Fuzzy Heterogeneous Neural Networks for Signal Forecasting

Lluís Belanche, Julio J. Valdés and René Alquézar
Dept. Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya
{belanche, valdes, alquezar}@lsi.upc.es
Barcelona, Spain

Abstract

Fuzzy heterogeneous neural networks are recently introduced models based on neurons accepting heterogeneous inputs (i.e. mixtures of numerical and non-numerical information possibly with missing data) with either crisp or imprecise character, which can be coupled with classical neurons. This paper compares the effectiveness of this kind of networks with time-delay and recurrent architectures that use classical neuron models and training algorithms in a signal forecasting problem, in the context of finding models of the central nervous system controllers.

1 Introduction

A fuzzy heterogeneous neuron is defined as a mapping $h : \hat{\mathcal{H}}^n \rightarrow \mathcal{R}_{out} \subseteq \mathbb{R}$, satisfying $h(\phi) = 0$ (ϕ is the empty set). Here $\mathbb{R}$ denotes the reals and $\hat{\mathcal{H}}^n$ is a cartesian product of an arbitrary number of *source sets*. Source sets may be families of extended reals $\hat{\mathcal{R}} = \mathbb{R} \cup \{\mathcal{X}\}$, extended fuzzy sets $\hat{\mathcal{F}}_i = \mathcal{F}_i \cup \{\mathcal{X}\}$, and extended finite sets of the form $\hat{\mathcal{O}}_i = \mathcal{O}_i \cup \{\mathcal{X}\}$, $\hat{\mathcal{M}}_i = \mathcal{M}_i \cup \{\mathcal{X}\}$, where each of the $\mathcal{O}_i$ has a full order relation, while the $\mathcal{M}_i$ have not. In all cases, the special symbol $\mathcal{X}$ denotes the unknown element (missing information) and it behaves as an incomparable element w.r.t. any ordering relation. According to this definition, neuron inputs are possibly empty arbitrary tuples, composed by n elements among which there might be reals, fuzzy sets, ordinals, nominals and missing data [1], [2]. Heterogeneous neurons are classified according to the nature of their image set (which need not be restricted to a subset of the reals). In the present study, since the image set is given by $\mathcal{R}_{out}$ the model is of the *real kind*, which is easily coupled with other classical neuron models (i.e. accepting only real inputs), thus leading to hybrid networks in a straightforward way. These networks have been used successfully in classification problems reported elsewhere [1], but their potential of application in other fields was not yet assessed experimentally. The purpose of this paper is to explore further the performance of fuzzy heterogeneous networks (in hybrid architectures) in a signal forecasting task concerning the central nervous system control.

The paper is organized as follows. Section 2 reviews the concept of fuzzy heterogeneous neurons and their use in configuring hybrid networks, while section 3 describes the problem at hand, covering also the different neural paradigms compared to the one presented, the experiment setup and the obtained results. Finally, section 4 presents the conclusions.

2 Heterogeneous Neural Networks

A particular class of heterogeneous networks (HNNs) is constructed by considering h as the composition of two mappings, that is, $h = f \circ s$, such that $s : \hat{\mathcal{H}}^n \to \mathcal{R}' \subseteq \mathbb{R}$ and $f : \mathcal{R}' \to \mathcal{R}_{out} \subseteq \mathbb{R}$. The mapping h can be considered as a n-ary function parameterized by a n-ary tuple $\hat{\vec{w}} \in \hat{\mathcal{H}}^n$ representing neuron's weights, i.e. $h(\hat{\vec{x}}, \hat{\vec{w}}) = f(s(\hat{\vec{x}}, \hat{\vec{w}}))$. In particular, function s represents a *similarity* and f a squashing non-linear function with its image in $[0, 1]$. Accordingly, the neuron is sensitive to the degree of similarity between its inputs –composed in general by a mixture of continuous and discrete quantities possibly with missing data– and its weights. More precisely, s is understood as a *similarity index*, or proximity relation (transitivity considerations are put aside). That is, a binary, reflexive and symmetric function $s(x, y)$ with image on $[0, 1]$ such that $s(x, x) = 1$ (strong reflexivity). The concrete instance of the model under study in the present paper uses as aggregation function a *Gower-like* similarity index in which the computation for heterogeneous entities is constructed as a weighted combination of partial similarities over subsets of variables. This coefficient has its values in the real interval $[0, 1]$ and for any two objects i, j given by tuples of cardinality n, is given by $s_{ij} = \frac{\sum_{k=1}^{n} g_{ijk}\, \delta_{ijk}}{\sum_{k=1}^{n} \delta_{ijk}}$ where g_{ijk} is a similarity *score* for objects i, j according to their value for variable k. These scores are in the interval $[0, 1]$ and are computed according to different schemes for numeric and qualitative variables. The factor δ_{ijk} is a binary function expressing whether objects i, j are comparable or not according to their values w.r.t. variable k. Gower's original definitions [3] for real-valued and discrete variables are kept, although other similarity functions are possible. For variables representing fuzzy sets, similarity relations from the point of view of fuzzy theory have been defined elsewhere [4] and different choices are possible. In our case, if $\mathcal{F}_i$ is an arbitrary family of fuzzy sets from the source set, and $\tilde{A}$, $\tilde{B}$ are two fuzzy sets such that $\tilde{A}, \tilde{B} \in \mathcal{F}_i$, the following similarity relation is used:

$$g(\tilde{A}, \tilde{B}) = \sup_x \, (\mu_{\tilde{A} \cap \tilde{B}}(x)) \text{ where } \mu_{\tilde{A} \cap \tilde{B}}(x) = min(\mu_{\tilde{A}}(x), \mu_{\tilde{B}}(x)).$$

For the activation function, a modified version of the classical logistic is used, which is an automorphism of the real interval $[0, 1]$.

$$f(x, p) = \begin{cases} \frac{-p}{(x-0.5)-a(p)} - a(p) & \text{if } x \leq 0.5 \\ \frac{-p}{(x-0.5)+a(p)} + a(p) + 1 & \text{otherwise} \end{cases}$$

where $a(p)$ is an auxiliary function given by $a(p) = \frac{-0.5+\sqrt{0.5^2+4*p}}{2}$ and p is a real-valued parameter controlling the curvature, set in the experiments to 0.1. The general training procedure for the HNN is based on genetic algorithms, since the heterogeneity of the variables involved and the non-differentiability of the similarity function prevent the use of gradient-based techniques [1].

3 A case study in signal forecasting

3.1 Problem description

The problem studied consists of forecasting the output signals of the Central Nervous System (CNS) controllers of the hemodynamical system. This system, together with the CNS control, form the cardiovascular system. The CNS generates the regulating signals for the blood vessels and the heart, and it is composed of five controllers: *heart rate*, *peripheral resistance*, *myocardial contractility*, *venous tone* and *coronary resistance*. All of these controllers are single-input/single-output (SISO) systems driven by the same input variable, namely the *carotid sinus pressure*. Whereas the structure and functioning of the hemodynamical system are well known and a number of quantitative models, mostly based on differential equations, have been developed, the functioning of the CNS control is of high complexity and still not completely understood. Although some differential equation models for the CNS have been postulated, these models are not accurate enough, and therefore, the use of other modeling approaches like neural networks or qualitative methodologies may offer an interesting alternative for capturing the behaviour of the CNS control [5].

3.2 Neural approaches used in the experiments

Two types of neural network architectures can be used for learning tasks involving a dynamic input/output relation, such as prediction and temporal association: *time-delay neural networks* (TDNNs) and *recurrent neural networks* (RNNs) [6]. The HNN model is to be compared to a RNN and two different TDNN models described below.

3.2.1 Time-delay neural networks

If some fixed-length segment of the most recent input values is considered enough to perform the task successfully, then a temporal sequence can be turned into a set of spatial patterns on the input layer of a multi-layer feedforward net trained with an appropriate algorithm such as backpropagation. These architectures are called TDNNs, since several values from an external signal are presented simultaneously at the network input using a moving window (shift register or tapped delay line) [6]. A main advantage of TDNNs in front of RNNs is their lower cost of training, which is very important in case of long training sequences. TDNNs have been applied extensively in recent years to different tasks, in particular to prediction and system modeling [7]. In the case of learning a SISO controller, with an input real-valued variable $x(t)$ and an output real-valued variable $y(t)$, the output layer of a TDNN consists of a single output unit that will provide the predicted value for $y(t)$, whereas the input layer holds some previous values $y(t-1), \ldots, y(t-m)$ and some recent values of the input variable $x(t), x(t-1), \ldots, x(t-p)$, from which the value $y(t)$ could be estimated (i.e. a total number of $m+p+1$ input units). Additionally, a hidden layer of N units (to be determined) is required. In the present study, two different TDNN approaches that differ in the training method have been tested: a standard backpropagation algorithm (TDNN-BP) using sinusoidal units, and a hybrid procedure composed of repeated cycles of simulated annealing coupled with a conjugate gradient algorithm (TDNN-AC) [8]. For the latter, hyperbolic tangent units form the hidden layer whereas the output layer is composed by a linear neuron. It should be noted that the HNN model as used here (TD-HNN) can be viewed as a TDNN that incorporates heterogeneous neurons and is trained by means of genetic algorithms.

