# Fuzzy Inputs and Missing Data in Similarity-Based Heterogeneous Neural Networks

Lluís A. Belanche and Julio J. Valdés

Secció d'Intel·ligència Artificial.
Dept. de Llenguatges i Sistemes Informàtics.
Universitat Politècnica de Catalunya.
c/Jordi Girona Salgado 1-3
08034 Barcelona, Spain.
{belanche, valdes}@lsi.upc.es
Phone: +34 93 401 56 44
Fax: +34 93 401 70 14

**Abstract.** Fuzzy heterogeneous networks are recently introduced neural network models composed of neurons of a general class whose inputs and weights are mixtures of continuous variables (crisp and/or fuzzy) with discrete quantities, also admitting missing data. These networks have net input functions based on similarity relations between the inputs and the weights of a neuron. They thus accept heterogeneous –possibly missing– inputs, and can be coupled with classical neurons in hybrid network architectures, trained by means of genetic algorithms or other evolutionary methods. This paper compares the effectiveness of the fuzzy heterogeneous model based on similarity with the classical feed-forward one, in the context of an investigation in the field of environmental sciences, namely, the geochemical study of natural waters in the Arctic (Spitzbergen). Classification performance, the effect of working with crisp or fuzzy inputs, the use of traditional scalar product *vs.* similarity-based functions, and the presence of missing data, are studied. The results obtained show that, from these standpoints, fuzzy heterogeneous networks based on similarity perform better than classical feed-forward models. This behaviour is consistent with previous results in other application domains.

## 1 Introduction

The notion of heterogeneous neurons was introduced in [11] as a model accepting as inputs vectors composed of a mixture of continuous real-valued and discrete quantities, possibly also containing missing data. The other feature of this model departuring from the classical was its definition as a general mapping from which different instance models can be derived. In particular, when the model is constructed as the composition of two mappings, different instance models can be derived by making concrete choices of the net input and activation functions, mimicking the classical neuron model. In this special case, whereas the classical neuron model uses dot product as net input, and sigmoid (or hyperbolic tangent) as squashing functions for activation, the heterogeneous model uses, respectively, a *similarity* or *proximity relation* [4] between the input and the weight tuples, and a sigmoid-like bijection of the reals in $[0,1]$. The choice of the specific similarity function should account for the heterogeneous nature of neuron inputs and the presence of missing data. This was shown to be a reasonable brick for constructing layered network architectures mixing heterogeneous with classical neurons, since the outputs of these neurons can be used as inputs for the classical ones. Such type of hybrid networks is composed of one hidden layer of heterogeneous networks and one output layer of classical neurons. In this case the heterogeneity of the solution space makes genetic algorithms a natural choice for a training procedure, and indeed, these networks were able to learn from non-trivial data sets with an effectiveness comparable, and sometimes better, than that of classical methods. They also exhibited a remarkable robustness when information degrades due to the increasing presence of missing data. One step further in the development of the heterogeneous neuron model was the inclusion of fuzzy quantities within the input set, extending the former use of real-valued quantities of crisp character. In this way, uncertainty and imprecision (in inputs *and* weights) can be explicitly considered within the model, making it more flexible. In the context of a real-world application example in geology [12], it was found that hybrid networks using fuzzy heterogeneous neurons perform better by treating the same data with its natural imprecision than considering them as crisp quantities, as is usually done. Moreover, in the same study it was found that hybrid networks with heterogeneous neurons in general (i.e. with or without fuzzy inputs) outperform feed-forward networks with classical neurons, even when trained with sophisticated procedures like a combination of gradient techniques with simulated annealing.

In this paper, the possibilities of this kind of neurons are illustrated by comparison to fully classical architectures in a real-world problem. The paper is organized as follows. Section 2 reviews the concept of heterogeneous neurons and their use in configuring hybrid neural networks for classification tasks. Section 3 describes the example application at hand, fruit of an environmental research in the Arctic, while Section 4 covers the different experiments performed: description, settings and discussion. Finally, Section 5 presents the conclusions.

