Engineering Letters, 14:2, EL_14_2_13 (Advance online publication: 16 May 2007)

Similarity-based Heterogeneous Neural Networks

Lluís A. Belanche Muñoz [*] Julio José Valdés Ramos[†]

Abstract

This research introduces a general class of functions serving as generalized *neuron models* to be used in artificial neural networks. They are cast in the common framework of computing a *similarity* function, a flexible definition of a neuron as a pattern recognizer. The similarity endows the model with a clear conceptual view and leads naturally to handle heterogeneous information, in the form of mixtures of continuous numbers (crisp or fuzzy), linguistic information and discrete quantities (ordinal, nominal and finite sets). Missing data are also explicitly considered. The absence of coding schemes and the precise computation attributed to the neurons makes the networks highly interpretable. The resulting *heterogeneous neural networks* are trained by means of a special-purpose genetic algorithm. The cooperative integration of different *soft computing* techniques (neural networks, evolutionary algorithms and fuzzy sets) makes these networks capable of learning from non-trivial data sets with a remarkable effectiveness, comparable or superior to that of classical models. This claim is demonstrated by a set of experiments on benchmarking real-world data sets.

Keywords: soft computing; neural networks; similarity measures; data heterogeneity; evolutionary algorithms

1 Introduction

Artificial Neural Networks (ANN) [6] are information processing structures evolved as an abstraction of known or assumed principles of how the brain might work. The network is said to *learn* when, as a result of exposure to examples, the parameters of the units are adapted to represent the information present in the examples in an optimal sense to be precised. The network relies upon the neuron model representation capacity as the cornerstone for a good approximation.

The strong points of ANNs are their appealing capacity to learn from examples, their distributed computation —which helps them tolerate partial failures to a certain extent— and the possibility, so often exploited, to use them as black-box models. This last characteristic is paradoxically one of the major weaknesses, given that in practice this autonomy of functioning involves no transfer of knowledge from or to the designer. With the exception of very specific architectures, the networks are forced to learn from scratch most of the times. The network *works* (in the sense that solves a problem to a certain satisfaction), but the weights convey information as high-dimensional real vectors whose meaning about the solution can be intricate. A significant part of the task is devoted to find structure in the data and transform it to a new *hidden* space (or space spanned by the hidden units) in such a way that the problem becomes easier (almost linearly separable in the case of the output layer). The internal workings of a neuron are thus obscure, because the weights have not been set to shape a previously defined (and considered adequate) similarity measure, but rather to adapt a general *physical measure* (like scalar product or Euclidean distance) to the problem at hand.

In practice the network has to discover the relations in the structure induced by the chosen coding scheme and find ways to *accommodate* the underlying similarity relationship (inherent in the training examples) to a fixed similarity computation. During the learning process, patterns seen as physically similar may have to be told apart and vice versa. In consequence, several layers may be needed for complex transformations, or a large amount of neurons per layer if we restrict the number of hidden layers to one or two, as is common proceeding. An increase in neurons leads to a corresponding growth in the number of free parameters, and these are less likely to be properly constrained by a limited size data set [6].

Besides all that, real-world data come from many different sources (continuous or discrete numerical pro-

[*]Dept. de Llenguatges i Sistemes Informàtics. Universitat Politècnica de Catalunya. Barcelona, Spain. e-mail: belanche@lsi.upc.edu

[†]National Research Council Institute for Information Technology, Ottawa, Canada. e-mail: Julio.Valdes@nrc-cnrc.gc.ca

cesses, symbolic information, etc.) and have their own peculiarities (vagueness, imprecision, incompleteness), and thus may require different treatments. For example, in the well-known UCI repository [24] over half of the problems contain *explicitly declared* nominal attributes, let alone other discrete types or fuzzy information, usually unreported. This heterogeneity is traditionally coped with by *preparing* the data using a number of methods. This preprocessing is not part of the original task and may involve an abrupt change in input dimension and distribution. The obtained high-dimensional patterns entail a lower data density, requiring stronger constraints on the problem solution [8].

This work deals with the development of new neuron models to be used as the basic computing elements in ANNs, where the notion of similarity between examples and parameters plays a central role. Conditions for the aggregation and composition of similarity measures are introduced, in which semantic information and missing data are explicitly considered. A number of measures to compute the similarity between non-standard data types are listed. The notion of a heterogeneous space is then defined as a cartesian product of single spaces of mixed variables. An heterogeneous neuron or H-neuron is then defined as a mapping from elements in a heterogeneous (input) space to an heterogeneous (output) space. An S-neuron is defined as an H-neuron whose mapping is a similarity measure. This leads to the development of general neuron models working in heterogeneous spaces.

The framework is flexible, offering means for the design of neuron models having certain desirable properties. The definition of these neuron models, whose computation as a pattern recognizer is explicitly defined as a similarity measure, and where different data sources are directly and specifically treated, permits a natural extension of existing approaches. In particular, one of the most widely used neuron model, that of Radial Basis Function (RBF) networks, can be easily cast as a form of similarity measure. Most important, it opens a way for devising new ones. A particularly useful class of S-neurons of the *real kind* (models for which the codomain is a subset of the reals) is created based on Gower's similarity index [17]. The resulting network is then tested experimentally, showing a remarkable performance in the presence of heterogeneity and missing values in the data.

This paper is organized as follows. In (§2) we review relevant related work. In (§3) we summarize basic material on similarity measures, their definition and construction, out of which we develop a general framework for designing neuron models of the kind described in this work. In (§4) we present performance results on a set of experiments to illustrate the use and validity of the approach. The paper ends with some conclusions on the knowledge gained.

2 Related work

In classical artificial intelligence (AI) systems, the notion of *similarity* appears mainly in case-based reasoning (CBR), where learning by analogy and instance-based learning fuse. The popularity of CBR systems has boosted its use and signaled its key role in cognitive tasks [1]. As a matter of fact, some form of similarity is inherent in the majority of AI approaches. There is an agreement in that it is easier to respond intelligently to a stimulus if previous responses made under *similar* circumstances can be recalled. Similarity coefficients have had a long and successful history in the literature of cluster analysis and data clustering algorithms [7].

The origins of learning by similarity in artificial neural systems can be traced back to the pioneering works of Hebb and his now classic book [19]. He postulated that the functionality of ANNs had to be determined by the strengths of the neural connections. This functionality should be adjusted to *increase* the likeliness of getting a *similar* response to *similar* inputs in the future, provided the elicited response is the desired one. In the opposite situation, the weights should be adjusted to decrease this likeliness. However, this inspiring idea has not been fully exploited in the prevalent neuron models.

There have been few attempts to incorporate heterogeneous information into the workings of an artificial neuron in a principled way. The main contribution is perhaps encountered in heterogeneous distance proposals, where separate distance calculations are used for nominal and continuous variables [31], where the authors present an extension of the RBF model to nominal quantities and missing values. The RBF network is used in its original interpolative definition (i.e., there is one hidden node for each example in the training set). The extension consists in the use of the Value Difference Metric [28] for nominal distance computation. An Euclidean metric is used to account for the partial distances. Missing values are handled with a *pessimistic* semantics, defining the distance involving a missing value to be the maximum possible distance (actually a value of one). The results point to an increase in performance w.r.t. standard Euclidean distance when the amount of nominal information is significant, giving support to a deeper and more precise formulation of neuron models as similarity computing devices.

We briefly review now traditional ways of representing non-standard information in ANNs [27, 6, 14].

Nominal variables are coded using a 1-out-of-k representation, being k the number of values. This introduces the rows of the $I_{k \times k}$ identity matrix in the training set, leading to an *structured* increase of model input variables, which translates in a notable increment in the number of parameters. Part of the network task is to discover that all these inputs are strongly related and, as a matter of fact, that they represent a single one.

