psi_i\, T_{\text{fac}}, \tag{1}$$

where t is the target value of the current example presented to the network, o represents the actual output of the neuron and ψ is the value of input unit i corresponding to the pattern being presented. T_{fac}, a relevant factor for the functioning of the algorithm, is defined as:

$$T_{\text{fac}} = \frac{T}{T_0} \exp\left\{-\frac{|\phi|}{T}\right\}. \tag{2}$$

In the previous equation, T is an introduced artificial temperature, T_0 the initial temperature value and ϕ is the synaptic potential defined as:

$$\phi = \sum_{i=1}^{N} w_i\, \psi_i \;+\; T_h, \tag{3}$$

where w_i are the synaptic weights, T_h is the neuron threshold, and ψ_i indicates the set of inputs. The thermal perceptron can be seen as a modification of the standard perceptron rule that incorporates a modulation factor forcing the neurons to learn only target examples close to the already learnt ones. For a deeper analysis of the thermal perceptron rule, see the original paper [8].

The network generated by the algorithm has an output neuron computing the majority function of the activation of the neurons belonging to the single hidden layer. If the target of a given example is not matched by the network output, implying that more than half of the neurons in the hidden layer wrongly classify the current input, the algorithm select one of the "wrong" neurons in a competitive process. Target values for the individual neurons (indicated by "t" in Eq. 1) are the same than for the whole problem, i.e., each individual neuron in the hidden layer tries to approximate the whole problem. The global network solution for a given input pattern (the output of the network), depends on the activations of the hidden neurons through an output neuron that implements the majority function of the hidden neurons (a voting scheme). When the output of the network does not coincide with the target value of an input example, a neuron is chosen in a competitive process implemented as follows: for a given input example that is wrongly classified by the whole network, the algorithm selects, among the neurons whose activation state does not coincide with the target value, the neuron with the largest value of T_{fac}. This chosen neuron will modify its synaptic weights according to the learning rule of the thermal perceptron indicated in Eq. 1, but only if its value of T_{fac} is larger than the value of a parameter of the algorithm named growing factor, g_{fac}. The g_{fac} parameter is set at the beginning ot the process and is related to the growth of the neural architecture. In case that the condition $T_{\text{fac}} > g_{\text{fac}}$ is not met, the network increases the size of its hidden layer by adding a new neuron that will learn the presented input. Everytime a neuron modifies it synaptic weight because has been selected to learn an input example, its internal temperature, T, is lowered. Also whenever a new neuron is added to the network, the temperature of all neurons is reset to the initial value T_0. The learning procedure continues in this way until enough neurons are present in the architecture and the network is able to learn the whole sets of inputs. Regarding the role of the parameters, higher values for T_0, the initial temperature, ensures that every neuron will be given a large number of learning iterations to learn and also that there will be an initial phase of global exploration for the weights values, as for high temperature values the synaptic weights have a larger probability of being modified. A second parameter of the algorithm is the growing factor, g_{fac}, related to the size of the final architectures, as it regulates whether a given input is going to be learnt by one of the existing neurons or whether a new unit is going to be added. The parameter setting for the algorithm is relatively simple as C-Mantec has been shown to be very robust to changes in a wide range of values. The convergence of the algorithm is ensured as the learning rule is very conservative, preserving the acquired knowledge of the neurons and because of the fact that new introduced units learn at least one input example.

One problem that affects constructive algorithms, and in general any classification system, is the problem of overfitting [13]. In the case of the C-Mantec and other constructive algorithms, the overfitting problem is magnified because these learning schemes normally try to learn the training set of patterns until zero error is achieved. Thus, in order to avoid overfitting and boost the generalization ability of the algorithm, an active learning scheme that eliminates input examples considered noisy was applied. The method is very straightforward and is based on counting the number of times that every input example is tried to be learnt by the network. These examples, that are difficult to be correctly classified by the system, are eliminated when the number of presentations for an input example is larger by more than two standard deviation from the mean for the whole dataset.