## 2 The Fuzzy Heterogeneous Neuron Model

A fuzzy heterogeneous neuron was defined in [12] as a mapping $h : \hat{\mathcal{H}}^n \to \mathcal{R}_{out} \subseteq \mathbb{R}$. Here $\mathbb{R}$ denotes the reals and $\hat{\mathcal{H}}^n$ is a cartesian product of an arbitrary number $n$ of *source sets*. These source sets may be extended reals $\hat{\mathcal{R}}_i = \mathbb{R}_i \cup \{\mathcal{X}\}$, extended families of (normalized) 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 extension is given by the special symbol $\mathcal{X}$, which denotes the unknown element (missing information) and it behaves as an *incomparable* element w.r.t. any ordering relation. Consider now the collection of $n_f$ extended fuzzy sets of the form $\hat{\mathcal{F}}_i = \mathcal{F}_i \cup \{\mathcal{X}\}$ and their cartesian product $\hat{\mathcal{F}}^{n_f} = \hat{\mathcal{F}}_1 \times \hat{\mathcal{F}}_2 \times \ldots \times \hat{\mathcal{F}}_{n_f}$. The resulting input set will then be $\hat{\mathcal{H}}^n = \hat{\mathcal{R}}^{n_r} \times \hat{\mathcal{F}}^{n_f} \times \hat{\mathcal{O}}^{n_o} \times \hat{\mathcal{M}}^{n_m}$, where the cartesian products for the other kinds of source sets $(\hat{\mathcal{R}}^{n_r}, \hat{\mathcal{O}}^{n_o}, \hat{\mathcal{M}}^{n_m})$ are constructed in a similar way from their respective cardinalities $n_r, n_o, n_m$, with $\hat{\mathcal{R}}^0 = \hat{\mathcal{F}}^0 = \hat{\mathcal{O}}^0 = \hat{\mathcal{M}}^o = \hat{\mathcal{H}}^o = \phi$, $n = n_r + n_f + n_o + n_m$, and $n > 0$. According to this definition, neuron inputs are vectors composed of $n$ elements among which there might be reals, fuzzy sets, ordinals, nominals and missing data.

An interesting particular class of heterogeneous submodels is constructed by considering $h$ as the composition of two mappings $h = f \circ s$ , 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 seen as a $n$-ary function parameterized by a $n$-ary vector $\hat{\boldsymbol{w}} \in \hat{\mathcal{H}}^n$ representing the neuron's weights, i.e. $h(\hat{\boldsymbol{x}}, \hat{\boldsymbol{w}}) = f(s(\hat{\boldsymbol{x}}, \hat{\boldsymbol{w}}))$. Within this framework, several of the most common artificial neuron models can be derived. For example, the classical scalar-product driven model is obtained by making

$n = n_r$ (and thus $n_f = n_o = n_m = 0$), no missing data at all, $s(\hat{\boldsymbol{x}}, \hat{\boldsymbol{w}}) = \hat{\boldsymbol{x}} \cdot \hat{\boldsymbol{w}}$, and choosing some suitable sigmoidal for $f$. However, there are many possible choices for the function $s$, and some of them are currently under investigation. In particular, from its very beginning [11], the function $s$ represents a *similarity index*, or proximity relation (where transitivity considerations are put aside): a binary, reflexive and symmetric function $s(x, y)$ with image in $[0, 1]$ such that $s(x, x) = 1$ (strong reflexivity). The semantics of $s(x, y) > s(x, z)$ is that object $x$ is more similar to object $y$ than is to object $z$. The function $f$ takes the familiar form of a squashing non-linear function with domain in $[0, 1]$. That is, the neuron is sensitive to the degree of similarity between its input and its weights, both composed in general by a mixture of continuous and discrete quantities, possibly with missing data. It has been our postulate that such a family of functions are, in general, better suited for pattern recognition devices than the classical scalar product and derived measures.

The concrete instance of the model used in the present paper uses a *Gower-like* similarity index [7] in which the computation for heterogeneous entities is constructed as a weighted combination of partial similarities over subsets of variables, which were singletons in the original definition, although any problem-specific partition is conceivable. 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 the expression

$$s_{ij} = \frac{\sum_{k=1}^{n} g_{ijk}\, \delta_{ijk}}{\sum_{k=1}^{n} \delta_{ijk}}$$

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. In particular, for a continuous variable $k$ and any two objects $i$, $j$ the following similarity score is used:

$$g_{ijk} = 1 - \frac{|v_{ik} - v_{jk}|}{\text{range }(v_{\cdot k})}$$

Here, $v_{ik}$ denotes the value of object $i$ for variable $k$ and range $(v_{\cdot k}) = \max_{i,j} (|v_{ik} - v_{jk}|)$ (see [7] for details on other kinds of variables). The $\delta_{ijk}$ is a binary function expressing whether both objects are comparable or not according to their values w.r.t. variable $k$. It is 1 if and only if both objects have values different from $\mathcal{X}$ for variable $k$, and 0 otherwise. This way, in the model considered here, Gower's original definitions for real-valued and discrete variables are kept. For variables representing fuzzy sets, similarity relations from the point of view of fuzzy theory have been defined elsewhere [5], [15], 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:

$$g(\tilde{A}, \tilde{B}) = \max_{x} \, (\mu_{\tilde{A} \cap \tilde{B}}(x))$$

where $\mu_{\tilde{A} \cap \tilde{B}}(x) = \min \, (\mu_{\tilde{A}}(x), \mu_{\tilde{B}}(x))$. As for the activation function, a modified version of the logistic is used (see [11]), that maps the real interval $[0, 1]$ on $(0, 1)$. The training procedure for fuzzy heterogeneous networks of the kind described is based on genetic algorithms ([13], [14]), because the, in general, non-differentiability of the net input function, and the presence of missing information prevent the use of gradient-based techniques. The resulting heterogeneous neuron can be used for configuring feed-forward network architectures in several ways. In this paper it is shown how layered feed-forward structures with a hidden layer composed of heterogeneous neurons and an output layer of classical units are natural choices better suited for the data than the fully classical counterparts.