Ordinal variables correspond to discrete and usually finite sets of values wherein an ordering has been defined. They are more than often treated as real-valued, mapped equidistantly on an arbitrary real interval. This imposes a continuum where there is not (e.g., stating that there are infinite possibilities between having two or three children). Further, since these variables need not represent numerical entities, this coding is assuring that "Wednesday" is three times "Monday", or that "D" plus "E" is "J".

A second possibility is to code them using a *thermometer*: being k the number of ordered values, k new *binary* inputs are then created. To represent value i, for $1 \leq i \leq k$, the leftmost $1, \ldots, i$ units are on, and the remaining $i+1, \ldots, k$ off. This representation is probably a worse choice. If the number of possibilities to be represented is not small the increase in variables may be simply not affordable. Further, since the "order" imposed by this coding is only apparent: an arbitrary permutation of the inputs and training data columns will result in an identical training set for the network.

Missing information is an old issue in statistical analysis [20], and very common in Engineering and Medicine, where many variables come from on-line sensors or device measurements. There are many causes for the absence of a value: technical limitations, temporal malfunctioning, costly measurements, invalid measurements, data lost during transcription, transmission or storage, reluctance to supply a value, different time intervals in measurements, etc. Missing information is difficult to handle, specially when the lost parts are of significant size. There are basically two ways of dealing with missing data: completing the item description in a hopefully optimal way, to be defined (i.e., "fill in the holes") or extend the methods to be able to work with incomplete descriptions. Simply discarding the involved data can not be considered as a "method" and is also frustrating because of the lost effort in collecting the information. It be also dangerous if the missing pieces are related with the non-missing ones (e.g. in time series). The vast majority of methods in the literature of ANNs belong to the first kind, including:

- Fill in by a *constant* (e.g. 0 or a *random* value). A fixed constant value can only harm the distribution and is rarely considered, while using the mean, median or nearest-neighbour value is in general not a good idea. It may well result in a significant alteration of the distribution, since it ignores the covariance in the observed data and the likely multimodality of the distribution.

- Set up a regression model for any variable having missing values over the present variables and use the obtained regression function to fill in the holes. This tends to underestimate the covariance in the data, as the filled-in points will fall along the regression line [6].

- Create a variable which is zero if the involved variable is present, and is one otherwise. This method is intuitive but not completely satisfactory, because the problem of inputting a value still remains and the new input does not represent any specific entity.

- Use the values obtained from a maximum-likelihood method, such as the Expectation-Maximization (EM) algorithm [13]. This algorithm can estimate the parameters of a mixture model (e.g. of Gaussians) by maximizing their likelihood, even in presence of missing data [16]. This involves modeling the input distribution itself, which can be a difficult and time-consuming task.

The problem with missing data is that we never know if all the efforts devoted to their estimation revert, in practice, in better-behaved data. The reviewed methods preprocess the data to make it acceptable by models that otherwise would not accept them. In the case of missing values, the data are *completed* since the available neural methods only admit complete data sets. An alternative solution is simply to *ignore* what is not known (and nothing more), in such a way that it does not affect the treatment of the known pieces of information.

Vagueness, imprecision and other **sources of uncertainty** are considerations usually put aside in the ANN paradigm. However, many variables in learning processes are likely to bear some form of uncertainty. In Engineering, for example, on-line sensors get old with time and continuous use. It is likely that a continuous variable taking on a value (say, 5.1) expresses reasonable approximation for a precise value that is unavailable. Sometimes the known value takes an interval form: "between 5.0 and 5.5", so that any transformation to a real value will result in a loss of information. A more common situation is the absence of numerical knowledge. For example, consider the value `fairly tall` for the variable *height*. Fuzzy Systems are comfortable with this, but for ANNs this is real trouble.

To sum up, in general the reviewed methods are prone to cause a distorsion (*e.g.*, by introducing a significant amount of zero input values), result in loss of information or increment the number of parameters to be estimated, among others.

3 Similarity measures

3.1 The definition of similarity

Let us represent objects belonging to a space $X \neq \emptyset$ (about which the only assumption is the existence of an equality relation) as vectors $\mathbf{x}_i$ of n elements, where each element x_{ij} represents the value of a particular feature (descriptive variable) a_j for object i, from a predefined set A of features. A *similarity measure* is a unique number expressing how "like" two given objects are, given only the features in A. Let us denote by s_{ij} the similarity between $\mathbf{x}_i$ and $\mathbf{x}_j$, that is, $s : X \times X \rightarrow \mathbb{R}^+ \cup \{0\}$ and $s_{ij} = s(\mathbf{x}_i, \mathbf{x}_j)$. The meaning of $s_{ij} > s_{ik}$ is that object i is more similar to object j than is to object k.

Definition 3.1 *A similarity measure in X fulfills the following properties:*

1. Non-negativity. $s_{ij} \geq 0 \quad \forall \mathbf{x}_i, \mathbf{x}_j \in X$
2. Symmetry. $s_{ij} = s_{ji} \quad \forall \mathbf{x}_i, \mathbf{x}_j \in X$
3. Boundedness. *There is a maximum attained similarity:* $\exists s_{max} \in \mathbb{R}^+ : \; s_{ij} \leq s_{max} \quad \forall \mathbf{x}_i, \mathbf{x}_j \in X$.
4. Minimality *(Reflexivity in the strong sense).* $s_{ij} = s_{max} \Longleftrightarrow \mathbf{x}_i = \mathbf{x}_j \quad \forall \mathbf{x}_i, \mathbf{x}_j \in X$

3.2 Similarity aggregation

The way different similarities should be combined to give an overall score is an elusive topic [18]. The aggregation step fulfills also a *semantic* role, which is very important as a means to express functional or psychological aspects of similarity. The following is a definition of similarity aggregation through a compilation of desirable properties. Note that it is made easier by the fact that the aggregated values are, by definition, bounded.

For $z \in \mathbb{R}, n \in \mathbb{N}^+$ and an operator T, let $T^n[z] = \begin{cases} z & \text{if } n = 1 \\ T(\underbrace{z, z, \ldots, z}_{n \; times}) & \text{if } n > 1 \end{cases}$

Definition 3.2 (Θ aggregation) *Consider a collection of $n_0 \geq 1$ real quantities, grouped by convenience as a vector $\mathbf{s}_0 = \{s_1, s_2, \ldots, s_{n_0}\}$, where $s_i \in [0, s_{max}] \cup \{\mathcal{X}\}$, with $s_{max} > 0$, where the symbol $\mathcal{X}$ denotes a missing element, for which only equality is defined. Let $F_{\mathcal{X}} : ([0, s_{max}] \cup \{\mathcal{X}\})^{n_0} \rightarrow ([0, s_{max}])^n$, with $n \leq n_0$ be a filter operator that returns a vector without the missing elements, in the original order. Let $\mathbf{s} = F_{\mathcal{X}}(\mathbf{s}_0)$. A similarity aggregation is a function $\Theta : ([0, s_{max}] \cup \{\mathcal{X}\})^{n_0} \rightarrow [0, s_{max}] \cup \{\mathcal{X}\}$ fulfilling:*

Consistency.

$$\Theta(\mathbf{s}) = 0 \Rightarrow \forall i : \; 1 \leq i \leq n : \; s_i = 0$$

and

$$\Theta(\mathbf{s}) = s_{max} \Rightarrow \forall i : \; 1 \leq i \leq n : \; s_i = s_{max}$$

Symmetry. $\Theta(\tau(\mathbf{s})) = \Theta(\mathbf{s})$ *for any* $\tau(\mathbf{s})$ *permutation of* $\mathbf{s}$.

