Ensembles of Artificial Neural Networks with Heterogeneous Topologies

Miguel Rocha[1], Paulo Cortez[2] and José Neves[1]

[1] Dep. Informática [2] Dep. Sistemas de Informação
Universidade do Minho, PORTUGAL
mrocha@di.uminho.pt pcortez@dsi.uminho.pt jneves@di.uminho.pt

Abstract

Within the *Machine Learning* field, the emergence of ensembles, combinations of learning models, has been boosting the performance of several algorithms. Under this context, *Artificial Neural Networks (ANNs)* make a fruitful arena, once they are inherently stochastic. In this work, ensembles of *ANNs* are approached, being used several output combination methods and two heuristic ensemble construction strategies. These were applied to real world classification and regression tasks. The results reveal some improvements of ensembles over single *ANNs*, favoring the combination of *ANNs* with distinct complexity (topologies) and the weighted averaging of the outputs as the combination method. The proposed approach is also able to perform automatic model selection.

Keywords: Ensembles, Multilayer Perceptrons, Classification, Regression

1 Introduction

With the advances in information technology, there has been an ever-increasing load of data in organizations, that can be used in order to enhance the decision making process. Yet, most of this data presents high complexity, while human experts are limited and may overlook important and useful information. Therefore, an increasing attention has been set over the *Machine Learning (ML)* field, which aims at building automatic problem solving models.

In particular, *Artificial Neural Networks (ANNs)* are connectionist models that mimic the central nervous system, that have been successfully applied in the design of intelligent systems, over a broad set of fields (e.g., *Data Mining* or *Engineering*). *ANNs* are appealing due to their capability to model complex and multi-dimensional data, even when noise is present [1]. In fact, when compared to other *ML* techniques, *ANNs* are known to behave well in terms of predictive knowledge [9], and there has also some been research in terms of explanatory knowledge (e.g., by extracting rules from trained *ANNs*) [4].

The interest in *ANNs* for supervised learning was stimulated by the advent of the *Backpropagation* algorithm and since then several fast variants have been proposed (e.g., *RPROP*) [11]. These training algorithms minimize an error function by tuning the modifiable parameters (or connection *weights*) of a fixed architecture (or *topology*), which needs to be set a priori. On the other hand, the design of *ANNs* is a complex task; i.e., a small network will provide limited learning capabilities, while a large one will induce generalization loss (i.e., overfitting).

One emergent research area involves the use of *ensembles* in supervised learning, where a set of *ML* models are combined in some way to produce an answer [3]. This interest arose due to the discovery that ensembles are often more accurate than individual models. Ensembles work if the errors made by the models are uncorrelated and this condition is easily met, specially when the algorithm is stochastic in nature.

Following this trend, the present work aims at adopting several ensemble combinations of *ANNs* in classification and regression tasks, in order to

gain improved performances. Furthermore, a novel heuristic method for ensemble construction will be proposed, based in the use of heterogeneous topologies (i.e., the use of *ANNs* with different learning complexities), which is expected to alleviate the burden of *ANN* design.

The paper is organized as follows: first, a description of the *ANNs* and ensembles is given; next, a set of experiments is conducted, being the results analyzed and discussed; finally, closing conclusions are drawn.

2 Artificial Neural Networks

In *MultiLayer Perceptrons (MLPs)*, one of the most popular *ANN* architectures, *neurons* are grouped into *layers* and only *forward connections* exist [1]. The state of a neuron (s_i) is given by:

$$s_i = f(w_{i,0} + \sum_{j \in I} w_{i,j} s_j) \tag{1}$$

where I represents the set of nodes reaching node i, f the activation function (possibly of nonlinear nature), $w_{i,j}$ the weight of the connection between nodes j and i (when $j = 0$, it is called *bias*), and $s_1 = x_1, \ldots, s_n = x_n$, being $x_1, \ldots, x_n$ the input vector values for a network with n inputs.

In this study, output representation is approached by a single binary output, when two classes are present, being used one boolean value per each possible class, for the other classification tasks. In regression problems, one real-valued output encodes the *dependent* target variable.