## 3 An example of application: environmental research in the Arctic

During the scientific expedition Spitzbergen'85, organized by the University of Silesia (Poland), a scientific team composed of specialists from this university, the National Center for Scientific Research (Cuba), and the Academy of Sciences of Cuba, performed glaciological and hydrogeological investigations in several regions of the Spitzbergen island (Svalbard archipelago, about $76^o N$ to $80^o N$). The purpose was to determine the mass and energy balance within experimental hydrogeological basins, the study of the interaction between natural waters and rock-forming minerals in the severe conditions of polar climate and their comparison with similar processes developed in tropical conditions. This has been a long-term research of several Polish universities (Silesia, Warsaw and Wroclaw) and the Polish Academy of Sciences since the First Geophysical Year in 1957, and represents an important contribution to the evaluation of the impact of global climatic changes. Complex interactions take place due to peculiar geological, geomorphological and hydrogeological conditions which, in the end, reflect in water geochemistry.

In this study, a collection of water samples were taken from different hydrogeological zones in two Spitzbergen regions. They were representative of many different zones: subglaciar, supraglaciar, endoglaciar, springs (some hydrothermal), lakes, streams, snow, ice, the tundra and coastal. Among the physical and chemical parameters determined for these water samples, the following nine were used for the present study: temperature, pH, electrical conductivity, hydrocarbonate, chloride, sulphate, calcium, magnesium and sodium-potasium. Geochemical and hydrogeological studies of these data [8], [9] have shown a relation between the different hydrogeological conditions present in Spitzbergen and the chemical composition of their waters, reflecting the existence of several families. That is, an indirect assessment of their hydrogeological origin is in principle possible from the information present in the geochemical parameters, thus enabling the use of a learning algorithm.

## 4 Experiments

### 4.1 General Information

The available set of $N = 114$ water samples from Spitzbergen, corresponding to $c = 5$ hydrogeological families of waters, was used for comparative studies of supervised classification performance (error and accuracy) using different neural architectures, described below. To express the distribution of samples among classes we introduce the notation $n_k$ to denote that there are $n$ samples of class $k$. This way, the actual distribution was $37_1, 29_2, 10_3, 11_4, 27_5$. *Default accuracy* (relative frequency of the most common class) is then 37/144 or 32.5%. *Entropy*, calculated as $-\sum_{k=1}^{c}(n_k/N)\ log_2(n_k/N)$, is equal to 2.15 bits. There were no missing data and all measurements were considered to have a maximum of 5% of imprecision w.r.t. the reported value. This aspect will be taken into account when considering uncertainty in the form of fuzzy inputs, since the fact that the physical parameters characterizing the samples as well as their chemical analysis were done *in situ* –in the extremely hard climatic and working conditions of the Arctic environment– makes them particularly suited to a kind of processing in which uncertainty and imprecision are an explicit part of the models used. Accordingly, hybrid feed-forward networks composed of a first (hidden) layer of heterogeneous neurons, mixed with an output layer of classical ones is the basic architectural choice for this case study. These hybrid architectures will be compared to their fully classical counterparts –under the same experimental settings– in order to assess their relative merits. To this end, the following notation is introduced: let $q_x$ denote a *single* layer of $q$ neurons of type $x$, where possibilities for $x$ are:

$n$ Classical: real inputs, scalar-product net input and logistic activation.
$h$ Heterogeneous: real inputs, similarity-based net input and (adapted) logistic activation.
$f$ Fuzzy heterogeneous. Triangular fuzzy inputs (converted from the original crisp reported value by adding a 5% of imprecision, see fig. 1) similarity-based net input and (adapted) logistic activation.

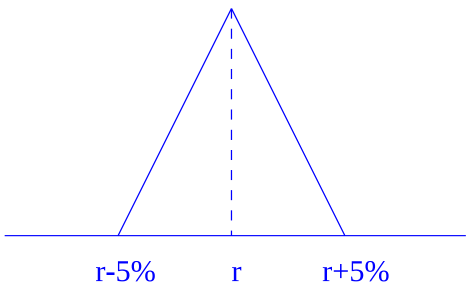


**Fig. 1.** A triangular fuzzy number constructed out of the reported crisp value $r$.

Accordingly, $p_x q_y$ denotes a feed-forward network composed of a hidden layer of $p$ neurons of type $x$ and an output layer of $q$ neurons of type $y$. For example $4_h 5_n$ is a network composed of a hidden layer of 4 neurons of type $h$ and an output layer of 5 neurons of type $n$. All units use the logistic as activation. Shortcut (direct input to output) connections are not considered.

