Similarity networks for classification: a case study in the Horse Colic problem

Lluís Belanche[*1] and Jerónimo Hernández[2]

[1]Software Department, Computer Science School
Technical University of Catalonia, Barcelona (Spain)
[2]Intelligent Systems Group
Department of Computer Science and Artificial Intelligence
University of the Basque Country, Donostia (Spain)
belanche@lsi.upc.edu
jeronimo.hernandez@ehu.es

Abstract. This paper develops a two-layer neural network in which the neuron model computes a user-defined *similarity function* between inputs and weights. The neuron transfer function is formed by composition of an adapted logistic function with the mean of the partial input-weight similarities. The resulting neuron model is capable of dealing directly with variables of potentially different nature (continuous, fuzzy, ordinal, categorical). There is also provision for missing values. The network is trained using a two-stage procedure very similar to that used to train a radial basis function (RBF) neural network. The network is compared to two types of RBF networks in a non-trivial dataset: the Horse Colic problem, taken as a case study and analyzed in detail.

Keywords: Similarity measures; Neural networks; Horse Colic problem

1 Introduction

The intuitive notion of *similarity* is very useful to group objects under specific criteria and has been used with great success in several fields within or related to Artificial Intelligence, like Case Based Reasoning [1], Information Retrieval [2] or Pattern Matching [3]. Under the conceptual cover of similarity, we develop a class of neurons that accept heterogeneous inputs and weights and compute a user-defined similarity function between these inputs and weights. The neuron transfer function is the composition of a parameterized sigmoid-like function adapted to the $[0,1]$ interval taking the averaged vector of partial input-weight similarities as argument. The basic idea is that a combination of similarity functions, comparing variables independently, is more capable of catching better the singularity of an heterogeneous dataset than other methods which require *a priori* data transformations. The resulting neuron model then accepts mixtures of

---

[*] Corresponding author.

continuous and discrete quantities, with explicit provision for missing information. Other data types are possible by extension of the model. The network is compared to two types of radial basis function (RBF) networks in the Horse Colic dataset, which is analysed in detail and used in two different classification tasks. At least for one of the tasks, the results point to a clear improvement in generalization performance. An appealing advantage is found in the enhanced interpretability of the learned weights, so often neglected in the neural network community.

The paper is organized as follows. Section 2 further motivates the basis of the approach and reviews previous work in relation to similarity measures and data heterogeneity; section 3 develops a clustering algorithm fully based on similarity measures; section 4 builds the similarity neural network. Finally, section 5 presents experimental work.

2   Preliminaries

2.1   Data types and missingness

In many important domains from the real world, objects are described by a mixture of continuous and discrete variables, usually containing missing information and characterized by an underlying uncertainty or imprecision. For example, in the well-known UCI repository [4] over half of the problems contain explicitly declared categorical attributes, let alone other data types, usually unreported. In the case of artificial neural networks (ANN), this *heterogeneous* information has to be encoded in the form of real-valued quantities, although in most cases there is enough domain knowledge to characterize the nature of the variables.

The integration of heterogeneous data sources in information processing systems has been advocated elsewhere [5]. In this sense, a shortcoming of the existent neuron models is the difficulty of adding prior knowledge to the model in a principled way. Current practice assumes that input vectors may be faithfully represented as a point in $\mathbb{R}^n$, and the geometry of this space is meant to capture the meaningful relations in input space. There is no particular reason why this should be the case. Moreover, the activity of the units should have a well defined meaning in terms of the input patterns [6]. Without the aim of being exhaustive, commonly used coding methods are (see, e.g. [7]):

Ordinal variables coded as real-valued or using a *thermometer* scale.
Categorical variables with c modalities are coded using a binary expansion representation (also known as a 1-out-of-c code).
Vagueness and uncertainty are considerations usually put aside.
Missing information is difficult to handle, specially when the lost parts are of significant size. Typical approaches remove the involved examples (or variables) or "fill in the holes" with the mean, median or nearest neighbor value. Statistical approaches need to model the input distribution itself [8], or are computationally very intensive [9].

Although these encodings may be intuitive, their precise effect on network performance (very specially in relation to overfitting) is far from clear. This is due to the change in input distribution, the increase (sometimes acute) in input dimension and other subtler effects, derived from imposing an order or a continuum where there was none.

2.2   Similarity measures

Let us represent patterns belonging to a space $X \neq \emptyset$ as a vector x of n components, where each component x_k represents the value of a particular feature k. A *similarity measure* is a unique number expressing how "like" two patterns are, given these features. It can be defined as an upper bounded, exhaustive and total function $s : X \times X \to I_s \subset R$ with $|I_s| > 1$ (therefore I_s is upper bounded and $s_{max} \equiv \sup_{\mathbb{R}} I_s$ exists). A similarity measure may fulfill many properties, like:

Reflexivity: $s(x, y) = s_{max} \Leftrightarrow x = y$.