3.2.2 Recurrent neural networks

In recent years, several RNN architectures including feedback connections, together with their associated training algorithms, have been devised to cope naturally with the learning and computation of tasks involving sequences and time series [6]. A type of RNN that has been proven useful in grammatical inference through next-symbol prediction is the first-order augmented single-layer RNN (or ASLRNN) [9], which is similar to Elman's SRN [10] except that is trained by a true gradient-descent method, using backpropagation for the feed-forward output layer and Schmidhuber's RTRL algorithm [11] for the fully-connected recurrent hidden layer. Although the use of sigmoidal activation functions has been common in RNNs, a better learning performance can be achieved using other activation functions such as the sine function [9]. Such networks with sinusoidal units can be seen as generalized discrete Fourier series with adjustable frequencies [7]. Hence, the ASLRNN model used here was built up with sinusoidal units.

3.3 Experiment setup

The data used in the training and test phases of the experiments came from a single subject. Five CNS control models, namely, *heart rate*, *peripheral resistance*, *myocardial contractility*, *venous tone* and *coronary resistance*, were inferred for this subject by means of the neural approaches aforementioned. The input and output signals of the CNS controllers were recorded with a sampling rate of 0.12 seconds from simulations of a purely differential equation model. This model had been tuned to represent a specific patient suffering from coronary arterial obstruction, by making the four *different* physiological variables (right auricular pressure, aortic pressure, coronary blood flow, and heart rate) of the simulation model agree with the measurement data taken from the patient. The training set was composed of 1,500 data points for each controller, whereas six data sets not used in the training process (600 points each) were used as forecasting targets, containing signals that represent specific morphologies. The HNN and the TDNN architectures were fixed to include 1 output unit, 8 hidden units, and 7 input units, corresponding to the values $x(t)$, $x(t-1)$, $x(t-2)$, $x(t-3)$, $y(t-1)$, $y(t-2)$ and $y(t-3)$, where $x(t)$ denotes the current value of the input variable and $y(t-1)$ denotes the value of the controller output in the previous time step. All inputs to the HNN were treated as fuzzy sets and the similarity relation given in Section 2 was used. The first-order ASLRNN architecture also included 1 output and 8 hidden units, but just

2 input units, corresponding to the values $x(t)$ and $y(t-1)$, though in this case the hidden layer incorporated additional weights for the feed-back connections.

In the testing process, the normalized mean square error (in percentage) between the predicted output value, $\hat{y}(t)$, and the controller output, $y(t)$, was used to determine the quality of each of the inferred models. This error is given by $MSE = \frac{E[(y(t)-\hat{y}(t))^2]}{y_{\text{var}}} \cdot 100\%$ where y_{var} denotes the variance of $y(t)$.

For each CNS controller and neural approach three different training trials were run using a different random weight initialization. The HNN was trained using a standard genetic algorithm with the following characteristics: binary-coded values, probability of crossover: 0.6, probability of mutation: 0.01, number of individuals: 100, linear scaling with factor: 1.5, selection mechanism: tournament. The algorithm stopped when no improvement was found for the last 1,000 generations (typical values were about 5,000). On the other hand, the TDNN-BP and ASLRNN nets were allotted 3,000 epochs using a small learning rate of $\alpha = 0.025$ to allow a smooth minimization trajectory. These parameters were tuned after some preliminary tests. For each run, the network yielding the smallest MSE error on the training set during learning was taken as the controller model. The TDNN-AC was trained in only one run and the process was stopped when a reasonable error was attained.