For the classifications problems, the foretold class (F_i) for the i example, is given by the nearest class value to the node's output, if one single node is used, otherwise the node with the highest output value is considered:

$$F_i = \begin{cases} C_{round(S_{i,1})+1} \ , \ if(M = 2) \\ C_j : S_{i,j} = max(\{S_{i,1}, \ldots, S_{i,M}\}) \ , \ else \end{cases} \tag{2}$$

where $\{C_1, C_2, ..., C_M\}$ denotes the set of classes, $S_{i,j}$ the j output node value for the i input example, and $round(x)$ gives nearest integer to the x value.

It is also possible to compute the probability ($P_{i,c}$) associated to a given class c:

$$P_{i,c} = \begin{cases} 1 - S_{i,1} & , \ if(c = C_1 \wedge M = 2) \\ S_{i,1} & , \ if(c = C_2 \wedge M = 2) \\ \frac{\sum_{k : c=C_k} S_{i,k}}{\sum_{j=1}^{M} S_{i,j}} & , \ else \end{cases} \tag{3}$$

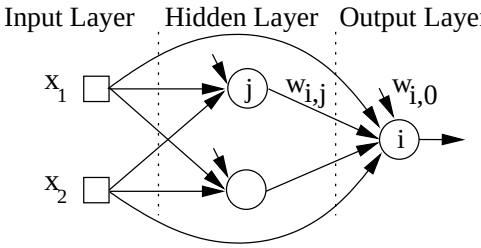

Figure 1: A fully connected *MLP* with 2 inputs, 2 hidden nodes , 1 output, bias and shortcut connections.

The *MLPs* used in this work make use of biases, sigmoid activation functions and fully connected topologies, with one hidden layer, containing a variable number of nodes (H). Different activation functions were adopted for the regression problems, since outputs may range out of the codomain ($[0, 1]$). In this case, the *logistic* activation function was applied on hidden nodes, while the output ones used shortcut connections and *linear* functions, to scale the range of the outputs (Figure 1). This solution avoids the need of filtering procedures, which may give rise to loose information (e.g., *rescaling*).

The initial weights will be randomly set within the range $[-1, 1]$. Then, the *RPROP* algorithm [11] is selected for training, due to its faster convergence and stability, being stopped after a maximum of 500 epochs or when the error slope is approaching zero (less than 0.01), using the criteria defined by Prechelt [10].

For each *MLP* learning model, an accuracy estimate was reached using K-fold cross-validation [5], where each data set is randomly subdivided into K disjoint subsets of equal cardinalities. Each subset is, in turn, not used for training, but only for testing being the overall estimate the average of the K values. The K-fold cross validation process is repeated N times and averaged, using different parti-

tions of the initial data set. In this case, the values of $K = 5$ and $N = 10$ were used. For each trial, 20% of the available data will be used in the test set. The remaining data will be used as training (50%, to set up the *MLP*'s weights) and validation (30%, to measure the *MLP*'s generalization capabilities).

Two distinct accuracy measures were adopted: the *Percentage of Correctly Classified Examples (PCCE)* (also known as *predictive accuracy*), used in classification tasks; and the *Normalized Root Mean Squared Error (NRMSE)*, applied in the regression ones. These metrics are given by the equations:

$$\begin{array}{l} PCCE = \frac{\sum_{i=1}^{N} 1\ ,\ if(F_i = T_i)}{N} 100\ (\%) \\ RMSE = \sqrt{\frac{\sum_{i=1}^{N} (T_i - S_{i,1})^2}{N}} \\ NRMSE = \frac{RMSE}{max(\{T_1,...,T_N\})} \end{array} \quad (4)$$

where N denotes the number of examples and T_i the target value for the i example.

3 Ensembles

When approaching ensembles, two main issues need to be contemplated: the method used to construct the ensemble and the how to combine its outputs.

Within *ensemble construction*, there are methods based in manipulating the training set (e.g., *bagging*, *cross validation* or *boosting*), while others work by *Injecting Randomness (IR)* [3]. This last approach is the most common with *ANNs* and will be followed in this work, consisting of training *ANNs* with distinct sets of random initial weights. Since *ANNs* employ local search training algorithms (e.g., *Backpropagation*), several uncorrelated local minima may be obtained, thus making advisable the use of *IR* ensembles.