Symmetry: $s(x, y) = s(y, x)$.

Lower boundedness: $\exists a \in R$ such that $s(x, y) \geq a$, for all $x, y \in X$ (note this is equivalent to ask that $\inf_{\mathbb{R}} I_s$ exists).

Closedness: given a lower bounded s, $\exists x, y \in X$ such that $s(x, y) = \inf_{\mathbb{R}} I_s$ (equivalent to ask that $\inf_{\mathbb{R}} I_s \in I_s$).

These axioms should be taken as *desiderata*. Some similarity relations may fulfill part or all of them [10]. Other properties (like transitivity) may be of great interest in some contexts, but are not relevant for this work. However, it is not difficult to show that reflexivity implies a basic form of transitivity [11].

2.3   Similarity measures for different variable types

We present in this section specific similarity measures defined in a common co-domain $I_s = [0, 1]$. Not only it is possible to find different types of variables, also different similarity measures could be used for different variables of the same type. For notational convenience, we use s_{ijk} to mean $s_k(x_{ik}, x_{jk})$.

Nominal variables It is assumed that no partial order exists among these values and the only possible comparison is equality. The basic similarity measure for these variables is the *overlap*. Let x_{ik}, x_{jk} be the modalities taken by two examples x_i, x_j, then $s_{ijk} = 1$ if $x_{ik} = x_{jk}$ and 0 otherwise.

We introduce in this paper a frequency-based approach that goes beyond this simple equal/not-equal scheme:

$$s_{ijk} = \begin{cases} 0 & \text{if } x_{ik} \neq x_{jk} \\ 1 - P_{ik} & \text{if } x_{ik} = x_{jk} \end{cases} \tag{1}$$

where P_{lk} is the *fraction* of examples in a sample that take the value x_{lk} for variable k (ideally, one could use the *probability* of this event, if this knowledge is available). Therefore, if the values are different, there is not similarity. If they happen to be equal, then the similarity is inversely proportional to their probability. For instance, if two patients are being compared on their current illness, both having a rare illness makes them more similar than both having a very common one. Other ways of inverting the probability are possible. In the absence of further knowledge, the linear one is the simplest choice.

Ordinal variables These variables can be seen as a bridge between categorical and continuous variables. It is assumed that the values of the variable form a linearly ordered space $(\mathcal{O}, \preceq)$. Let $x_{ik}, x_{jk} \in \mathcal{O}$, such that $x_{ik} \preceq x_{jk}$, and P_{lk} be defined as above. Then,

$$s_{ijk} = \frac{2 \log(P_{ik} + \ldots + P_{jk})}{\log P_{ik} + \log P_{jk}} \tag{2}$$

where the summation runs through all the ordinal values x_{lk} such that $x_{ik} \preceq x_{lk}$ and $x_{lk} \preceq x_{jk}$ [12].

Continuous variables Let $x_{ik}, x_{jk} \in A = [r^-, r^+] \subset \mathbb{R}, r^+ > r^-$. The standard metric in $\mathbb{R}$ is a metric in A. Therefore, for any two values $x_{ik}, x_{jk} \in A$:

$$s_{ijk} = \hat{s}\left(\frac{|x_{ik} - x_{jk}|}{r^+ - r^-}\right) \tag{3}$$

where $\hat{s} : [0,1] \longrightarrow [0,1]$ is a decreasing continuous function. A very simple family is $\hat{s}(z) = (1 - z^\beta)^\alpha, 0 < \beta \le 1, \alpha \ge 1$. We use here the simplest choice $\alpha = \beta = 1$.

Fuzzy variables For variables representing fuzzy sets, similarity relations from the point of view of fuzzy theory have been defined elsewhere [13], and different choices are possible. In possibility theory, the *possibility* expresses the likeliness of co-occurrence of two vague propositions, with a value of one standing for absolute certainty. For two fuzzy sets $\tilde{A}, \tilde{B}$ of a reference set X, it is defined as:

$$\Pi_{(\tilde{A})}(\tilde{B}) = \sup_{u \in X} \left(\mu_{\tilde{A} \cap \tilde{B}}(u)\right) = \sup_{u \in X} \left(\min\left(\mu_{\tilde{A}}(u), \mu_{\tilde{B}}(u)\right)\right)$$

In our case, if $\mathcal{F}_k$ is an arbitrary family of fuzzy sets, and $x_{ik}, x_{jk} \in \mathcal{F}_k$, the following similarity relation can be used $s_{ijk} = \Pi_{(x_{ik})}(x_{jk})$.