All neural architectures will be trained using a standard genetic algorithm (SGA) with the following characteristics: binary-coded values, probability of crossover: 0.6, probability of mutation: 0.01, number of individuals: 52, linear rank scaling with factor: 1.5, selection mechanism: stochastic universal, replace procedure: worst. The algorithm was stopped unconditionally after 5,000 generations or if there was no improvement for the last 1,000. This last criterion helps evaluating the goodness of the architecture being trained and saves unuseful computing time.

## 4.2 Experiment Settings

In the present study, *all* models (including the classical feed-forward one) were trained using exactly the same procedure and parameters in order to exclude this source of variation from the analysis. Of course, fully classical architectures need not be trained using the SGA. They could instead be trained using any standard (or more sophisticated) algorithm using gradient information. However, this would have made direct comparison much more difficult, since one could not attribute differences in performance exclusively to the different neuron models, but also to their training algorithms. The experiment settings were the following:

**Training regime** The training set was composed of 32 representative samples (28% of the whole data set), whereas the remaining 82 (72%) constituted the test set, a deliberately chosen hard split for generalization purposes. Class distribution is $8_1, 7_2, 5_3, 5_4, 7_5$ in training and $29_1, 22_2, 5_3, 6_4, 20_5$ in test. Default accuracies are 25.0% and 35.4%, respectively.

**Architectures** We will explore the following architectures: $5_x, 2_x5_n, 4_x5_n, 6_x5_n$ and $8_x5_n$, for $x$ in $n, h, f$. Note that the output layer is always composed of five units, one for each water class.

**Number of runs** Every architecture was allowed $R = 5$ runs varying the initial population. All of them were included in the results.

**Weight range** The weights concerning units of type $n$ were limited to be in the range $[-10.0, 10.0]$, to prevent saturation, whereas heterogeneous weights adopt (by definition of the heterogeneous neuron) the same range as their corresponding input variable.

**Error functions** The target error function to be minimized by the training algorithms is the usual **least squared error**, defined as follows:

$$\mathrm{LSE} = \sum_i^p \sum_j^m [\hat{y}_j^i - y_j^i]^2$$

where $y_j^i$ is the $j$-th component of the output vector $\boldsymbol{y}^i$ computed by the network at a given time, when the input vector $\boldsymbol{x}^i$ is presented, and $\hat{y}_j^i = \phi_j(\boldsymbol{x}^i)$, is the target for $x_j^i$, where $\phi_j$ represents the characteristic function for class $j$. The error displayed will be the **mean squared error**, defined as $\mathrm{MSE} = \frac{1}{mp}\mathrm{LSE}$, where $m$ is the number of outputs and $p$ the number of patterns.

## 4.3 Presentation of the Results (I)

Let the classification accuracy for training (TR) and test (TE) sets, calculated with a winner-take-all strategy, be denoted $\mathrm{CA}_{\mathrm{TR}}(r)$ and $\mathrm{CA}_{\mathrm{TE}}(r)$, respectively, for a given run $r$. The errors $\mathrm{MSE}_{\mathrm{TR}}(r)$ and $\mathrm{MSE}_{\mathrm{TE}}(r)$ are similarly defined. For each neural architecture, the following data is displayed:

**Accuracy:** Mean classification accuracy on training $\mathrm{MCA}_{\mathrm{TR}} = \frac{1}{R}\sum_{run=1}^{R} \mathrm{CA}_{\mathrm{TR}}(run)$, same on test $\mathrm{MCA}_{\mathrm{TE}} = \frac{1}{R}\sum_{run=1}^{R} \mathrm{CA}_{\mathrm{TE}}(run)$, and best classification accuracy (BCA) defined as the pair $< \mathrm{CA}_{\mathrm{TR}}(r), \mathrm{CA}_{\mathrm{TE}}(r) >$ with higher $\mathrm{CA}_{\mathrm{TE}}(r)$.

**Error:** Mean MSE in training defined as $\mathrm{MMSE_{TR}} = \frac{1}{R}\sum_{run=1}^{R} \mathrm{MSE_{TR}}(run)$, sample variance in training defined as

$$\mathrm{SVMSE_{TR}} = \frac{1}{R-1}\sum_{run=1}^{R} [\mathrm{MSE_{TR}}(run) - \mathrm{MMSE_{TR}}]^2$$

and similarly defined values $\mathrm{MMSE_{TE}}$ and $\mathrm{SVMSE_{TE}}$ for the test set.