However, *IR* still requires a correct design of the *ANN* topology. To solve this handicap, another construction method will be proposed, which involves the use of *Heterogeneous Topologies (HT)*; i.e, the ensemble population is made up of different models, ranging from the simplest linear *ANN*, with no hidden nodes ($H = 0$), to more complex ones ($H = L - 1$, where L denotes denotes the size of the ensemble *ANNs*).

There are several ways to combine multiple *ANNs*. In this work, four possibilities will be considered (only the last two ones can be used in regression tasks) [6][7]:

Voting. The output is given by the majority output of the *ANNs*:

$$\begin{array}{l} V_{i,c} = \sum_{k=1}^{L} 1\ ,\ if(c = F_{i,k}) \\ E_i = C_j : V_{i,j} = max_{k \in \{1,...,M\}}(V_{i,k}) \end{array} \quad (5)$$

where $V_{i,c}$ denotes the number of votes for class c and input i, $\{A_1, ..., A_L\}$ the set of *ANNs* in the ensemble, $F_{i,k}$ the foretold class given by A_k for example i, and E_i the final ensemble output class for example i.

Winner-takes-all. The output is decided by the *ANN* whose output class has the highest probability:

$$E_i = c : P_{i,c,k} = max_{k \in \{1,.....,L\}}(P_{i,F_{i,k},k}) \quad (6)$$

where $P_{i,c,k}$ denotes the probability of class c for example i, given by A_k.

Average. In this method, the *ANN* outputs are directly manipulated, being computed as a simple average over all *ANNs* in the ensemble, as given by:

$$E_{i,j} = (\sum_{k=1}^{L} S_{i,j,k})/L \quad (7)$$

where $S_{i,j,k}$ denotes the j output node value for the i example, given by A_k, $E_{i,j}$ is the final ensemble output for the i example and j output node. For classification tasks, the interpretation of the outputs is made over the averaged values, being E_i given by replacing $S_{i,j}$ with $E_{i,j}$ in equation 2.

Weighted average. Similar to the previous method, although each model is weighted (W_k) in proportion to its ranking (computed by the error in the validation set):

$$\begin{array}{l} W_k = \frac{L - R_k + 1}{L(L+1)/2} \\ E_{i,j} = (\sum_{k=1}^{L} W_k S_{i,j,k})/L \end{array} \quad (8)$$

where R_k denotes the ranking of A_k.

4 Experiments

4.1 Data Sets

The fifteen data sets used in the experiments were selected from the UCI *ML* repository [2] and its main features are listed in Table 1, namely the number of numeric (Num), binary (Bin) and nominal (Nom) input attributes, as well as the number of examples and classes, being the regression tasks identified by the symbol $\Re$ in the last field. The attributes labeled as *nominal* are discrete with three or more distinct values.

4.2 Preliminary Experiments

All experiments were conducted using a *MLP* software package developed in the *Java* programming language and were run on a *Pentium IV 2.4 MHz* processor, under the *Linux* operating system.

For the initial experiments, a single *MLP* was tested, with the number of hidden nodes ranging from 0 (simple perceptron) to a maximum of 20. Tables 2 and 3 show the *MLPs* with the best generalization capabilities (measured by the validation error, which is computed during the training phase). Then, the final accuracy error is calculated over the test set (column *Error*).

In terms of the *MLP* complexity, it is interesting to notice two distinct types of learning tasks: the ones whose best predictive model is linear (e.g., *Credit*, *Auto-horse*) and those which require high nonlinearity (e.g., *Balance*, *Housing*).

Table 2: Single ANN Classification Results.

Problem	*Hidden Nodes*	*Error (PCCE)*
Balance	20	94.8
Bupa	1	68.4
Car	16	97.4
Credit	0	84.9
Glass	0	63.4
Ionosphere	19	88.9
Pima	0	77.1
Sonar	11	79.9
Yeast	3	58.2
Mean		79.2

Table 3: Single ANN Regression Results.

Problem	*Hidden Nodes*	*Error (NMSE)*
Auto	11	0.070
Autos-horse	0	0.048
Cholesterol	1	0.096
Cpu	6	0.064
Housing	17	0.083
Servo	16	0.119
Mean		0.080

4.3 Injecting Randomness

In order to evaluate the prediction accuracy obtained by considering ensembles of *MLPs*, a set of experiments was conducted. For each task, the best topology obtained in the previous experiments was adopted, using an ensemble of L *MLPs*, initialized with different sets of random weights. The value of L was set to 21, an odd number (prevents ties in *Voting*) chosen to allow comparisons with other approaches. All parameters related to the *MLP*'s training are kept from the previous experiments.