Monotonicity. Let $\mathbf{s}'$ *be a second similarity judgement, then the predicate*

$$\exists i : 1 \leq i \leq n : (s_i > s'_i \wedge \forall j \neq i : s_j \geq s'_j)$$

implies $\Theta(\mathbf{s}) > \Theta(\mathbf{s}')$

Idempotency. For an arbitrary s_i, $\Theta^n[s_i] = s_i, \forall n \in \mathbb{N}^+$. *Note this specifically includes the boundary values* $\Theta^n[s_{max}] = s_{max}$ *and* $\Theta^n[0] = 0$, *the opposite of Consistency.*

Continuity. *Θ should be continuous in all its arguments so the reactions to small changes are not "jumpy". This property ensures that all possible values of $\Theta(\mathbf{s})$ can in principle be generated.*

A number of additional properties are easy consequences of this definition:

Stability. $s_i^{n+1}[\Theta] = s_i^n[\Theta], \forall n \in \mathbb{N}^+$. This property is directly implied by idempotency.

Cancellation. Denoting by $(\Theta)_k(\cdot;\mathbf{s})$ the operator $\Theta(\mathbf{s})$ with all elements of $\mathbf{s}$ frozen except s_k, let $\mathbf{s}'$ be equal to $\mathbf{s}$ for all $i \neq k$. Then, $(\Theta)_k(s_k;\mathbf{s}) = (\Theta)_k(s'_k;\mathbf{s}') \Rightarrow \mathbf{s} = \mathbf{s}'$. This property is derived from monotonicity (since $(\Theta)_k(\cdot;\mathbf{s})$ is inversible for all k).

Compensativeness. $\min_i s_i \leq \Theta(\mathbf{s}) \leq \max_i s_i$. This property comes from monotonicity and idempotency.

Notice that *weighted* aggregations could be conceivable, provided they are compliant with the above conditions. These defining properties or axioms are a way of adopting a *specific* semantics, but others could be possible. In particular, the above properties express that:

1. Even small contributions can only *add* something in favour for the overall measure;
2. The aggregation of a number n of equal values yields that same value, regardless of n;
3. High values in specific features are balanced by low values in other features (and vice versa);
4. The eventually missing pieces are regarded as *ignorance* and do not contribute in favour nor against the overall measure.

Proposition 3.1 *Given $\mathbf{s} = \{s_1, \ldots, s_n\}$ similarities s_i for objects in X_i, a new measure obtained from any such aggregation operator $\Theta(\mathbf{s})$ is a similarity measure in $X = X_1 \times X_2 \times \ldots \times X_n$.*

In order to make Definition (3.2) clearer, we provide with a family of aggregators.

Proposition 3.2 *The following is a valid family of similarity aggregations:*

$$\Theta(\mathbf{s}) = f^{-1}\left(\frac{1}{n}\sum_{i=1}^{n} f(s_i)\right) \qquad (1)$$

where f is a strictly increasing and continuous function such that $f(0) = 0$ and $f(s_{max}) = s_{max}$ (i.e., a bijection in $[0, s_{max}]$).

A specific example within this family is the normalized *modulus*:

$$\Theta(\mathbf{s}) = \frac{1}{\sqrt[q]{n}}\left(\sum_{i=1}^{n}(s_i)^q\right)^{\frac{1}{q}}, \ q \geq 1 \in \mathbb{R}, \qquad (2)$$

obtained by taking $f(z) = z^q$. This is a simple and useful measure, general enough to be applicable in many situations. It is also useful to introduce a class of functions that maintain the defining similarity properties when composed to a similarity measure.

Definition 3.3 (Similarity keeping) *A function $\check{s} : [0, s_{max}] \to [0, \check{s}_{max}]$ that is strictly increasing and continuous, and fulfills $\check{s}(0) = 0, \check{s}(s_{max}) = \check{s}_{max} > 0$.*

It is not difficult to show that composition to a similarity measure yields a further similarity measure. These functions are useful to act as activation functions in the context of neural networks. The final piece for constructing flexible similarity measures is a transformation function, to obtain a similarity out of metric distances.

Definition 3.4 (Similarity transforming) *A similarity transforming function $\hat{s}$ is a strictly decreasing monotonic and continuous function $\hat{s} : [0, +\infty) \to [0, s_{max}]$ such that $\hat{s}(0) = s_{max}$ and $\lim_{z \to +\infty} \hat{s}(z) = 0$. If z is bounded* [1] *, then $\hat{s} : [0, d_{max}] \to [0, s_{max}]$ and the limit condition is replaced by $\hat{s}(d_{max}) = 0$.*

Proposition 3.3 *Let d be a distance function defined in X and $\hat{s}$ a similarity transforming function. Then $s = \hat{s} \circ d$ is a similarity function in X.*

[1]A distance d is bounded in a set X if $d_{max} \equiv \sup_{x,y \in X} d(x, y)$ is finite.

Proposition 3.4 *Let $\check{s}_1, \check{s}_2$ be similarity keeping functions. Then $\check{s}_1 \circ \check{s}_2$ is a similarity keeping function. Let $\check{s}, \hat{s}$ be similarity keeping and transforming functions, respectively. Then $\check{s} \circ \hat{s}$ is a similarity transforming function.*

The following are example classes of $\hat{s}$ functions:

- $\hat{s}_0(z) = \left(1 - z^d\right)^\alpha$, $0 < d \leq 1, \alpha \geq 1, z \in [0, 1]$
- $\hat{s}_1(z) = \frac{1}{1+(az)^\alpha}$, $\alpha > 1, a > 0, z \in [0, \infty)$
- $\hat{s}_2(z) = e^{-(az)^\alpha}$, $\alpha > 0, a > 0, z \in [0, \infty)$
- $\hat{s}_3(z) = 2(1 - \sigma_\alpha(z))$, for $z \in [0, \infty), \alpha \in \mathbb{R}$, where $\sigma_\alpha(z) = (1 + exp(-\alpha z))^{-1}$ is the logistic function.

Some of them are well known in the context of RBF networks [26] or are commonly found in the data analysis literature [7]. In the proposed framework, the requirement for inclusion in the list is fulfillment of Definition (3.4). All the functions are set to yield $s_{max} = 1$.

3.3 Heterogeneous similarity measures

We consider in this work the following types of variables, for which corresponding similarity measures are listed.

Nominal (categorical) : non-numerical variable on which no order relation has been defined. It thus can be seen as having a *set* of values (finite or not).

Ordinal : variable (numerical or not) for which a linear order relation has been defined on a finite number of values, where each value has a crisp or precise sense.

Continuous : numerical and crisp variable where a linear order is defined on a continuum of values.

Set : variable whose values are classical sets.

Fuzzy Number : continuous variable whose values are fuzzy numbers (expressing *imprecision*).

Linguistic : ordinal variable whose values are general fuzzy sets (expressing *vagueness*).

The values of the last two variables can be obtained where appropriate by converting each crisp value (ordinal or continuous) into a fuzzy quantity. The following are basic similarity measures (in the sense of being universally applicable) fulfilling definition (3.1) and are defined such that $s_{max} = 1$.

3.3.1 Nominal variables

The most basic similarity measure for these variables is the overlap. Let $\mathcal{N}$ denote a *nominal* space:

$$s(x, y) = \begin{cases} 1 & \text{if } x = y \\ 0 & \text{if } x \neq y \end{cases}, \qquad x, y \in \mathcal{N} \tag{3}$$

3.3.2 Ordinal variables

Let $(\mathcal{O}, \preceq)$ be an ordered *ordinal* space and $x, y \in \mathcal{O}$, with an equality relation $x = y \equiv x \preceq y \land y \preceq x$ and strict precedence $x \prec y \equiv x \preceq y \land \neg(x = y)$ defined in the usual way. Let $\#$ denote set cardinality.