The results are collectively shown in table 1. As an additional reference measure of performance, the $k$-nearest neighbours algorithm (with $k = 5$) is also run on the data –with the same train/test partition– yielding an accuracy in test equal to 58.5%.

| Architecture | Training | | | Test | | | BCA | |
|---|---|---|---|---|---|---|---|---|
| | $\mathrm{MCA_{TR}}$ | $\mathrm{MMSE_{TR}}$ | $\mathrm{SVMSE_{TR}}$ | $\mathrm{MCA_{TE}}$ | $\mathrm{MMSE_{TE}}$ | $\mathrm{SVMSE_{TE}}$ | | |
| $5_n$ | 54.4% | 0.1075 | 2.4e-04 | 46.6% | 0.1661 | 5.8e-05 | 65.6% | 53.7% |
| $5_h$ | 66.3% | 0.1084 | 8.0e-06 | 67.1% | 0.1202 | 1.6e-05 | 75.0% | 76.8% |
| $5_f$ | 99.4% | 0.0338 | 3.0e-06 | 69.3% | 0.0917 | 1.1e-05 | 100% | 75.6% |
| $2_n5_n$ | 41.9% | 0.1314 | 4.3e-04 | 45.4% | 0.1420 | 4.9e-04 | 68.8% | 67.1% |
| $2_h5_n$ | 71.9% | 0.0968 | 2.0e-04 | 69.5% | 0.1088 | 2.6e-04 | 81.3% | 85.4% |
| $2_f5_n$ | 86.3% | 0.0635 | 1.2e-04 | 71.7% | 0.0995 | 9.3e-05 | 81.3% | 81.7% |
| $4_n5_n$ | 70.6% | 0.0785 | 6.1e-05 | 58.3% | 0.1288 | 3.5e-05 | 71.9% | 61.0% |
| $4_h5_n$ | 90.0% | 0.0614 | 1.0e-05 | 79.0% | 0.0786 | 2.9e-05 | 93.8% | 82.9% |
| $4_f5_n$ | 98.1% | 0.0201 | 1.4e-04 | 81.2% | 0.0620 | 1.3e-04 | 100% | 86.6% |
| $6_n5_n$ | 70.0% | 0.0802 | 2.6e-04 | 55.4% | 0.1389 | 7.7e-05 | 81.3% | 58.5% |
| $6_h5_n$ | 91.3% | 0.0508 | 5.0e-05 | 83.7% | 0.0803 | 5.6e-05 | 93.8% | 87.8% |
| $6_f5_n$ | 100% | 0.0106 | 3.0e-06 | 84.9% | 0.0553 | 1.1e-05 | 100% | 90.2% |
| $8_n5_n$ | 87.6% | 0.0396 | 5.7e-05 | 63.7% | 0.1231 | 2.2e-04 | 87.5% | 68.3% |
| $8_h5_n$ | 93.8% | 0.0456 | 1.9e-05 | 86.6% | 0.0603 | 4.0e-05 | 93.8% | 90.2% |
| $8_f5_n$ | 100% | 0.0064 | 4.0e-06 | 80.5% | 0.0541 | 4.3e-05 | 100% | 84.1% |

**Table 1.** Results of the experiments. See text for an explanation of entries.

## 4.4 Analysis of the results (I)

As stated, the experiments were oriented to reveal the influence of several factors:

a) the kind of neural model used (heterogeneous *vs.* classical)
b) the effect of considering imprecision (fuzzy inputs *vs.* crisp inputs), and
c) the effect of missing data in the test set.

The effect of factor (a) can be assessed by comparison, for all the architectures, of the first entry against the other two, column by column. The effect of (b) reflects in the difference between the second vs. the third.

**Single-layer architectures** Let us begin by analysing the results for the architectures with no hidden units, that is, the first three rows of table 1. The interpolation capabilities of the three neuron models can be seen by comparing the value of $\mathrm{MCA_{TR}}$. The mean error $\mathrm{MMSE_{TR}}$ is also a good indicator. The robustness (in the sense of expected variability) can also be assessed by the value of $\mathrm{SVMSE_{TR}}$. It can be seen how the heterogeneous neurons are in general better and much more robust than the classical one. Especially, the fuzzy neuron can learn from the data set to almost perfection very robustly. Similar results are achieved in the test set. Again, an increasing accuracy and decreasing errors and variance indicate an overall better performance. However, the $f$ units are clearly overfitting the data, a fact that shows in the highly unbalanced TR and TE accuracy ratios (both in average and in the best pair BCA).

**Multi-layer architectures** For the four groups of architectures selected (the $p_x 5_n$), there are two aspects amenable to be discussed. First, the relative behaviour of elements of the form $p_x 5_n$, for a fixed $p$. Second, their relative behaviour for a fixed $x$. These two dimensions will collectively give light on any coherent behaviour present in the results.