The results are shown in Tables 4 and 5, in terms of the accuracy measures for all data sets and output combining methods. The last row averages the results of the two types of problems.

First, the classification results will be analyzed. Regarding the *Voting* method, the results are similar to those obtained with a single *MLP*. This is probably the result of the use of homogeneous models. The *Winner-takes-all* shows some improvement, which makes clear the advantage of using probabilistic information. However, the best results are obtained by the averaging methods, both presenting similar performances. These methods give an extra importance to higher output values, thus implicitly manipulating the classes' probabilities.

In the regression tasks, only averaging methods are applicable. As before, both behave similarly, being the improvement over single *MLPs* quite visible.

4.4 Heterogeneous Topologies

A different strategy was followed in what concerns the construction of the ensemble, considering *Heterogeneous Topologies (HT)*. In this case, the num-

Table 1: A summary of the data sets used in the experiments.

Problem	**Input Attributes**				**Examples**	**Classes**
	Num	**Bin**	**Nom**	**Total**		(M)
Auto	5	0	2	7	398	$\Re$
Autos-horse (auto-mpg)	17	3	5	25	205	$\Re$
Balance scale	4	0	0	4	625	3
Bupa (liver-disorders)	6	0	0	6	345	2
Car	0	0	6	6	1728	4
Cholesterol (heart-disease)	6	3	4	13	303	$\Re$
Credit (crx)	6	4	5	15	690	2
Cpu (machine)	6	0	0	6	209	$\Re$
Glass	9	0	0	9	214	6
Housing	12	1	0	13	506	$\Re$
Ionosphere	34	0	0	34	351	2
Pima	8	0	0	8	768	2
Servo	2	0	2	4	167	$\Re$
Sonar	60	0	0	60	104	2
Yeast	7	1	0	8	1484	10

Table 4: *IR* Ensemble Classification Results ($PCCE$).

Problem	*Voting*	*Winner-takes-all*	*Average*	*Weighted Average*
Balance	94.8	95.3	96.3	96.5
Bupa	68.4	70.5	70.6	70.2
Car	97.4	98.2	98.7	98.8
Credit	84.7	84.8	85.1	84.3
Glass	62.2	62.6	62.9	64.0
Ionosphere	89.8	90.8	92.3	92.3
Pima	76.8	77.0	77.0	76.7
Sonar	78.1	81.3	82.9	82.6
Yeast	59.0	59.2	59.7	59.5
Mean	79.0	80.0	80.6	80.5

Table 5: *IR* Ensemble Regression Results ($NRMSE$).

Problem	*Average*	*Weighted Average*
Auto	0.062	0.060
Autos-horse	0.046	0.049
Cholesterol	0.095	0.096
Cpu	0.049	0.051
Housing	0.073	0.073
Servo	0.095	0.090
Mean	0.070	0.070

ber of hidden nodes ranges from $H = 0$ to $H = 20$, thus $L = 21$. The results are shown in Tables 6 and 7.

When comparing these results with the *IR* ensembles, the accuracy is quite similar. The exception is the voting scheme, where there is a decrease in the performance. This is specially true in the problems where nonlinear models are necessary (e.g., *balance*), being the votes coming from models with insufficient complexity a source of noise.

The *Weighted Average* method presents, in this case, some advantages over regular averaging, which is natural since heterogeneous models, with different complexities, are contemplated. The error in the validation set provides a good basis for model

differentiation. Indeed, this alternative manages to obtain the best overall results.

It must be noticed that the *HT* approach does not require a previous model selection stage (which occurs with the *IR* method), thus requiring less computational effort.

An analysis of the individual accuracies, in each data set, reveals that the problems, whose best models are simpler, are the ones in which ensembles are unable to improve significantly (e.g., *Credit*, *Cholesterol*). In some cases (e.g., *Pima*), there is even no ensemble strategy capable of outperforming a single *ANN*.