Define now $\eta : \mathcal{O} \to [1, m] \subset \mathbb{N}^+$ as $\eta(x) = \#\{x' \in \mathcal{O} : x' \prec x\} + 1$. Since the order $\preceq$ is linear, this is a bijection and $\eta(x') = \eta(x) + 1 \Leftrightarrow x' = succ(x)$. Since every subset of a metric space is metric, a distance in $\mathcal{O}$ can be set resorting to the standard metric in $\mathbb{R}$ if $\mathcal{O}$ is finite:

$$d(x, y) = \frac{|\eta(x) - \eta(y)|}{\#\mathcal{O} - 1} \in [0, 1] \qquad x, y \in \mathcal{O} \tag{4}$$

and, by selecting any $\hat{s}(z)$, $s(x, y) = \hat{s}(d(x, y))$ is a similarity measure, by Proposition (3.3). If $\mathcal{O}$ is infinite but countable (like $\mathbb{N}$ or $\mathbb{Z}$), then for $x, y \in \mathcal{O}$ one may use:

$$s(x, y) = \frac{1}{m - 1}\left(m\frac{min(\eta(x), \eta(y))}{max(\eta(x), \eta(y))} - 1\right) \in (0, 1] \tag{5}$$

3.3.3 Continuous variables

Let $x, y \in \Delta \subset \mathbb{R}$. Any metric in $\mathbb{R}$ is a metric in Δ. In particular, any positive power of the standard metric in $\mathbb{R}$, $d(x, y) = |x - y|^q, q > 0 \in \mathbb{R}$ is a metric in Δ. This family of distances assigns greater importance to smaller distances for decreasing $q < 1$ and less importance for increasing $q > 1$. In these conditions, any $s(x, y) = \hat{s}(d(x, y))$ is a similarity measure in Δ, by Proposition (3.3). These distances can be normalized by a factor obtained from a sample of the distribution. Specifically, weighting by the standard deviation is equivalent to an unweighted measure on standardized data. Self-normalizing measures like $d(x,y) = \frac{|x-y|}{|x+y|}$ (Canberra distance) are also possible.

3.3.4 Set variables

These can be regarded as generalized *nominal* variables whose values are not single elements of the space but subsets thereof. Let $\mathcal{S}$ be a *set* space and $x, y \in \mathcal{S}$. Provided $\mathcal{S}$ is finite (so that all its subsets are),

$$s(x,y) = \frac{\#\{x \cap y\}}{\#\{x \cup y\}}, \qquad x \neq \emptyset \vee y \neq \emptyset \tag{6}$$

with $s(\emptyset, \emptyset) = 1$, is a similarity measure in S, with $s_{max} = 1$. It reduces to (3) for singleton sets.

3.3.5 Fuzzy variables

The word *fuzzy* is viewed in this work taking the *epistemic* interpretation of a fuzzy set, i.e., as describing the vague observation of a theoretically crisp object. The imprecision stems from the indetermination about the specific value of a variable, in a situation where the set of all its possible values is known. Let $[\mathbb{R}] = \{[a,b] \subset \mathbb{R} \;/\; a,b \in \mathbb{R}, a \leq b\}$. We begin by defining a classic concept:

Definition 3.5 (Fuzzy number) *A fuzzy number in X (the reference set) is a convex and normalized fuzzy set F with piecewise continuous μ_F. Symmetry of μ_F is not required.*

Let $\mathbb{F}_n(X)$ be the (crisp) set of all the fuzzy numbers in X, where X is assumed a continuum. For variables representing fuzzy sets, similarity relations from the point of view of fuzzy set theory (similarity as a fuzzy equivalence relation) have been defined elsewhere [12]. The present case is a crisp relation between fuzzy entities. The *possibility* Π measures the co-occurrence of two vague propositions, with a value of one standing for absolute certainty [32]. For two fuzzy sets $\tilde{A}, \tilde{B} \in \mathbb{F}_n(X)$, it is defined as:

$$\Pi_{\tilde{A}}(\tilde{B}) = \sup_{u \in X} \{min\ (\mu_{\tilde{A}}(u),\, \mu_{\tilde{B}}(u))\}. \tag{7}$$

Proposition 3.5 *Given $\tilde{x}, \tilde{y} \in \mathcal{F} \subset \mathbb{F}_n(\Delta), \Delta \in [\mathbb{R}]$ ($\mathcal{F}$ a given family of fuzzy numbers), the function $s : \mathcal{F}, \mathcal{F} \to [0,1]$ defined as $s(\tilde{x}, \tilde{y}) = \Pi_{\tilde{x}}(\tilde{y})$ is a similarity measure in $\mathcal{F}$.*

The addition of linguistic features to express vagueness both in the input patterns *and* in the inner workings of the system itself can lead to new architectures with enhanced expressiveness and flexibility [25]. Since words are less precise than numbers, this approach enables the use of vague information, in cases where a precise quantity is unknown.

Definition 3.6 (Fuzzy quantity) *A fuzzy quantity in $\mathbb{R}$ is a normalized fuzzy set F with continuous μ_F with support in $[\mathbb{R}]$.*

Definition 3.7 (Fuzzy interval) *A fuzzy interval in $\mathbb{R}$ is a convex fuzzy quantity in $\mathbb{R}$. Symmetry of μ_F is a useful simplifying assumption although is not required.*

Definition 3.8 (Fuzzy p-quantity) *A fuzzy p-quantity in $\mathbb{R}$ is a fuzzy interval in $\mathbb{R}$, such that μ_F can be decomposed in three parts (from left to right): a function with at most one inflection point and a positive first derivative, a flat zone (null derivative) and a function with at most one inflection point and a negative first derivative. This is a kind of fuzzy interval of the LR-type [33], defined for convenience.*

A p-quantity with support in $[a,b]$ models the vague proposition "roughly between a and b". Among the several forms such fuzzy sets can take, the most popular are trapezoidal and bell-shaped. Note that a fuzzy number is a unimodal fuzzy p-quantity. These shapes are illustrated in Fig. (1). Let $\mathbb{F}_q(X)$ be the (crisp) set of all the fuzzy p-quantities in X. Given $\tilde{x}, \tilde{y} \in \mathbb{F}_q(\Delta), \Delta \in [\mathbb{R}]$, with respective support sets $\Delta_{\tilde{x}}, \Delta_{\tilde{y}} \subset \Delta$, a similarity can be obtained as:

$$s(\tilde{x}, \tilde{y}) = \frac{\int_{\Delta_{\tilde{x}} \cup \Delta_{\tilde{y}}} \mu_{\tilde{x} \cap \tilde{y}}(u) du}{\int_{\Delta_{\tilde{x}} \cup \Delta_{\tilde{y}}} \mu_{\tilde{x} \cup \tilde{y}}(u) du} \tag{8}$$

These variables are to be used whenever there is a knowledge that moves us to define linguistic terms instead of simple ordinals. If precise membership functions are known, these should be used.