2.4 Missing value treatment

Missing information is a recurrent problem in data analysis because there are many causes for the absence of a value. The problem acquires more relevance when significant parts of a data sample are lost or unknown. There are basically three ways of dealing with *missing values*: fill in the examples, extend the learning methods to cope with incomplete data or discard the examples (or the variables) with missing values. We advocate for the second possibility, for which there exist some possible approaches:

1. The first proposal is based on Gower's general similarity measure [14]:

$$S_G(\boldsymbol{x}_i, \boldsymbol{x}_j) = \frac{\sum\limits_{k=1}^{n} s_{ijk}\delta_{ijk}}{\sum\limits_{k=1}^{n} \delta_{ijk}} \tag{4}$$

   where s_k is a partial similarity function to be aggregated and $\delta_{ijk} \in \{0,1\}$ is equal to 0 every time s_{ijk} is *missing* (because one or both of x_{ik}, x_{jk} are missing). It is not difficult to realize that this is equivalent to the replacement of the missing similarities by the average of the non-missing ones. Therefore, the conjecture is that the missing values, if known, would not change the overall similarity.
2. The second proposal is even simpler: to replace the missing partial similarity measures by a constant quantity, namely the midpoint of the similarity co-domain I_s. For instance, when $I_s = [0,1]$, this constant would be $\frac{1}{2}$. Doing this we are assuming that the missing similarities, if known, would make the example as similar to any other example as the average similarity.

Both methods look very naive and indeed they are; on the other hand, they are very intuitive and computationally simple. The appealing trait of these two approaches is that they do not try to estimate the missing information (a delicate and risky undertaking) but to estimate the *overall* similarity between two observations, given that some of the partial similarities could not be computed. We argue that this second task is easier and, after all, is what we really are interested in: the *similarity value*. This is the reason why we consider missing value treatment together with the construction of the overall similarity value.

2.5 Normalized aggregation of similarities

When we aggregate (e.g. by averaging) the partial similarities we are assuming that all of them have the same importance. However, each partial similarity covers its co-domain $[0,1]$ in a different way. The partial similarities that accumulate on the upper half of the interval have more influence in the overall value, because they do a more important contribution to the aggregation. We argue that this biased behavior should be corrected so that all the partial similarities have a common baseline.

Let $s_{..k}$ be the mean similarity among all examples in the analyzed data sample, according to variable k only. We first rescale all the similarities as $\hat{s}_{ijk} = \frac{s_{ijk}}{s_{..k}}$. Then a normalization function $n : (0, +\infty) \to (0,1)$ is applied:

$$n(z) = \frac{z^a}{z^a + 1} \tag{5}$$

where a conveniently controls the shape of the function. When a similarity computation is needed, it is calculated as $n(\hat{s}_{ijk})$ instead of s_{ijk}. The decision on missing values in section 2.4 can now be better justified. The similarity between two elements x_{ik}, x_{jk} is now computed as:

$$s_{ijk} = \begin{cases} n\left(\frac{s(x_{ik},x_{jk})}{s_{..k}}\right) & \text{if neither of } x_{ik}, x_{jk} \text{ are missing} \\ \frac{1}{2} & \text{otherwise} \end{cases} \tag{6}$$

This is so because, when $s_{..k}$ is used to replace the missing similarity *value*, we have $n(\frac{s(x_{ik},x_{jk})}{s_{..k}}) = n(\frac{s_{..k}}{s_{..k}}) = \frac{1}{2}$ (this holds regardless of the value of a).

3 Clustering similarity data

In a clustering task the examples are grouped attending to some similarity measure. The LEADER algorithm is a simple and attractive unsupervised clustering method [15]. In essence, the algorithm processes the examples of the dataset taking one at a time and evaluating if it can belong to any cluster already created. If it cannot, a new cluster will be created using this new example as leader.

We have developed a new LEADER 2 version of the algorithm that represents an improvement in two ways. First, the algorithm now works using general similarities instead of metric distance functions. Second, given s_0, the method is guaranteed to fulfill a number of interesting properties:

1. For any example, the similarity with its leader is *at least* s_0.
2. The similarity between any two leaders is *lower* than s_0.
3. If two examples are repeated in the dataset, they will have the *same* leader.
4. For any example, the similarity with its leader is *higher* than that with any other leader.

One immediate consequence of these properties is that the lowest similarity of an example with its leader will be higher than the highest similarity between two different leaders.

In summary, the algorithm needs the specification of one parameter ($s_0 \in I_s$) and the returned leaders are a subset of the data set (thus there is no problem in delivering "impossible centroids" as many algorithms do). The number of clusters cannot be estimated beforehand, but it is possible to establish a relationship with the s_0 parameter: a higher s_0 implying a higher number of clusters.