Table 7: *HT* Ensemble Regression Results ($NRMSE$).

Problem	*Average*	*Weighted Average*
Auto	0.060	0.061
Autos-horse	0.047	0.047
Cholesterol	0.097	0.098
Cpu	0.055	0.053
Housing	0.075	0.071
Servo	0.097	0.095
Mean	0.072	0.071

5 Conclusions

The surge of connectionist techniques, such as *MLPs*, has created new exciting possibilities for the field of *Machine Learning*. On the other hand, considering ensembles of learning models to improve its accuracy has been a focus of attention by the research community.

In this work, an evaluation of ensembles of *MLPs*, in supervised learning tasks (e.g., classification and regression) is followed, being tested several combination schemes, under quite simple construction heuristics.

The results obtained show that even simple ensembles are able to gain improved performances over the use of single *ANNs*. In terms of combination methods, the simple averaging of the *ANN*'s outputs leads to the best results, being *Voting*, a widely used approach, quite inadequate.

The proposed *HT* construction method, considering a set of models of distinct complexity, together with a *Weighted Average* combination method, based on the accuracy measure over a validation set, obtains the best overall performance, Furthermore, considerable gains were achieved when compared to the use of a single *ANN* (specially for the tasks which require high nonlinear models), even when the latter is optimized by a model selection stage. The *HT* strategy does not require this previous step, thus implying an equivalent computational effort.

In future work, there are several promising directions. First, distinct methods for ensemble construction should be considered (e.g., *bagging* or *boosting*). In what concerns combinations methods, more elaborate strategies should be attempted (e.g., *Weighted Voting*, by disregarding *ANNs* with low outputs).

Another interesting field is based in the use of *Evolutionary Algorithms* [8]. These can be used to evolve populations of *MLPs*, which constitute the ensemble. By considering elaborated fitness functions and combination methods, specialization could be attained [7].

Acknowledgements

The authors would like to acknowledge the support from FCT, given under the project POSI/ROBO/43904/2002, which is partially funded by FEDER.

References

[1] C. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

[2] C. Blake and C. Merz. UCI Repository of Machine Learning Databases, 1998.

[3] T. Dietterich. Machine Learning Research: Four Current Directions. *AI Magazine*, 18(4):97–136, 1997.

[4] S. Gallant. *Neural Network Learning and Expert Systems*. MIT Press, Cambridge, USA, 1995.

[5] R. Kohavi. A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. In *Proceedings of the International*

Table 6: *HT* Ensemble Classification Results ($PCCE$).

Problem	*Voting*	*Winner-takes-all*	*Average*	*Weighted Average*
Balance	87.7	94.0	95.7	96.2
Bupa	68.1	67.9	68.5	70.4
Car	90.7	98.1	98.3	98.7
Credit	84.5	85.2	84.7	85.0
Glass	63.1	65.2	67.8	69.5
Ionosphere	85.2	89.5	92.3	91.9
Pima	77.0	75.5	75.5	76.1
Sonar	72.5	81.0	80.9	81.7
Yeast	58.2	59.5	59.9	60.1
Mean	76.3	79.5	80.4	81.1

Joint Conference on Artificial Intelligence (IJCAI), Montreal, Quebec, Canada, August 1995.

[6] S. Lee, J. Ahn, and S. Cho. Exploiting Diversity of Neural Ensembles with Speciated Evolution. In *Proceedings of IEEE/INNS International Joint Conference on Neural Networks*, August 2001.

[7] Y. Liu, X. Yao, and T. Higuchi. Evolutionary Ensembles with Negative Correlation Learning. *IEEE Transactions on Evolutionary Computation*, 4(4):380–387, 2000.

[8] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs.* Springer-Verlag, USA, third edition, 1996.

[9] D. Michie, D. Spiegelhalter, and C. Taylor. *Machine Learning, Neural and Statistical Classification.* NY: Ellis Horwood, 1994.

[10] L. Prechelt. *Early Stopping – but when?* In: *Neural Networks: Tricks of the trade*, Springer Verlag, Heidelberg, 1998.

[11] M. Riedmiller. Supervised Learning in Multilayer Perceptrons - from Backpropagation to Adaptive Learning Techniques. *Computer Standards and Interfaces*, 16, 1994.