3.4 Definition of heterogeneous spaces

An heterogeneous space, denoted $\hat{\mathcal{H}}^n$, is defined as the Cartesian product of a number n of *source* sets,

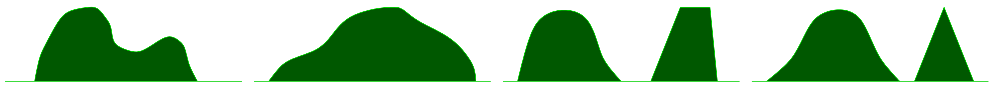


Figure 1: Several specific forms for a continuous fuzzy set in $\mathbb{R}$. From left to right: a fuzzy quantity, a fuzzy interval, two fuzzy p-quantities and two fuzzy numbers.

as follows. Let $X^{(\alpha)} = X_1 \times \cdots \times X_\alpha$ denote the Cartesian product with $X^{(0)} = \emptyset$. Consider now a collection of extended sets $\hat{\mathcal{R}}_1, \ldots, \hat{\mathcal{R}}_{n_r}$, where $\hat{\mathcal{R}}_i = \mathbb{R}_i \cup \{\mathcal{X}\}, \mathbb{R}_i \in [\mathbb{R}]$, and $1 \leq i \leq n_r$. Consider also collections of extended sets $\hat{\mathcal{O}}_1, \ldots, \hat{\mathcal{O}}_{n_o}$, with $\hat{\mathcal{O}}_i = \mathcal{O}_i \cup \{\mathcal{X}\}, 1 \leq i \leq n_o$, where each $\mathcal{O}_i$ is a finite and linearly ordered set, and extended sets $\hat{\mathcal{N}}_1, \ldots, \hat{\mathcal{N}}_{n_n}$, with $\hat{\mathcal{N}}_i = \mathcal{N}_i \cup \{\mathcal{X}\}, 1 \leq i \leq n_n$, where each $\mathcal{N}_i$ is a finite and unordered set. Consider now the collection of n_f extended families of fuzzy sets of the form $\hat{\mathcal{F}}_1, \ldots, \hat{\mathcal{F}}_{n_f}$, where $\hat{\mathcal{F}}_i = \mathcal{F}_i \cup \{\mathcal{X}\}, \mathcal{F}_i \subset \mathbb{F}_q(\Delta_i), \Delta_i \in [\mathbb{R}]$, and $1 \leq i \leq n_f$. Note that $\mathbb{F}_n(\Delta_i) \subset \mathbb{F}_q(\Delta_i)$. Finally, consider collections $\hat{\mathcal{S}}_1, \ldots, \hat{\mathcal{S}}_{n_s}$, with $\hat{\mathcal{S}}_i = \mathcal{S}_i \cup \{\mathcal{X}\}, 1 \leq i \leq n_s$, where each $\mathcal{S}_i$ are sets whose elements are sets. In all cases, the extension is given by the special symbol $\mathcal{X}$, which denotes the *unknown* element (missing information) for which only equality is defined and behaving as an *incomparable* element w.r.t. any ordering relation. In these conditions, set:

$$\hat{\mathcal{H}}^n \equiv \hat{\mathcal{R}}^{(n_r)} \times \hat{\mathcal{O}}^{(n_o)} \times \hat{\mathcal{N}}^{(n_n)} \times \hat{\mathcal{F}}^{(n_f)} \times \hat{\mathcal{S}}^{(n_s)} \quad (9)$$

with $n = n_r + n_f + n_o + n_n + n_s > 0$. According to (9), the elements of $\hat{\mathcal{H}}^n$ are general tuples of n components: real numbers, fuzzy numbers or quantities, ordinals, nominals, sets and missing values. We call such a structure an *heterogeneous space*.

Corollary 3.1 *Let Θ be an aggregation. Given a group of heterogeneous similarities $\mathbf{s} = \{\sigma_1, \ldots, \sigma_n\}$, where $\sigma_i : \hat{\mathcal{H}}_i, \hat{\mathcal{H}}_i \to [0, \sigma_{max}] \cup \{\mathcal{X}\}$, any measure s defined as $s(\mathbf{x}, \mathbf{y}) = \Theta(\sigma_1(x_1, y_1), \ldots, \sigma_n(x_n, y_n))$, is a similarity measure in $\hat{\mathcal{H}}^n = \hat{\mathcal{H}}_1 \times \cdots \times \hat{\mathcal{H}}_n$.*

To see this, make $X_i \equiv \hat{\mathcal{H}}_i$ and $s_i = \sigma_i(x_i, y_i)$ in Proposition (3.1).

3.5 Similarity-based neuron models

Definition 3.9 (H-neuron) *An heterogeneous neuron or H-neuron is any function of the class:*

$$\Gamma = \{\gamma_i : \hat{\mathcal{H}}^n \to \hat{\mathcal{H}} \mid \gamma_i(\mathbf{x}) = \gamma(\mathbf{x}, \alpha_i)), \ \alpha_i \in A\} \quad (10)$$

where $\mathbf{x} \in \hat{\mathcal{H}}^n$ and A is the neuron parameter space.

This definition accounts for neuron models performing general mappings from (heterogeneous) input and output spaces. In consequence, H-neurons can be classified according to the nature of their codomain (not necessarily restricted to a subset of the reals). In the present work, a model with a codomain given by $\hat{\mathcal{H}} \in \mathbb{R} \cup \{\mathcal{X}\}$ is set forth and called of the *real kind*.

Definition 3.10 (S-neuron) *A similarity-based neuron or S-neuron (of the real kind) is an H-neuron for which γ is a similarity in $\hat{\mathcal{H}}^n$ and $\hat{\mathcal{H}} = [0, s_{max}] \cup \{\mathcal{X}\}, s_{max} > 0 \in \mathbb{R}$.*

Accordingly, the S-neuron is sensitive to the degree of similarity between its heterogeneous inputs and weights. The reason to consider in the first place neuron models of the real kind is twofold. On the one hand, there is a natural coupling with classical neuron models (i.e. accepting only real-valued and complete inputs). On the other hand, in this work the emphasis is put on the definition of *scalar* similarities on bounded real intervals. Nonetheless, Definition (3.10) makes provision for neurons computing similarity functions in other codomains.

Definition 3.11 (HNN) *A feed-forward Heterogeneous Neural Network (or HNN) is a feed-forward network composed of $l \geq 1$ layers of S-neurons, and a linear output layer.*

A simple parsimony principle leads to consider feed-forward architectures of one hidden layer of S-neurons and a linear output layer. From a conceptual point of view, a second hidden layer would compute to what degree two patterns are similar by comparing their representations in hidden space. When the model is constructed as a composition of two mappings (a distance followed by a decreasing activation function $\hat{s}$), it is immediate to realize that the usual RBF model is a particular case of HNN in which $\hat{\mathcal{H}}^n = \mathbb{R}^n$ (hence $n = n_r$) and $l = 1$.

3.6 A worked example of S-neuron

A basic but very useful S-neuron can be devised using a Gower-like similarity index, well-known in the literature on multivariate data analysis [17]. For any two objects $\mathbf{x}_i$, $\mathbf{x}_j$ given by tuples of cardinality n, this index is given by the expression:

$$s_G(\mathbf{x}_i, \mathbf{x}_j) = \frac{\sum_{k=1}^{n} s_k(x_{ik}, x_{jk})\, \delta_{ijk}}{\sum_{k=1}^{n} \delta_{ijk}} \tag{11}$$

where s_k is the similarity according to variable k, and δ_{ijk} is a binary function expressing whether the objects are *comparable* or not according to variable k, equal to 1 if $x_{ik} \neq \mathcal{X} \wedge x_{jk} \neq \mathcal{X}$ and to 0 otherwise. The ignorance of the absent elements normalized by the number of present ones has been found superior to other treatments in standard data analysis experiments [11].

The possibility that none of the partial similarities can be performed has to be handled with special care, since the result is $\frac{0}{0}$, which is unfeasible from the point of view of a computation. Pessimistic solutions like setting $s_G(\mathbf{x}_i, \mathbf{x}_j) = 0$ (objects are *incomparable*) or optimistic ones like setting $s_G(\mathbf{x}_i, \mathbf{x}_j) = 1$ (objects are *indistinguishable*) are not fully satisfactory. The present framework offers a third possibility, namely, to set $s_G(\mathbf{x}_i, \mathbf{x}_j) = \mathcal{X}$ expressing that this computation of similarity is lacking. This missing value can be eventually processed by other S-neurons in subsequent layers.