4 Similarity Neural Networks

4.1 The S-Neuron Model

Consider $s : \mathcal{H}^n \times \mathcal{H}^n \to I_s$ a similarity function in $\mathcal{H}^n = \mathcal{H}^{(1)} \times \ldots \times \mathcal{H}^{(n)}$, the cartesian product of an arbitrary number n of *source sets*, where $I_s = [0,1]$. This function is formed by combination of n partial similarities $s_k : \mathcal{H}^{(k)} \times \mathcal{H}^{(k)} \to I_s$, $k = 1, \ldots, n$, each $\mathcal{H}^{(k)}$ being the domain of the predictive variable k.

The s_k are normalized to a common real interval ($[0,1]$ in this case) and computed according to different formulas for different variables (possibly but not

necessarily determined by variable type alone). A *neuron model* can be devised that is both similarity-based and handles data heterogeneity and missing values, as follows. Let $\Sigma_i(\boldsymbol{x})$ the function computed by the i-th neuron, where $\boldsymbol{x} \in \mathcal{H}^n$ having a weight vector $\mu_i \in \mathcal{H}^n$ and smoothing parameter p_i, defined as:

$$\Sigma_i(\boldsymbol{x}) = f(s(\boldsymbol{x}, \mu_i), p_i), \quad \text{with } s(\boldsymbol{x}, \mu_i) = \frac{1}{n} \sum_{k=1}^{n} s_k(x_k, \mu_{ik}) \tag{7}$$

This S-neuron adds a non-linear *activation* function to the linearly aggregated similarities. Such function could be any sigmoid-like automorphism (a monotonic bijection) in $[0, 1]$. In particular, we consider the simple family of functions:

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

$$a(p) = \frac{-0.5 + \sqrt{0.5^2 + 4p}}{2} \tag{8}$$

where $p > 0$ is a parameter controlling the shape of the function (Fig. 1). The function fulfills $\forall p \in \mathbb{R}^+,\ f(0, p) = 0,\ f(1, p) = 1,\ \lim_{p \to \infty} f(x, p) = x$ and $f(x, 0) = H(x - 0.5)$, being H the Heaviside function.
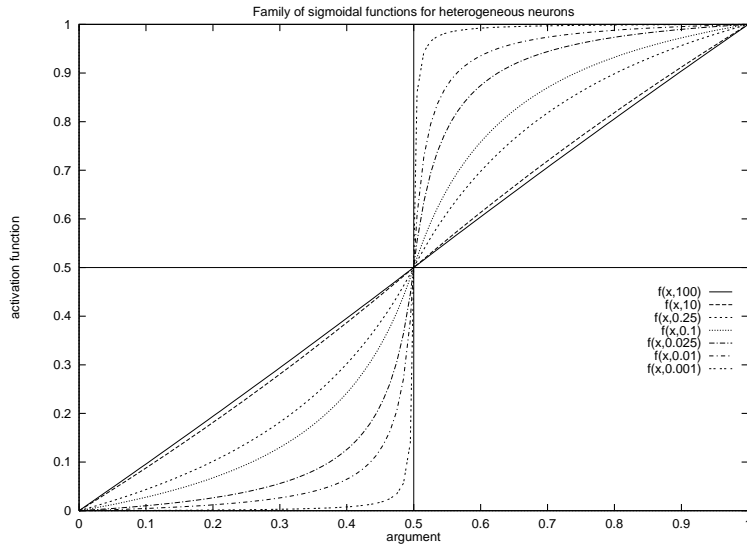

Fig. 1: Family of sigmoidal functions $f(\cdot, p)$ for different values of p.

4.2   Similarity Neural Networks

Similarity neural networks (SNN) are neural architectures built out of the previously defined S-neurons, thus allowing for heterogeneous or missing inputs. A feed-forward architecture, with a hidden layer composed of heterogeneous neurons and a linear output layer is a straightforward choice, thus conforming a *hybrid* structure. The k-th output neuron of the SNN computes the function:

$$f_k(\boldsymbol{x}) = \sum_{i=1}^{h} w_{ki} \Sigma_i(\boldsymbol{x}) + w_{k0}, \ k = 1, \ldots, m$$

where $h > 0$ is the number of hidden S-neurons, m is the number of outputs and $\{w_{ki}\}$ is the set of mixing coefficients. The SNN thus keeps linearity in the output layer and interpretability in the hidden layer. It can be naturally seen as a generalization of the RBF. This is so because the response of hidden neurons is localized: centered at a given object (the neuron weight, where response is maximum), falling down as the input is less and less similar to this center.

4.3   Training the SNN

Let $\{(\boldsymbol{x}_l, y_l)\}_{l=1}^{N}$ represent a training data sample. Since the SNN is a two-layer feed-forward neural network with local computation units in the first layer, training can be solved very efficiently in a two-stage procedure, as detailed next:

First layer weights The first layer centers are a subset of the examples in the sample dataset. These centers are chosen to be the cluster *leaders* returned by the LEADER 2 clustering algorithm acting on the set $\{\boldsymbol{x}_l\}_{l=1}^{N}$ (hence h is set to the number of leaders). This algorithm uses the user-defined similarity as explained in previous sections.