To begin with, it can be seen that for all the architectures $2_x 5_n$, $4_x 5_n$, $6_x 5_n$ and $8_x 5_n$, as we go through the sequence $n, h, f$, the behaviour is *consistent*: mean accuracies increase, and mean errors and their variances decrease, *both* in training and in test, with the only exception of the error variance in the case $4_x 5_n$. This shows a general superior performance of $h$ neurons over $n$ neurons, and of $f$ neurons over $h$. The absolute differences between neuron models are also noteworthy. In all training respects, the $p_f 5_n$ families show very good interpolation capabilities, explaining the 100% of the TR set *starting from* $p = 4$ in BCA and from $p = 6$ in $\mathrm{MCA_{TR}}$. This trend is followed –to a less extent– by the $p_h 5_n$. The same consistent behaviour is observed in all test indicators. Here the two heterogeneous families show a similar behaviour, with the $f$ neurons slightly above the $h$ ones, until for $p = 8$, the architectures $p_f 5_n$ end up overfitting the data so strongly that their performance in test begins to fall down.

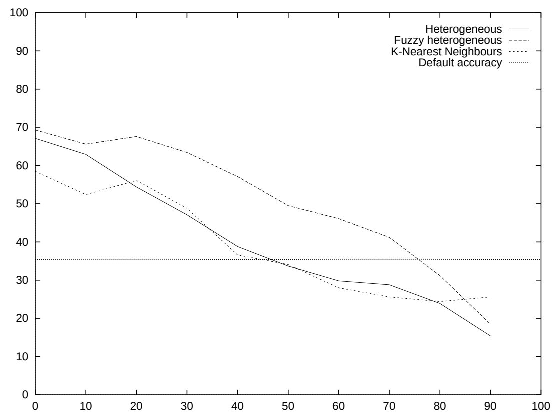
As for the second aspect, $p_x 5_n$ fixing $x$, it can be checked that all neuron models show an increasing performance when the number of hidden neurons is increased, as can reasonably be expected. In conclusion, for all of the architectures it is clear that the use of heterogeneous neuron models leads to higher accuracy rates in the training and test sets. Moreover, when imprecision is allowed by accepting that each value is endowed with the above mentioned uncertainty, the fuzzy heterogeneous model also outperforms its crisp counterpart.

## 4.5 Presentation of the Results (II)

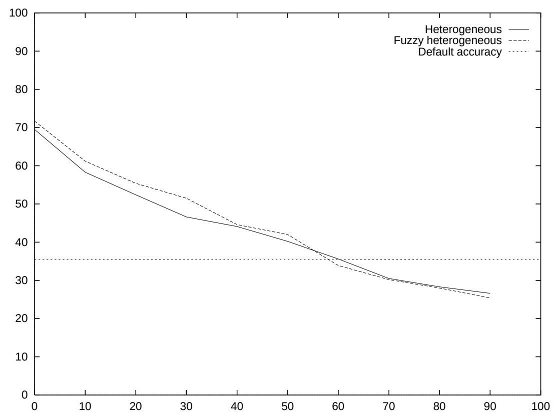
The neural nets obtained in the previous experiment can now be used to assess the effect of factor (c), the influence of missing values in the data. The purpose of this experiment is twofold: first, it is useful studying to what extent missing information degrades performance. This is an indication of robustness and is important from the point of view of the *methods*. Second, in this particular problem, studying the effect of missing data is very interesting, because it can give an answer to the following questions:

1. What *predictive* performance could we expect if we do not supply all the information? (and just a fraction of it).
2. What would have happened had we presented to the net *incomplete training* information from the outset?

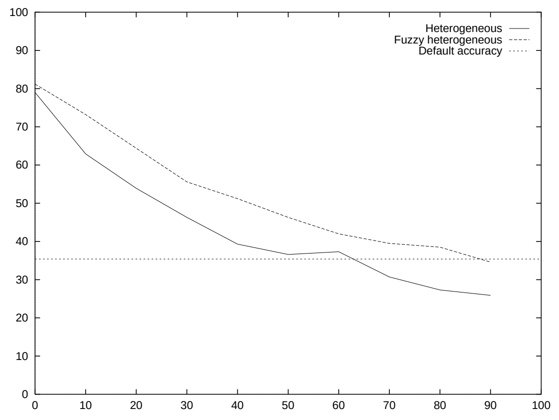
This scenario makes sense in our case study, for which a rich set of complete data may be impossible to obtain, because of lack or damage of resources, physical or practical unrealizability, lack of time, climatic conditions, etc. Note that it is not that a particular variable cannot be measured (we could readily remove it) but that *some* realizations of (potentially) *all* variables may be missing. These experiments were performed with the same nets found in the previous section. This time, however, they were each run on different test sets, obtained by artificially and randomly (with