As a further consequence of Definition (3.10), missing values are allowed in the weights. This is interpreted by the neuron in a way analogous to a missing input: it is ignored (as if the connection were not there). This is consistent with the fact that a similarity measure must be symmetric and could be beneficial to a learning algorithm, enabling the units to ignore some of their inputs. If a weight is missing at the end of a training process, then the connection can be removed (a form of network pruning).

Since (11) is a *linear* aggregation operator, a non-linear component can be added to in the form of a similarity keeping function $\check{s}$. In particular, the widespread *logistic* function can be used by adapting it to map the real interval $[0, 1]$ on $(0, 1)$. Computationally cheap families of sigmoidal functions can be especially designed to operate in the $[0, 1]$ interval, such as $\check{s}(\cdot) = f(\cdot, k)$ [30]:

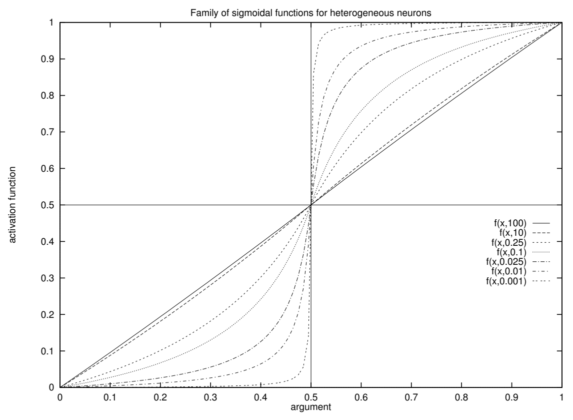


Figure 2: The family of sigmoidal functions $f(x, k)$, for different values of k.

$$f(x,k) = \begin{cases} \frac{-k}{(x-0.5)-a(k)} - a(k) & \text{if } x \leq 0.5 \\ \frac{-k}{(x-0.5)+a(k)} + a(k) + 1 & \text{if } x \geq 0.5 \end{cases} \tag{12}$$

where $k > 0 \in \mathbb{R}$ is a parameter controlling the shape. This family of functions (Fig. 2) meets the conditions in Def. (3.3). They are all continuous bijections in $[0, 1]$, fulfilling $\forall k \in \mathbb{R}^+$, $f(0, k) = 0$, $f(1, k) = 1$, $\lim_{k \to \infty} f(x, k) = x$ and $f(x, 0) = H(x - 0.5)$, being H the Heaviside function. The expression $a(k)$ is solution of $a(k)^2 + \frac{a(k)}{2} - k = 0$, obtained by imposing the first two equalities.

This S-neuron has many advantages: it is intuitive, fulfills all the desired properties for aggregation, is computationally cheap and is also a generic measure, in the sense that it does not take assumptions about how the partial similarities are computed.

4 An experimental study

4.1 Problem description

The following data sets are studied: the well-known *Pima Diabetes, Horse Colic, Credit Card, Heart Disease* and *Solar Flares* taken from the Proben repository [27], *Sinus-Cosinus* from [5] and *SISO-Bench* from [29], for a total of seven learning tasks, altogether representative of the kinds of variables typically found in real problems, while displaying different degrees of missing information (from 0% to 26%). Most of the problems found in these repositories were originally meant for general machine learning approaches, and cannot be directly used by traditional neural systems because of the usual presence of non-continuous or missing information. Their main characteristics are displayed in Table 1.

4.2 Experimental methodology

The primary aim has been to compare the *relative* performance of the HNN over the classical RBF. The network architectures are composed of a hidden layer of $h \in \{4, 8, 12, 16\}$ neurons plus as many output units as required by the task, sigmoidal for classification problems and linear otherwise. It should be pointed out that a principled model selection process for a problem is a subject of research in itself [21]. Hence, the results on standard benchmarks should not be taken as state-of-the-art (although they are not out of tune) but as developed examples of application.

4.3 Training methodology

A form of nested k-fold non-stratified cross-validation (CV) is used. The idea of nested or *double* cross-validation dates from the seventies [22], and works as follows: divide the whole data set into k_1 pieces, keeping each back in turn for independent testing, and using the rest for k_2-fold CV. This fits k_1k_2 models. Using one of the k_1 pieces for test, one for validation and the remaining $k_1 - 2$ for training, this is equivalent to set $k_2 = k_1 - 1$, so that a total of $k_1(k_1 - 1)$ fits are to be done. This is a very general framework using no special model assumptions. Although it cannot "validate" a model, it gives an unbiased estimate of generalization [14]. We have worked out a simplification to avoid an excessive computational overhead, by performing only half of the computations involved. In particular, we set $k_1 = 5$, so that three folds go for training, one for validation and one for test, forming twenty 60%-20%-20% partitions. A representative half of the combinations can be obtained by selecting a balanced subset, such that each of the 5 folds appears twice as a validation fold, twice as a test, and 6 times (three times two) in one of the 3 training positions. It is also ensured that no two training parts are generated out of the same 3 folds but in a different order.

Early stopping is used in conjunction with double CV as follows. The network is trained up to a maximum number of error function evaluations, keeping track of the best validation error. Strictly speaking, there is no early stopping (only the use of a validation set), as long as the process is not halted when a minimum of the validation error is attained. When done, the reported error is given by the error on the test part using the network that produced the lowest error in the validation part. This scheme is simple and has been reported to be superior to regularization methods in many cases (e.g. [15]). For each of the ten selected partitions, ten runs are carried out, varying the initial conditions of the learner. Therefore, a total of 100 fits per dataset, architecture and model are performed. To avoid predetermined initial arrangements, the data sets are randomly shuffled.

The example S-neuron described in (§3.6) is compared to a classical RBF neuron based on Euclidean distance, followed by a Gaussian with its own variance σ_i^2 (to be learned), as $\gamma_i(\mathbf{x}) = g(||\mathbf{x} - \mathbf{w}_i||)$ where $g(z) = exp\{-\frac{1}{2}\frac{z^2}{\sigma_i^2}\}$. Specifically, the standard metric in $\mathbb{R}$ weighted by the maximum deviation (§3.3.3) using $\hat{s}_0$ ($\alpha = 4, d = 1$) for continuous variables, and the measures (3), (4), (7) and (8), for ordinal, nominal, fuzzy and linguistic variables, respectively, followed by a logistic $\check{s}(z) = f(z, 0.1)$ (12). It can be shown that the obtained measure is not affected by any linear normalization of the variables (including standardization), rendering this usual preprocessing step redundant.

The data sets for the RBF network are formed using the less distorting techniques among those explained in (§2): ordinal variables are mapped to an equidistant linear scale, a 1-out-of-k coding is used for nominal ones and an extra input is added for those variables with missing values. Real-valued variables are normalized to $[0, 1]$. This is not needed by the S-neuron because it already computes a normalized measure, but is beneficial for the RBF network.

4.3.1 Training procedure

In traditional neural learning algorithms having a *differentiable* unit is essential for the application of gradient-descent techniques, which restricts their applicability. Due to the existence of domains other than the real continuum, the presence of missing data and the eventual use of a non-differentiable similarity function. The one chosen in this work is based on a *Breeder Genetic Algorithm* (BGA) [23], a method in mid position between Genetic Algorithms (GA) [9] and Evolution Strategies [2] that deals directly with continuous information. While in GA selection is stochastic and meant to mimic Darwinian evolution, BGA selection is driven by a deterministic *breeding*

Table 1: Basic features of the data sets: Def. (default accuracy), Missing (percentage of missing values), Missing-1 (percentage of cases with at least one missing value). Last column shows original heterogeneity.