A difference of the C-Mantec algorithm with existing constructive algorithms is the fact that C-Mantec does not freeze the weights of the existing neurons. In this way all neurons can learn during the whole training procedure in a competitive process between them. This constitutes a big difference in comparison to previous research, as the whole model can adapt globally its weights and size according to the presentation of inputs. In order to keep all neurons continuously learning, it is necessary to use a conservative learning rule at the level of a single neuron, like the thermal perceptron, that ensures that the knowledge incorporated into the system is preserved.

Multi-Class Classification Using Multiple Binary Neural Network

A K-Class pattern recognition task can be implemented by a system formed by *M* binary neural networks and an additional decision module. The *M* binary neural networks are trained separately using a common training data set or a subset of the data set, and a decision module is used to select the final classification result based on the outputs from the *M* neural networks, as shown in Fig. 1. The value of *M* and the training methodology depends on the modelling scheme used. In this work, we applied three different approaches, One-Against-All (OAA), One-Against-One (OAO), and P-Against-Q (PAQ), that are described below and then tested using the C-Mantec algorithm.

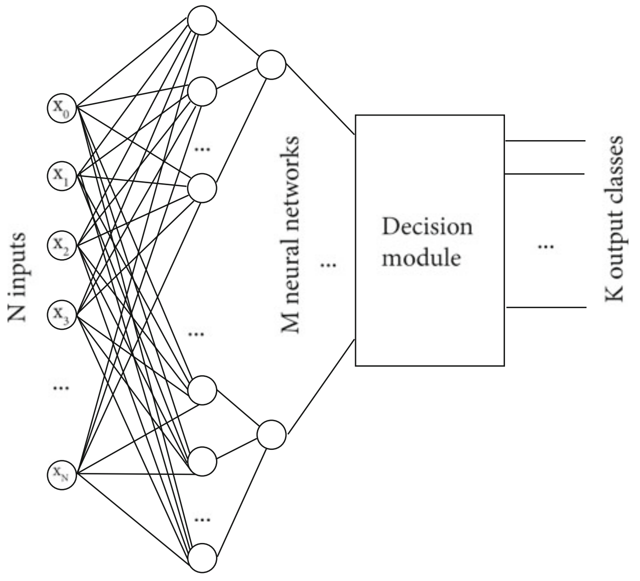

Fig. 1 Schematic drawing of the architecture of a neural classifier for a K-classes problem: *M* binary networks are trained individually and a final decision module is used for selecting the output class

One-Against-All (OAA) Neural Networks

The One-Against-All (OAA) scheme utilizes $M = K$ binary neural networks, where K is the number of classes of the original task. Each neural network is trained with the same training data set but with different objective values. In each of the *K* networks, one of the K-classes is assigned the target value 1 while the rest of classes is assigned the value 0. In this scheme, there are three possible cases to be analyzed by the decision module. First, if only one of the *K* neural networks outputs 1 and all others output 0, the decision is easily made. Second, if more than one of the binary networks output 1, the decision module have to decide whether to output a symbol to indicate a tie, or employs a more sophisticated scheme to force a classification decision. Third, if none of the neural networks outputs a 1, then the decision module needs to indicate that no classification is made. For the second case, in which more than one output module outputs a "1", our implementation prioritizes the neural network with the better classification rate obtained thus far, computed on-line. To resolve the third case, in which no output is produced by the binary modules, we select the class corresponding to the neural network with the worse classification rate. One problem associated with this type of architecture is that the training data for individual neural networks can be heavily unbalanced, in the ratio of 1:(K − 1) if the number of the training samples in each class is about the same size. Figure 2a illustrates an implementation of the OAA scheme for a three-class problem. This implementation generate three neural networks that separate one class against all others. While there are three regions where only one network is active and a decision can be easily made, there are other three where two neural network outputs are active, and a region where none active response is made. These "ill-defined" regions have to be solved by some strategy and we decided the response of the network based on the proportion of examples allocated previously to each class.