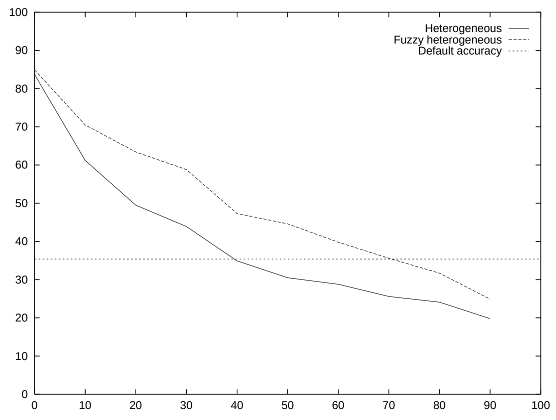
(a) $5_h$ and $5_f$



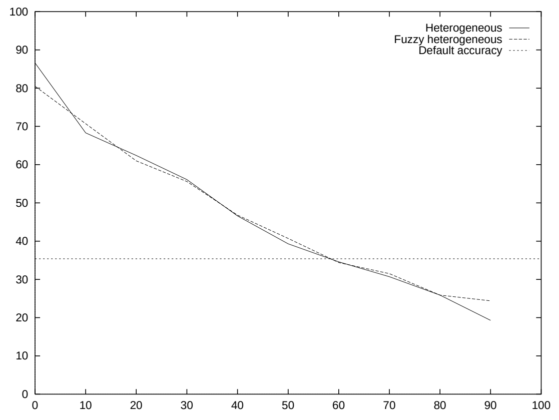
(b) $2_h5_n$ and $2_f5_n$



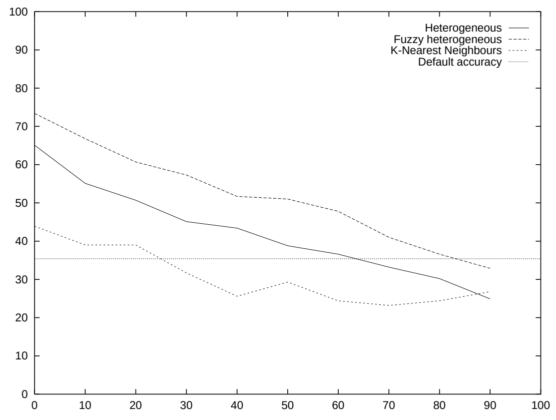
(c) $4_h5_n$ and $4_f5_n$



(d) $6_h5_n$ and $6_f5_n$



(e) $8_h5_n$ and $8_f5_n$



(f) $6_h5_n$ and $6_f5_n$

**Fig. 2.** Increasing presence of missing data in test. Mean test classification accuracy for the heterogeneous ($p_h5_n$) and fuzzy heterogeneous ($p_f5_n$) families. (a) $5_h$ and $5_f$ (b) $2_h5_n$ and $2_f5_n$ (c) $4_h5_n$ and $4_f5_n$ (d) $6_h5_n$ and $6_f5_n$ (e) $8_h5_n$ and $8_f5_n$ (f) Mean test classification accuracy for $6_h5_n$ and $6_f5_n$ when trained with a 30% of missing information. See text for an explanation of axis.

a uniform distribution) adding different percentages of missing information. These percentages range from 10% to 90%, in intervals of 10%. The results are presented, for the whole set of heterogeneous architectures[1] displayed in table 1, in a graphical form, through figs. 2 (a) to 2 (e). The $x$-axis represents the total percentage of missing values in the test set, while the $y$-axis stands for the $\mathrm{MCA}_{\mathrm{TE}}$ (that is, again, data shown for each point is the average for $R = 5$ runs). The horizontal line represents the size of the major class (35.4%) to be taken as a reference, and the same $k$-nearest neighbours algorithm is run and shown in fig. 2 (a).

### 4.6 Analysis of the Results (II)

Both neuron models $h, f$ are very robust, a fact that shows in the curves, which follow a quasilinear decay. The accuracies are consistently higher for the fuzzy model than for the crisp counterpart for all the network architectures, again showing that allowing imprecision increases effectiveness and robustness. Performance, in general, is well above the default accuracy until a $50\% - 60\%$ of missing information is introduced. In many cases, mean classification accuracy is still above for as far as $70\% - 90\%$, which is very remarkable. This graceful degradation of fuzzy heterogeneous models should not be overlooked, since it is a very desirable feature in any model willing to be useful in real-world problems.

The last figure –fig. 2 (f)– shows the effect of a different training outset. Choosing what seems to be the best group of architectures for the given problem, the $6_h5_n$ and $6_f5_n$, these networks were trained again, this time with a modified training set: adding to it a 30% of missing information, in the same way it was done for the test set, and using them again to predict the increasingly diluted test sets. As usual, the horizontal line represents the size of the major class and $k$-nearest neighbours performance is also shown. Training and test accuracies were this time lower (as one should expect) and equal to $\mathrm{MCA}_{\mathrm{TR}} = 88.8\%$ for $6_h5_n$ and to $\mathrm{MCA}_{\mathrm{TR}} = 96.3\%$ for $6_f5_n$. However, the differences with previous performance are relatively low. Some simple calculations show that, although the amount of data is 70% that of the previous situation, new accuracies are 97.3% and 96.3% of those obtained with full information for $6_h5_n$ and $6_f5_n$, respectively. Performance in test sets is also noteworthy: although the new curves begin at a lower point than before, the degradation is still quasilinear. What is more, the slope of this linear trend is lower (in absolute value), resulting in a slight raising up of the curves (in both of them).