Name	Type	#Cases	Def.	Missing	Missing-1	In→Out	Data
Pima Diabetes	C	768	65.1%	10.6%	48.8%	$8 \rightarrow 2$	6R, 0N, 2O
Credit Card	C	690	55.5%	0.65%	5.4%	$15 \rightarrow 2$	6R, 9N, 0O
Horse Colic	C	364	61.5%	26.1%	98.1%	$20 \rightarrow 3$	5R, 5N,10O
Heart Disease	C	920	55.3%	16.2%	67.5%	$13 \rightarrow 2$	3R, 6N, 4O
Solar Flares	R	1066	-	0.0%	0.0%	$9 \rightarrow 3$	0R, 5N, 4O
Sinus-Cosinus	R	400	-	0.0%	0.0%	$2 \rightarrow 1$	2R, 0N, 0O
SISO-Bench	R	500	-	0.0%	0.0%	$2 \rightarrow 1$	2R, 0N, 0O

C classification R regression R real N nominal O ordinal

mechanism, an artificial method in which only the best individuals –usually a fixed percentage τ of total population size μ– are selected to be recombined and mutated, as the basis to form a new generation. The BGA as used here does not need any coding scheme, since its genetic operators are extended to handle heterogeneous information [4].

The algorithm is used with an identical parameter setup for all the experiments, regardless of the neuron model, architecture or data set, to exclude this source of variation from the analysis. The BGA task is to minimize the RSE (13) on the training part, until $30,000$ error evaluations are used (*end of resources*) in each run. Ten independent runs are performed for each specific training scenario. The BGA is set to the following parameters: $\mu = 100, \tau = 25$, extended intermediate recombination with $\delta = 0.45$ and continuous mutation with $\rho = 0.5, k = 8$. A total of 300 generations $(30,000/100)$ are carried out for each run. This setting has been proven very useful in the context of ANN optimization [10].

4.4 Results obtained

We present performance results measured across four coordinates: **generalization ability**, **network complexity**, **readability** and **computational cost**.

To measure the **generalization ability**, we report the test set *normalized root square error* (NRSE) over the $n = 100$ fits, given by:

$$NRSE = \sqrt{\frac{\sum_i ||\zeta_i - \mathbf{y}_i||^2}{\sum_i || < \mathbf{y}_i > - \mathbf{y}_i||^2}} \qquad (13)$$

where ζ_i is the network's response to input pattern $\mathbf{x}_i$, and $< \mathbf{y}_i >$ represents the mean of the target function over the data set. This value gives an impression of how good a result is and, being normalized, permits a comparison across different data sets. Reasonable values are to be expected in $[0, 1]$. Values lower than 0.5 indicate fair fits. Values greater than 1.0 indicate a poor performance. A value of 1.0 actually signals a model as good as the average predictor $\zeta_i =< \mathbf{y}_i >$. For the classification problems, we also report the average test set correct classification rates. The results are summarized in Tables 3 (left and right) and 2, in which the results are presented from the perspective of the architectures, averaged over the seven data sets, and vice versa (this makes sense because the errors are normalized).

Table 2: Generalization NRSEs for every architecture, averaged over the seven data sets.

	RBF	HNN
4 hidden	0.6569	0.5745
8 hidden	0.6427	0.5724
12 hidden	0.6362	0.5795
16 hidden	0.6352	0.5938

The cornerstone of the networks grounded on S-neurons relies in its good average behaviour. Across all the data sets, the results for these networks are consistently better (about a 10%) than those obtained by the RBF network. This margin can certainly make a difference in some application.

We define the **network complexity** $\hat{p}$ as its number of parameters to be optimized. For a network

Table 3: Left: Mean generalization accuracies for classification data sets, averaged over the four architectures. The results in parentheses show an average of the *best* values across the ten partitions (all these averages take test set sizes into account). Right: Generalization NRSEs for regression data sets, averaged over the four architectures.

Problem	RBF	HNN
Pima Diabetes	69.98 (73.66)	74.92 (77.72)
Horse Colic	64.09 (66.15)	65.08 (69.90)
Heart Disease	80.14 (81.51)	81.07 (83.42)
Credit Card	76.18 (81.19)	81.84 (84.86)

Problem	RBF	HNN
Solar Flares	0.8553	0.9374
Sinus-Cosinus	0.3840	0.0424
SISO-Bench	0.1146	0.0234

Table 4: Complexities for the 12-hidden networks.

Problem	RBF		HNN	
	$\hat{n}$	$\hat{p}$	$\hat{n}$	$\hat{p}$
Pima Diabetes	13	194	10	146
Horse Colic	54	699	46	591
Heart Disease	33	434	14	194
Credit Card	51	650	21	278
Solar Flares	23	327	21	291
Sinus-Cosinus	5	49	2	37
SISO-Bench	5	49	2	37

with a hidden layer of h neurons and an output layer of l linear neurons, this number is given by $\hat{p} = h \cdot m(\hat{n}) + (h+1) \cdot l$, where $m(\hat{n})$ is the number of *model* inputs (number of inputs *after* pre-processing). These numbers are shown in Table 4 just for 12-hidden networks as an example; it can be seen that they are lower for the HNN (sometimes substantially).

The **readability** of the obtained solutions is illustrated in Table 5. We show the heterogeneous weights of a hidden neuron taken at random from one of the hundred networks delivered by cross-validation for the *Horse Colic* problem. This problem is chosen because it displays a good amount of heterogeneity. Linguistic terms are shown in graphical form, whereas triangular fuzzy numbers are shown in numerical form –rounded to one decimal– for clarity. Note the obtained linguistic terms are almost symmetric, a characteristic found by the network itself. Although it would require an expert judgement, this example neuron could be regarded as the *soft* prototype of a somewhat "standard" young horse. This assigns a well-defined meaning to individual weights and to the neuron outcome (a similarity degree in $[0, 1]$), and at the same time permits to interpret the trained network in the original input domain (i.e., a collection of measurements about horses). In contrast, the weight vector for each RBF neuron consists (for this problem) of 54 real numbers.

The **computational cost** associated to each neuron model is estimated as follows. From the total CPU time of each training session (for all data sets and architectures) the total time taken by a hidden neuron is computed, normalizing by the number of patterns, and averaging across the different data sets. The resulting quantity is then divided by the number of times a neuron is asked to process the entire data set which, in this study, amounts to $10 \times 10 \times 30,000$ (10 folds, 10 runs per fold, 30,000 data set evaluations per run). The obtained quantity is the average number of CPU milliseconds that a hidden neuron takes to evaluate one pattern of the training set, across all the data sets[2]. The resulting numbers are 0.00184 ms (RBF) and 0.00175 ms (HNN).

In summary, the obtained results can be said to be satisfactory in a number of senses:

1. The generalization ability is remarkably better, on all but one of the data sets and very much on the average across all the data sets.

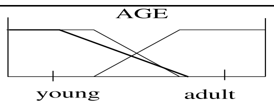
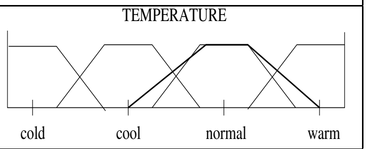
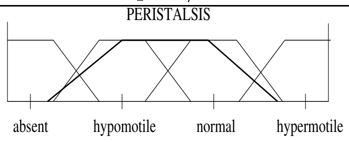
2. The number of parameters, for same numbers of hidden units, is lower.