The value of p Based on the information delivered by the clustering, we associate the *compactness* of a cluster with a greater slope of the f function (a lower p, Fig. 1). When a cluster is more compact (there is a big number of examples with a high similarity with the leader), it is easier to decide whether a new example belongs to that cluster or not because the cluster is well-defined and the limits are clear. This situation corresponds to $f(\cdot, p)$ working similar to a Heaviside function ($p \to 0$). This behavior can be achieved by computing first a *relative compactness index*:

$$\chi_i = \frac{m_i l_i}{m_i l_i + \alpha \overline{m}\, \overline{l}} \tag{9}$$

where l_i is the number of examples in cluster i and m_i is the average similarity of these examples to their leader; the quantities $\overline{m}$ and $\overline{l}$ are the corresponding global averages (across the whole clustering). This index can be used to obtain the smoothing parameter required in eq. 7 as $p_i = -\ln \chi_i$. The value of α is set at $\exp(0.1) - 1$, which is related to the value of $p = 0.1$ corresponding to an "average" compact cluster (relative to the current clustering).

Second layer weights Regularization is a technique that incorporates additional information to the fit, usually a complexity penalty to prevent overfitting.

$$SSE_\lambda = \sum_{i=1}^{N} \sum_{k=1}^{m} (y_{ki} - f_k(\boldsymbol{x}_i))^2 + \lambda \sum_{j=1}^{h} \sum_{k=1}^{m} w_{kj}^2$$

where the first term is the sum of squared errors and the second is the regularization term (known as *ridge regression*, in this case). The minimization of SSE_λ forces to compensate smaller errors against smaller weights. We define the $H = (h_{ij})$ matrix as $h_{ij} = \Sigma_j(\boldsymbol{x}_i), i = 1, \ldots, N, j = 0, \ldots, h$.

$$H = \begin{bmatrix} \Sigma_0(\boldsymbol{x}_1) & \Sigma_1(\boldsymbol{x}_1) & \Sigma_2(\boldsymbol{x}_1) & \ldots & \Sigma_h(\boldsymbol{x}_1) \\ \Sigma_0(\boldsymbol{x}_2) & \Sigma_1(\boldsymbol{x}_2) & \Sigma_2(\boldsymbol{x}_2) & \ldots & \Sigma_h(\boldsymbol{x}_2) \\ \ldots & \ldots & \ldots & \ddots & \vdots \\ \Sigma_0(\boldsymbol{x}_N) & \Sigma_1(\boldsymbol{x}_N) & \Sigma_2(\boldsymbol{x}_N) & \ldots & \Sigma_h(\boldsymbol{x}_N) \end{bmatrix} \tag{10}$$

where $\Sigma_0(\cdot) = 1$. Let $A = H^T H + \lambda I$, $P = I - HA^{-1}H^T$, and y the vector of outputs, where I is an identity matrix of appropriate dimensions. The optimal weight vector is $\boldsymbol{w}^* = A^{-1}H^T y$, the minimizer of SSE_λ for a certain λ. The *Generalized Cross Validation* error is:

$$GCV = \frac{N y^T P^2 y}{(Tr\,(P))^2} \tag{11}$$

When the derivative of GCV is set to zero, the resulting equation can be manipulated so that one λ appears isolated in one side of the equation. The value of λ can be re-estimated iteratively until convergence [16], using

$$\lambda = \frac{Tr\,\left(A^{-1} - \lambda A^{-2}\right) \cdot y^T P^2 y}{Tr\,(P) \cdot (\boldsymbol{w}^*)^T A^{-1} \boldsymbol{w}^*} \tag{12}$$

An initial guess for λ is used to evaluate the updating expression, which produces a new guess. The obtained sequence converges to a local minimum of GCV. In this work, this initial set is $\lambda \in \{10^{-6}, 10^{-3}, 1\}$. In addition, a maximum number of 100 iterations is set.

5   A case study: the Horse Colic problem

We develop in this section a fully worked application example on a challenging dataset. This problem has been selected as characteristic of modern modeling problems because of the diversity in data heterogeneity and the presence of missing values [17]. This dataset is made available at the UCI repository [4] and created by M. McLeish and M. Cecile (Computer Science Department, University of Guelph, Ontario, Canada). Each example is the clinical record of a horse. The attributes (variables) are specially well documented. The number of examples

is modest, and therefore the chances of overfitting are increased due to a bad
pre-processing. In summary, there are 368 examples described by 28 attributes
(continuous, discrete, and categorical) and a 30% of missing values.