One-Against-One(OAO) Neural Networks

The One-Against-One scheme transforms a K-Class pattern classification problem into $M = K(K - 1)/2$ sub-problems containing only two-class examples. Each neural network solves a classification problem of an individual class against another and is trained only with subset of the data set where these two classes are active. The collective output from all *M* binary neural networks for an input

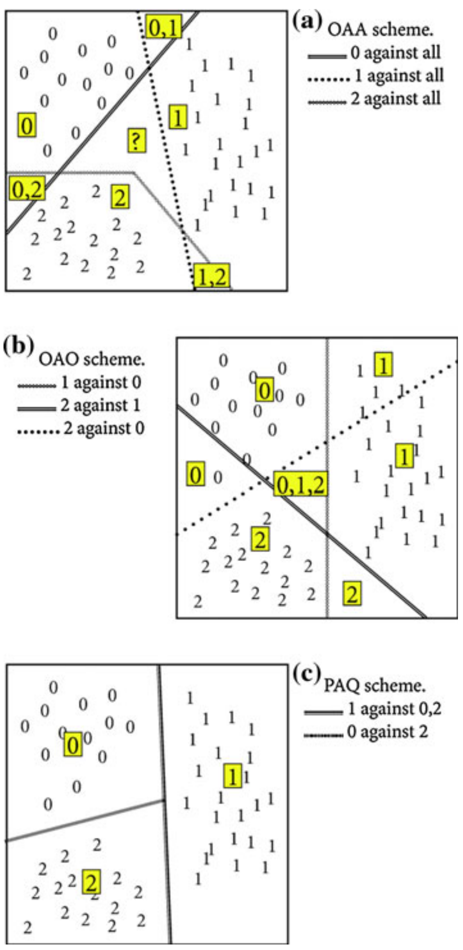

Fig. 2 Schematic drawing of the three different approaches used for multi-class classification using binary neural network modules, exemplified for a three classes problem with output values 0,1 and 2. The top graph shows the One-Against-All (OAA) scheme, in the middle the One-Against-One (OAO) approach is presented, and the bottom figure shows the P-Against-Q (PAQ) case

feature vector ‘x’ represents a combination of M votes for K classes, and the decision module needs to decide the output class of the scheme. A simple voting scheme can be used by the decision module. This module counts the votes for each class based on the outputs from the M neural networks and the class with more votes is assigned to the input feature vector ‘x’. The problem in this case is how the decision module resolves ties (equal number of votes for two or more classes). A tie can be broken by chosen the class with higher prior-probability, weighting the votes by the confidence value associated with each neural network output taken into account. The major advantage of OAO approaches is that provides redundancy that can lead to a more robust system. The disadvantage of this approach is that generates a large number of neural network modules, specially where K is large. Figure 2b illustrates a possible implementation of the OAO scheme for a three-class problem. Each class is classified against all others in different regions of the space. An input example that belongs to one of the regions will be classified according to the majority of the votes. In the middle regions ties occur and for these cases extra rules are employed. In our implementation, ties where solved by restricting the dataset to the classes involved and computing the voting only from these restricted cases. When the tie cannot even be resolved by this procedure, a random choice between the involved classes is taken.

P-Against-Q (PAQ) Neural Networks

The P-Against-Q classification scheme can be considered as a halfway approach between the other two mentioned before. In this approach, the original classes are grouped M different times as two class problems, in a way that from the output of these M groups is possible to infer the output class. The implementation can be considered as M binary codes of length K, where each code has P bits equal to one and $Q = M - P$ bits equal to zero. One type of P-against-Q encoding consists in using the shorter code that specify all classes, $M = \log_2 K$ bits. This dense encoding is efficient in terms of the resulting size of the architecture but not in terms of the generalization obtained, as some redundancy on the encoding is usually beneficial. In our implementation $2K$ modules were used and the grouping of the classes was done by random choices. Figure 2c illustrates a possible implementation of the PAQ scheme for a three-class problem. In the example, the encoding uses only two bits with codeworks ‘00’,‘11’,‘01’ for the three classes 0,1,2, respectively.