## 5 Conclusions

Experiments carried out with data coming from a real-world problem in the domain of environmental studies have shown that allowing imprecise inputs, and using fuzzy heterogeneous neurons based on similarity, yields much better prediction indicators –mean accuracies, mean errors and their variances and absolute best models found– than those from classical crisp real-valued models. These results for heterogeneous

[1] These experiments could not be performed for the $p_n5_n$ architectures, for they do not accept missing information. Although there are estimation techniques, they are not an integrated part of the models, and would have introduced a bias.

networks confirm the features observed in other studies [1] [2] [3] [11] [12] concerning their mapping effectiveness and their robustness with respect to the presence of uncertainty and missing data. Their ability to consider directly imprecise data and their performance under those circumstances deserve closer attention, due to their implications for real-world problems from the point of view of neurofuzzy systems. However, the study of these networks is still in its initial stage. Several other architectures are possible, along with different (partial) similarity measures, and further investigations are being made in order to explore in more extent their properties, and to make the scope of their application more precise.

## References

1. Ll. Belanche and J.J. Valdés. "Using Fuzzy Heterogeneous Neural Networks to Learn a Model of the Central Nervous System Control". In Procs. of EUFIT'98, 6th European Congress on Intelligent Techniques and Soft Computing, pp. 1858-62. Elite Foundation. Aachen, Germany, 1998.
2. Ll. Belanche, J.J. Valdés and R. Alquézar. "Fuzzy Heterogeneous Neural Networks for Signal Forecasting". In Procs. of ICANN'98, Intl. Conf. on Natural and Artificial Neural Networks (Perspectives in Neural Computing), pp. 1089-94. Skövde, Sweden. Springer-Verlag, 1998.
3. Ll. Belanche, J.J. Valdés, J. Comas, I.-R. Roda and M. Poch. "Modeling the Input-Output Behaviour of Wastewater Treatment Plants using Soft Computing Techniques". In Procs. of BESAI'98, Binding Environmental Sciences and AI, held as part of ECAI'98, European Conference on Artificial Intelligence, pp. 81-94, Brighton, UK, 1998.
4. Chandon, J.L, Pinson, S: Analyse Typologique. Théorie et Applications. Masson, 1981.
5. Dubois D., Esteva F., García P., Godo L., Prade H.: A logical approach to interpolation based on similarity relations. Instituto de Investigación en Inteligencia Artificial. Consejo Superior de Investigaciones Científicas, Barcelona, España. Research Report IIIA 96/07, 1996.
6. Dubois D., Prade H., Esteva F., García P., Godo L., López de Mántaras R: Fuzzy set modelling in case-based reasoning. *Int. Journal of Intelligent Systems* (in press) (1997).
7. Gower, J.C. A General Coefficient of Similarity and some of its Properties. *Biometrics 27, 857-871*, 1971
8. Fagundo, J.R, Valdés J.J, Rodríguez, J. E.: Karst Hydrochemistry (in Spanish). *Research Group of Water Resources and Environmental Geology, University of Granada*, Ediciones Osuna, pp 212, Granada, Spain, 1996.
9. Fagundo, J.R, Valdés J.J, Pulina, M.: Hydrochemical investigations in extreme climatic areas, Cuba and Spitzbergen. *In: Water Resources Management and Protection in Tropical Climates*, pp 45-54, Havana, Stockholm, 1990.
10. G.J. Klir, T.A. Folger.: Fuzzy Sets, Uncertainty and Information. Prentice Hall Int. Editions, 1988.
11. Valdés J.J, García R.: A model for heterogeneous neurons and its use in configuring neural networks for classification problems. In Procs. of IWANN'97, International World Conference on Artificial and Natural Neural Networks. Lecture Notes in Computer Science 1240, pp. 237-246. Springer Verlag, 1997.
12. Valdés J.J., Belanche Ll., Alquézar R. Fuzzy heterogeneous neurons based on similarity. *International Journal of Intelligent Systems* (accepted for publication, 1999). Also in Procs. of CCIA'98: Congrés Català per a la Intel.ligència Artificial (Catalan Congress for Artificial Intelligence), Tarragona, Spain, 1998. Also in LSI Research Report LSI-98-33-R. Universitat Politècnica de Catalunya, Barcelona (1998).
13. Goldberg, D.E.: Genetic Algorithms for Search, Optimization & Machine Learning. Addison-Wesley (1989).
14. Davis, L.D.: Handbook of Genetic Algorithms. Van Nostrand Reinhold (1991).
15. Zimmermann H.J.: Fuzzy set theory and its applications. Kluver Academic Publishers (1992).