5.1   The Horse Colic dataset

The problem description and the dataset itself are taken from the UCI repository
[4]. The available documentation is analyzed for an assessment on the more
appropriate treatment. Missing information is also properly identified. There are
several possible tasks that can be chosen for this dataset. The two most common
settings are the prediction of attributes 23 ('what happened to the horse?') and
24 ('was the problem (lesion) surgical?'), using attributes 1,2 and 4 to 22 as
predictors. We call these two separate tasks HC23 and HC24, respectively.

In task HC23 there are two examples less because these two examples have
a missing value in the class variable. In accordance to the documentation, at-
tributes 3 and 28 are not used because they do not provide useful information.
Attributes 25, 26 and 27 ('type of lesion?') are also discarded because they repre-
sent alternative class variables. It should be noted that the missing value counts
are based on the full dataset. Class distribution is as follows:

HC23 : what eventually happened to the horse (lived - 61.5%, died - 24.3% or
was euthanized - 14.2%)

HC24 : was the lesion surgical? (yes - 63% or no - 37%).

The following list details the used attributes, their characteristics and the
decision taken on the type of attribute.

– Variable 1: **Surgery?** (Yes, it had surgery; It was treated without surgery)
  Comments: none.
  Decision: `Categorical`.
– Variable 2: **Age** (Adult horse; Young ($<$ 6 months))
  Comments: none.
  Decision: `Categorical`.
– Variable 3: **Hospital number**
  Comments: the case number assigned to the horse.
  Decision: REMOVED.
– Variable 4: **Rectal temperature**
  Comments: Temperature of the horse in degrees Celsius.
  Decision: `Continuous`.
– Variable 5: **Pulse**
  Comments: The heart rate in beats per minute.
  Decision: `Continuous`.
– Variable 6: **Respiratory rate**
  Comments: none.
  Decision: `Continuous`.
– Variable 7: **Temperature of extremities** (Normal; Warm; Cool; Cold)
  Comments: an indicator of the peripheral circulation. The values are re-ordered
  as: cold, cool, normal, warm.
  Decision: `Ordinal`.

– Variable 8: **Peripheral pulse** (Normal; Increased; Reduced; Absent)
   Comments: The values are re-ordered as: Absent; Reduced; Normal; Increased.
   Decision: `Ordinal`.
– Variable 9: **Mucous membranes** (normal pink; bright pink; pale pink; pale cyan-otic; bright red / injected; dark cyanotic)
   Comments: a measurement of membrane color. Could it be considered ordinal?
   Decision: `Categorical`.
– Variable 10: **Capillary refill time** (< 3 seconds; $>= 3$ seconds)
   Comments: could have been a continuous variable originally.
   Decision: `Categorical`.
– Variable 11: **Pain** (alert, no pain; depressed; intermittent mild pain; intermittent severe pain; continuous severe pain)
   Comments: a subjective judgment of the horse's pain level. Despite donor's advice, these values are clearly ordered, so we consider it to be an ordinal variable ('the more painful, the more likely it is to require surgery').
   Decision: `Ordinal`.
– Variable 12: **Peristalsis** (hypermotile; normal; hypomotile; absent)
   Comments: an indication of the activity in the horse's gut (note order has to be reversed).
   Decision: `Ordinal`.
– Variable 13: **Abdominal distension** (none; slight; moderate; severe)
   Comments: none.
   Decision: `Ordinal`.
– Variable 14: **Nasogastric tube** (none; slight; significant)
   Comments: it refers to any gas coming out of the tube.
   Decision: `Ordinal`.
– Variable 15: **Nasogastric reflux** (none; <1 liter; >1 liter)
   Comments: none.
   Decision: `Ordinal`.
– Variable 16: **Nasogastric reflux PH**. Comments: none.
   Decision: `Continuous`.
– Variable 17: **Rectal examination - feces** (normal; increased; decreased; absent)
   Comments: The values are re-ordered as: absent; decreased; normal; increased. Could it be considered categorical?
   Decision: `Ordinal`.
– Variable 18: **Abdomen** (normal; firm feces in the large intestine; distended small intestine; distended large intestine; other)
   Comments: none.
   Decision: `Categorical`.
– Variable 19: **Packed cell volume**
   Comments: the number of red cells by volume in the blood.
   Decision: `Continuous`.
– Variable 20: **Total protein**
   Comments: none.
   Decision: `Continuous`.
– Variable 21: **Abdominocentesis appearance** (clear; cloudy; serosanguinous)
   Comments: appearance of fluid obtained from the abdominal cavity.
   Decision: `Categorical`.
– Variable 22: **Abdominocentesis total protein**
   Comments: none.
   Decision: `Continuous`.

After this process, the dataset is described by 21 variables: 6 categorical, 7 continuous and 8 ordinal.

5.2   Experimental settings

The SNN is compared to two RBFs, as described next:

RBFk: a standard RBF where the centers are decided using the k-means clustering algorithm, and the hidden-to-output weights are set by solving the regularized least squares problem. The value of the smoothing parameter σ^2 is set according to the method described in [8].