Experiments and Results

In a first set of experiments we tested the three multi-class schemes mentioned before (OAA, OAO and PAQ) using the C-Mantec constructive algorithm on a well known set of benchmark problems. The set consists in five multi-class problems with a number of classes between 3 and 19. The C-Mantec algorithm was run with a maximum number of iterations of 10.000, and an initial temperature (T_0) value equals to the number of inputs of the analyzed functions. The data was split in training and test sets using 75 and 25% of the available instances respectively. The results obtained are shown in Tables 1 and 2 where the size of the obtained architectures and the generalization ability obtained are shown for the three schemes applied. Both features were measured as the mean of 25 independent runs and the standard deviation computed. The last column of Table 2 shows, as a comparison, the generalization ability values obtained by Prechelt [14] in a work where he

Table 1 Results for the number of neurons obtained with the C-Mantec algorithm using three different multiclass scheme

Function	Inp.	Cl.	N. OAA	N. OAO	N. PAQ
Thyroid	21	3	2 ± 0	3 ± 0	5 ± 0
Horse1	58	3	4.7 ± 0.7	3 ± 0	9.4 ± 0.9
Gene1	120	3	5.2 ± 3.0	3.5 ± 1.1	8.4 ± 2.7
Glass	9	6	11.4 ± 2.0	17.8 ± 1.7	35.1 ± 3.5
Soybean	82	19	19 ± 0	171 ± 0	24 ± 0
Average			8.5 ± 6.8	39.6 ± 73.7	16.4 ± 12.7

analyzed in a systematic way the prediction capabilities of different topologies neural networks, and thus we believe that these reported values are highly optimized. Among the results between the three schemes analyzed using the C-Mantec algorithm, the PAQ approach leads to the best average results; the OAO scheme generalization values came close behind, but it has to be said that this approach needs a much larger number of neurons for problems with large number of classes. The OAA approach leads to the worst results but were obtained with a lower number of units. In comparison to the results by Prechelt [14], the C-Mantec algorithm combined with the PAQ Multi-Class scheme outperformed them in 3 out of the 5 considered problems with an average generalization ability 2.06% larger.

We also carried a second set of experiments in which we compared the generalization ability obtained using the C-Mantec algorithm (using the OAO multi-class scheme) to the results from some standard algorithms frequently used in the literature: C4.5 decision trees, Multilayer Perceptrons trained by standard backpropagation (MLP) and Support Vector Machines (SVM). The framework WEKA was used for the comparison with the default settings for the parameters values. The OAO approach was chosen for the C-Mantec algorithm because good results were observed in the first test (Table 2) and even if they were slightly worse than those from the PAQ approach the implementation of the OAO scheme is much more straightforward. For the comparison, four multi-class benchmark sets from the UCI dataset were used in a cross-validation procedure with a training/test splitting of 75/25%. The results are presented in Table 3, where it can be shown that the results of the C-Mantec algorithm are very close to the best ones, that were obtained by the MLP.

As an overall conclusion, the experiments presented in this work show the suitability of the C-Mantec algorithm for multi-class classification problems. These results indicates that the C-Mantec algorithm is a valid alternative to standard FFNN and other popular predictive models with the advantage of avoiding the problem of selecting an adequate architecture, that is automatically selected by the algorithm. We are currently on the process of carrying complementary experiments using larger sets of benchmark functions and optimizing some parameters of the C-Mantec algorithm through the use of a validation set.