3. The readability of the obtained solutions is enhanced.

4. The training time is slightly less than for a RBF network.

[2]The machine used for the experiments is a dedicated SUN^{TM} Enterprise E-250 (2 CPU ULTRA-SPARC-II at 400Mhz, 128Mb RAM), rated at 16.8 SPECint95 and 13.5 SPECint_base95 marks.

Table 5: Some heterogeneous weights of a hidden neuron for the *Horse Colic* problem.

#	Name	Type	Value
1	Age	*linguistic*	
2	Rectal temperature (celsius)	*fuzzy number*	37.3 ± 2.1
3	Pulse (beats per minute)	*ordinal*	70
4	Respiratory rate (times per minute)	*ordinal*	64
5	(Subjective) temperature of extremities	*linguistic*	
7	Mucous membranes color	*nominal*	“normal pink”
9	(Subjective) pain estimation	*nominal*	“no pain/alert”
10	Peristalsis: gut activity	*linguistic*	
14	Nasogastric reflux PH	*fuzzy number*	2.1 ± 0.3
16	Abdomen	*nominal*	“distended small intestine”
18	Total protein (grs./dl)	*fuzzy number*	73.6 ± 19.8

5 Conclusions and outlook

The conceptual and functional definition of a neuron as a similarity computing device permits to base the computation on principled grounds, and carries a number of advantages: first, the task performed by the hidden neurons is clearly delimited, allowing for a deeper influence on the workings of a neural network; second, it gives a precise *semantics* to the computation, thereby facilitating the interpretation process; third, the possibility of adding prior knowledge to the model and the treatment of different data sources are nicely integrated in the framework. The cooperative work of different *soft computing* techniques (neural networks, evolutionary algorithms and fuzzy sets) makes these networks capable of learning from non-trivial data sets.

We would like to stress the fact that the experiments presented constitute worked examples of use, for which little domain knowledge was available (only the data types) and some design decisions had to be made. Therefore, a different decision on how to treat a given variable could have been taken. This is, however, one of the advantages of the approach. The incorporation of correct prior knowledge, together with a methodology that respects the nature of the data and endows the networks with a clear semantics are on the basis of the altogether very satisfactory results. This pattern of behaviour has already been observed on experiments carried out with a preliminary studied model, applied to real-world problems in Environmental Sciences [3]. We hence believe that our contributions can be effective in a broad spectrum of situations, and at the same time offering the possibility to be tailored to specific problems. It is our hypothesis that these models can add to the *practical* computing power of artificial neural models.

References

[1] Aamodt, A., Plaza, E. “Case-based reasoning: Foundational issues, methodological variations and system approaches”. *AI Communications*, 7(1):39-59, 1994.

[2] Bäck, Th., Schwefel, H.P. “A Survey of Evolution Strategies”. In Procs. of the 4th Intl. Conf. on Genetic Algorithms, Booker and Belew (eds.), Morgan Kaufmann: San Mateo, pp. 2-9, 1991.

[3] Belanche, Ll., Valdés, J.J., Comas, J., Roda, I., Poch, M. “Prediction of the bulking phenomenon in wastewater treatment plants”. *Artificial Intelligence in Engineering*, 14(4): 307-317, 2000.

[4] Belanche, Ll. "Evolutionary Optimization of Heterogeneous Problems". In *Parallel Problem Solving from Nature.* Merelo et al. (Eds.) LNCS 2439, pp. 475-484. Springer-Verlag, 2002.

[5] Bersini H., Bontempi, G. "Now comes the time to defuzzify neuro-fuzzy models". *Fuzzy sets and systems*, 90(2), 1997.

[6] Bishop, C. *Neural Networks for Pattern Recognition.* Clarendon Press, 1995.

[7] Chandon, J.L., Pinson, S. *Analyse Typologique. Théorie et Applications.* Masson, 1981.

[8] Cherkassky, V., Mulier, F. *Learning from data: concepts, theory and methods.* Wiley, 1998.

[9] Davis, L.D. *Handbook of Genetic Algorithms.* Van Nostrand Reinhold, 1991.

[10] De Falco, I., Della Cioppa, A., Natale, P., Tarantino, E. "Artificial neural networks optimization by means of evolutionary algorithms". In *Soft Computing in Eng. Design & Manufacturing.* Springer, 1997.

[11] Dixon, J.K. "Pattern recognition with partly missing data". *IEEE Trans. on Systems, Man and Cybernetics*, 9: 617-621, 1979.

[12] Dubois, D., Esteva, F., García, P., Godo, L. Prade, H. "A logical approach to interpolation based on similarity relations". IIIA Research Report 96-07. Barcelona, Spain, 1996.

[13] Duda, R.O., Hart, P.E. *Pattern classification and scene analysis.* John Wiley, 1973.

[14] Fiesler, E., Beale, R. (Eds.) *Handbook of Neural Computation.* IOP & Oxford Univ. Press, 1997.

[15] Finnof W., Hergert F., Zimmermann, H.J. "Improving model selection by non-converging methods". *Neural Networks*, 6: 771-783, 1993.

[16] Ghahramani, Z., Jordan, M.I. "Learning from incomplete data". Memo. 1509, MIT, 1994.

[17] Gower, J. "A general coefficient of similarity and some of its properties". *Biometrics*, 27, 1971.

[18] Hahn, U., Chater, N., "Understanding Similarity: A Joint Project for Psychology, Case-Based Reasoning, and Law". *Artificial Intelligence Review*, 12: 393-427, 1998.

[19] Hebb, D. *The organization of behaviour*, John Wiley, 1949.

[20] Little, R.J.A., Rubin, D.B. *Statistical analysis with missing data.* John Wiley, 1987.

[21] Moody, J. "Prediction Risk and Architecture Selection for Neural Networks". *From Statistics to Neural Networks: Theory and Pattern Recognition Applications.* Cherkassky, et al. (Eds.), NATO ASI Series F, 136. Springer, 1994.

[22] Mosteller, F., Tukey, F. *Data analysis & regression.* Addison-Wesley, 1977.

[23] Mühlenbein, H., Schlierkamp-Voosen, D. "Predictive Models for the Breeder Genetic Algorithm". *Evolutionary Computation*, 1(1): 25-49, 1993.

[24] Murphy, P.M., Aha, D. "UCI Repository of machine learning databases". UCI Dept. of Information and Computer Science, 1991.

[25] Pedrycz, W. "Fuzzy sets in pattern recognition: accomplishments and challenges". *Fuzzy sets and systems*, Vol. 90, No. 2, 1997.

[26] Poggio T., Girosi, F. "A theory of networks for approximation and learning". MIT Memo 1140, 1989.

[27] Prechelt, L. "Proben1: A set of Neural Network Benchmark Problems and Benchmarking Rules". Facultät für Informatik. Univ. Karlsruhe. Technical Report 21/94, 1994.

[28] Stanfill, C., Waltz, D. "Toward memory-based reasoning". *Communs. of the ACM* 29, 1986.

[29] Su, Y., Sheen, Y. "Neural networks for system identification". *Intl. J. of Systems Science*, 23(12), 1992.

[30] Valdés, J.J, García, R. "A model for heterogeneous neurons and its use in configuring neural networks for classification problems". LNCS-1240, pp. 237-246. Springer-Verlag, 1997.

[31] Wilson, D.R., Martinez, T.R, "Heterogeneous Radial Basis Function Networks". In Procs. of the 6th Intl. Conf. on Neural Networks (ICANN'96), vol. 2, pp. 1263-1267, 1996.

[32] Zadeh, L. "Fuzzy Sets as a basis for a theory of possibility". *Fuzzy Sets & Systems* (1),3-28, 1978.

[33] Zimmermann, H.J. *Fuzzy set theory and its applications.* Kluver Academic Publishers, 1992.