RBF2: a standard RBF where the centers are decided using the LEADER 2 clustering algorithm, and the hidden-to-output weights are set by solving the regularized least squares problem. The value of the smoothing parameter is set in the same way than for the RBFk.

The reason for choosing two RBFs instead of one lies in the interest in assessing any differences due to the clustering algorithm, since the LEADER 2 algorithm can also be used to set the centers of a standard *RBF* network. This way it is easier to separate the effect of the similarity processing. The two RBFs need a pre-processing of the data, carried out following the recommendations in [7]. The input variables for the RBFs are then standardized (to zero mean, unit standard deviation). This is not needed by the SNN, but is beneficial for the RBF methods. The values of s_0 for the SNN and the RBF2 as well as the value for k in k-means for the RBFk are chosen after some preliminary trials.

The resampling method used in this work is based on Dietterich [18]. This method consists in five repetitions of two-fold cross-validation (5×2 CV), returning ten test set performance estimations, that are combined as:

$$ t = \frac{p_1^{(1)}}{\sqrt{\frac{1}{5}\sum_{i=1}^{5} s_i^2}} \qquad (13) $$

where $p_i^{(j)} = p_i^{(j)}[A] - p_i^{(j)}[B]$ is the difference between the proportions of the two methods (A, B) being compared, in partition (i) of replication (j), for $i \in \{1, \dots, 5\}$ and $j \in \{1, 2\}$; then we have the estimated variances $s_i^2 = \left(p_i^{(1)} - \overline{p_i}\right)^2 + \left(p_i^{(2)} - \overline{p_i}\right)^2$, where $\overline{p_i} = (p_i^{(1)} + p_i^{(2)})/2$.

A paired t test can then be computed to assess statistical significance in any possible differences in performance. The hypothesis of both methods having the same error rate can be rejected at the 95% level when $t > 2.571$.

As an alternative, Alpaydin defends the use of an F test, where all the differences are combined [19]:

$$ F = \frac{\sum_{i=1}^{5}\sum_{j=1}^{2}\left(p_i^{(j)}\right)^2}{2\sum_{i=1}^{5} s_i^2} \qquad (14) $$

The hypothesis of both methods having the same error rate can be rejected at the 95% level when $F > 4.74$. This approach combines better the ten statistics calculated and thus can be expected to increase the robustness at no additional cost.

5.3 Discussion

The prediction errors for the HC23 problem are displayed in Table 1. The three results are very similar, with a slight advantage for the SNN, that could be perfectly due to sampling variability. This is confirmed by the obtained statistical significances, displayed in Table 2. In this table, performance of the SNN is compared to that of the two RBFs. The three hypotheses raised are that SNN performance is not equal to the other two, one by one. Only the t test for averaged classification error turns out to be significant (though by a small margin). The other hypotheses cannot be rejected (also by a small margin). The conclusion is that none of the networks is able to adequately capture the complex relationship between predictors and target class, attending to the relatively high normalized squared errors.

Table 1: Averaged prediction errors for the HC23 problem. *Error* is the percentage of errors; *MSE* is the mean squared error; *NRMSE* is the normalized MSE.

Method	Error	MSE	NRMSE
SNN	32.79	0.147	0.901
RBF2	33.33	0.149	0.907
RBFk	33.22	0.148	0.906

Table 2: Statistical significances for the HC23 problem. Positive results are shown in bold.

Obs.	Test	RBF2	RBFk
Error	t	**2.692**	2.050
	F	1.783	1.193
MSE	t	1.880	2.251
	F	0.847	1.058
NRMSE	t	1.894	2.26
	F	0.855	1.056

The prediction errors for the HC24 problem are displayed in Table 3. This time SNN performance seems much better than that of the two RBFs. This is also suggested by the obtained statistical significances, displayed in Table 4. Rather surprisingly, the t and F tests do not turn out to be significant for averaged

classification error, but they are for the two types of squared errors. This effect can be caused by the rather indirect relation between classification accuracy and squared error. Given that the networks were trained to minimize squared errors, these should be the quantities to be taken as reference to evaluate performance.

A different matter is overfitting analysis. If a network obtains lower squared errors because it concentrates on reducing the error on certain examples (and ignoring others), this would be reflected in a similar or worse classification accuracy. Since this is not the case (lower square errors entail lower prediction errors) the conclusion is that the SNN shows a better fit for the problem. This said, the relatively high normalized squared errors point out that the fit could be much better.

Table 3: Averaged prediction errors for the HC24 problem. *Error* is the percentage of errors; *MSE* is the mean squared error; *NRMSE* is the normalized MSE.

Method	Error	MSE	NRMSE
SNN	16.73	0.128	0.740
RBF2	20.01	0.152	0.808
RBFk	20.06	0.153	0.809

Table 4: Statistical significance for the HC24 problem. Positive results are shown in bold.