Table 2 Generalization ability obtained with the C-Mantec algorithm using three different multiclass scheme

Function	Inp.	Cl.	G. OAA	G. OAO	G. PAQ	Gen. [14]
Thyroid	21	3	94.2 ± 0.6	94.2 ± 0.6	94.1 ± 0.5	93.4 ± 0.0
Horse1	58	3	64.8 ± 6.4	66.2 ± 5.2	67.7 ± 5.7	73.3 ± 1.9
Gene1	120	3	84.0 ± 1.1	88.5 ± 1.0	86.2 ± 1.1	86.4 ± 0.1
Glass	9	6	58.4 ± 7.8	64.8 ± 6.3	67.0 ± 6.3	53.9 ± 2.2
Soybean	82	19	90.6 ± 3.2	92.6 ± 2.0	92.9 ± 2.6	90.6 ± 0.5
Average			78.4 ± 15.9	81.3 ± 14.6	81.6 ± 13.3	79.5 ± 16.2

The last column shows the results from [14] (See text for more details)

Table 3 Generalization ability obtained with the C-Mantec algorithm using the One-against-one (OAO) scheme and by other three algorithms (C4.5, MLP and SVM) on a set of 4 multiclass problems

Function	Inp.	Cl.	C-Mantec	C4.5	MLP	SVM
Iris	4	3	94.86 ± 2.3	94.44 ± 3.1	95.17 ± 3.8	96.55 ± 1.3
Balance-scale	4	3	90.58 ± 1.1	78.46 ± 3.4	90.99 ± 1.4	87.21 ± 1.5
Waveform	40	3	86.32 ± 0.7	74.47 ± 1.1	83.36 ± 0.7	86.02 ± 1.0
Abalone	8	29	21.73 ± 1.3	20.92 ± 1.2	26.55 ± 1.4	24.63 ± 1.1
Average			73.37	67.07	74.02	73.60

Acknowledgements The authors acknowledge support from MICIIN (Spain) through grant TIN2008-04985 (including FEDER funds) and from Junta de Andalucía through grants P06-TIC-01615 and P08-TIC-04026.

References

1. Haykin S. Neural networks: a comprehensive foundation. NY: Macmillan/IEEE Press;1994.
2. Lawrence S, Giles CL, Tsoi AC. What size neural network gives optimal generalization ? Convergence properties of backpropagation. In: Technical report UMIACS-TR-96-22 and CS-TR-3617, Institute for Advanced Computer Studies, Univ. of Maryland. 1996.
3. Gómez I, Franco L, Jerez JM. Neural network architecture selection: Can function complexity help? Neural Process Lett. 2009;30:71–87.
4. Franco L, Elizondo D, Jerez JM, editors. Constructive neural networks. Berlin: Springer; 2010.
5. Smieja FJ. Neural network constructive algorithms: trading generalization for learning efficiency? Circuits Syst Signal Process. 1993;12:331–74.
6. do Carmo Nicoletti MC, Bertini JR. An empirical evaluation of constructive neural network algorithms in classification tasks. Int J Innov Comput Appl. 2007;1:2–13.
7. Mezard M, Nadal JP. Learning in feedforward layered networks: the tiling algorithm. J Phys A. 1989;22:2191–204.
8. Frean M. The upstart algorithm: a method for constructing and training feedforward neural networks. Neural Comput. 1990;2:198–209.
9. Parekh R, Yang J, Honavar V. Constructive neural-network learning algorithms for pattern classification. IEEE Trans Neural Netw. 2000;11:436–51.
10. Subirats JL, Jerez JM, Franco L. A new decomposition algorithm for threshold synthesis and generalization of Boolean functions. IEEE Trans Circuits Syst I. 2008;55:3188–96.
11. Ou G, Murphey YL. Multi-class pattern classification using neural networks. Pattern Recognit. 2007;40:418.
12. Subirats JL, Jerez JM, Franco L. A local stable learning rule and global competition for generating compact neural network architectures with good generalization abilities: the C-Mantec algorithm. Submitted. 2010.
13. Hawkins DM. The problem of overfitting. J Chem Info Comput Sci. 2004; 44:1–12.
14. Prechelt L. Proben 1—A set of benchmarks and benchmarking rules for neural network training algorithms. Technical Report. 1994.