3.4 Results

The nets resulting from the training phase were applied to the training set and to the six test data sets associated with each controller. The normalized MSE errors for these sets were calculated, together with their averages for the different training runs and test sets. The summary of the errors obtained by the different neural approaches is displayed in Table 1.

	TD-HNN		**TDNN-BP**		**TDNN-AC**		**ASLRNN**	
	Train.	Test	Train.	Test	Train.	Test	Train.	Test
HRC	0.11%	0.18%	1.15%	1.52%	0.15%	0.13%	1.63%	1.91%
PRC	0.09%	0.12%	0.94%	1.27%	0.26%	0.14%	0.84%	1.10%
MCC	0.03%	0.06%	0.81%	1.33%	0.09%	0.08%	0.71%	1.18%
VTC	0.03%	0.06%	0.81%	1.33%	0.09%	0.08%	0.71%	1.18%
CRC	0.10%	0.11%	0.47%	0.66%	0.03%	0.04%	0.41%	0.53%
mean	0.07%	0.11%	0.84%	1.22%	0.12%	0.09%	0.86%	1.18%

Table 1: Average normalized MSE errors for the training sets (left) and test sets (right) of the CNS controller models inferred by each neural approach.

It is interesting to observe the excellent results yielded by the models inferred by both the HNN and the TDNN-AC, especially as compared to the TDNN-BP and ASLRNN, which showed an almost identical prediction performance, possibly caused by a short depth of temporal dependencies in the modeled system (i.e. all relevant past information could be included in the moving window that selects the inputs of a TDNN).

4 Conclusions

Heterogeneous neural networks have been successfully tested in a signal forecasting task (learning central nervous system controllers). The learning and generalization performance of HNNs are comparable to that of TDNNs trained with sophisticated optimization algorithms and better than that of TDNNs trained with backpropagation and RNNs trained with a true gradient-descent algorithm. However, further experiments addressing this kind of problems should be carried out towards a better understanding of their capabilities.

References

[1] Valdés, J.J., García, R., A model for heterogeneous neurons and its use in configuring neural networks for classification problems, *Procs. of IWANN'97, Intl. World Conf. on Artificial and Natural Neural Networks.* Lecture Notes in Computer Science 1240, Springer-Verlag, pp. 237-246.

[2] Belanche, Ll., Valdés, J.J., Using Fuzzy Heterogeneous Neural Networks to Learn a Model of the Central Nervous System Control. Procs. of *EUFIT'98, 6th European Congress on Intelligent Techniques and Soft Computing.* Aachen, Germany.

[3] Gower, J.C., A general coefficient of similarity and some of its properties, *Biometrics*, 27, pp. 857-871, 1971.

[4] Dubois D., Prade H., Esteva F., García P., Godo L., López de Mántaras, R., Fuzzy set modeling in case-based reasoning. Intl. Journal of Intelligent Systems, 1997 (to appear).

[5] J. Cueva, R. Alquézar and A. Nebot, "Experimental comparison of fuzzy and neural network techniques in learning models of the central nervous system control", *Proc. of EUFIT'97, 5th European Congress on Intell. Tech. and Soft Comput.*, Aachen, Germany, pp.1014-1018, September 1997.

[6] J. Hertz, A. Krogh and R.G. Palmer, *Introduction to the Theory of Neural Computation*, Addison-Wesley, Redwood City, 1991.

[7] A. Lapedes and R. Farber, *Nonlinear signal processing using neural networks: prediction and system modeling*, Tech. Rep. LA-UR-87-2662, Los Alamos National Laboratory, Los Alamos NM, 1987.

[8] Ackley, D. *A connectionist machine for genetic hillclimbing* Kluwer Acad. Press, 1987.

[9] J.M. Sopena and R. Alquézar, Improvement of learning in recurrent networks by substituting the sigmoid activation function. *ICANN'94, Proc. Int. Conf. Artif. Neural Networks*, Sorrento, Italy, Springer-Verlag, Vol.1, pp.417-420, 1994.

[10] J.L. Elman, Finding structure in time. *Cogn. Sci.* Vol. 14, pp.179-211, 1990.

[11] J. Schmidhuber, A fixed size storage $O(n^3)$ time complexity learning algorithm for fully recurrent continually running networks, *Neural Computation* Vol. 4, pp.243-248, 1992.