Obs.	Test	RBF2	RBFk
Error	*t*	0.505	1.370
	F	1.874	3.053
MSE	*t*	2.286	**2.986**
	F	**15.119**	**12.786**
NRMSE	*t*	2.351	**2.755**
	F	**14.602**	**10.346**

A final point is made on the causes of this differential performance among the nets. The only difference in the two RBFs is found in the way the first-layer weights (the centers) are set up. The virtually non-existent differences among the two methods (for both problems) suggest that the clustering algorithm is playing no role in SNN performance. Therefore, any difference should be attributable to the similarity processing.

6   Conclusions

In a neural network training process, the hidden layer(s) try to find a new, more convenient representation for the problem *given* the data representation chosen,

a crucial factor for a successful learning process. This part of the solution can be seen as a clever pre-processing of the dataset to better capture the underlying similarity relations between the examples. Subsequent (similarity) data processing can be delivered to linear modeling methods (such as those used in this paper). However, nothing prevents the use of a non-linear method (as a support vector machine) that will hopefully solve better the remaining target variability.

We advocate for the use of expert knowledge, whenever available, to choose the best similarity functions for each variable. This part is a two-blade sword, in that the enormous flexibility of the method is in balance to the laborious work of analysing each variable and taking appropriate decisions. Since we are not able to contemplate all data types in advance, we have presented a set of very basic partial similarity functions that can be taken as *default* methods.

The interpretability of the network is greatly enhanced, for two reasons. First, the neurons are centered at known examples acting as prototypes. Second, the output of the k-th neuron is a linear combination of the similarities of the input to the set of prototype neurons.

We reckon that much work remains to be done until the SNN can turn into a viable off-the-shelf modeling method. In this sense, and in the light of the analysed Horse Colic problem, it is worth noticing that the SNN could be further taylored with ease, something much more difficult in the case of the two RBFs.

References

1. H. Osborne, D. Bridge: Models of similarity for case-based reasoning. In: Interdisciplinary Workshop on Similarity and Categorization, pp. 173–179 (1997)
2. R. Baeza-Yates, B. Ribeiro-Neto: Modern information Retrieval, ACM Press, New York (1999)
3. R.C. Veltkamp, M. Hagedoorn: Shape similarity measures, properties and constructions. In VISUAL 2000: Advances in Visual Information Systems, LNCS, vol. 1929, Springer, Heidelberg (2000)
4. Frank, A., Asuncion, A. UCI Machine Learning Repository `http://archive.ics.uci.edu/ml`. Irvine, CA: University of California, School of Information and Computer Science (2010)
5. Kaburlassos, V., Petridis, V. Learning and decision-making in the framework of fuzzy lattices. New learning paradigms in soft computing, Physica-Verlag (2002)
6. Omohundro, S. M. Geometric Learning Algorithms. Technical Report 89-041. Univ. of Berkeley, CA (1989)
7. L. Prechelt. Proben1-A set of Neural Network Benchmark Problems and Benchmarking Rules. Fac. für Informatik. U. Karlsruhe T.R. 21/94 (1994)
8. C. Bishop. *Neural Networks for Pattern Recognition.* Oxford (1995)
9. V. Tresp, S. Ahmad, R. Neumeier. Training Neural Networks with Deficient Data. In NIPS 6, Morgan Kaufmann (1994)
10. J. L. Chandon, S. Pinson. *Analyse Typologique.* Masson (1981)
11. J. Orozco: Similarity and dissimilarity concepts in machine learning. Technical Report LSI-04-9-R, Universitat Politècnica de Catalunya, Barcelona, Spain (2004)
12. D. Lin: An information-theoretic definition of similarity. In: International Conference on Machine Learning, Madison, Wisconsin, USA (1998)

13. Zadeh, L. Fuzzy Sets as a basis for a theory of possibility. Fuzzy Sets and Systems, 1: 3-28 (1978)
14. J.C. Gower. A General Coefficient of Similarity and Some of Its Properties. Biometrika, 27(4), pp. 857–871 (1971)
15. J. Hartigan. Clustering Algorithms. Wiley (1975)
16. M. Orr. Introduction to Radial Basis Function Networks. Technical Report of the Institute for Adaptive and Neural Computation. Available at `http://www.anc.ed.ac.uk/rbf/papers/intro.ps.gz` (1996)
17. R. Xu, D. Wunsch. Clustering. IEEE Press Series on Computational Intelligence. Wiley (2008)
18. T. Dietterich. Approximate statistical tests for comparing supervised classification learning algorithms. Neural Computation 10(7), pp. 1895–1923 (1998)
19. E. Alpaydin. Combined 5×2 cv F Test for Comparing Supervised Classification Learning Algorithms. Neural Computation 11(8), pp. 1885–1892 (1999)