A Theory for Heterogeneous Neuron Models based on Similarity

Lluís A. Belanche Muñoz

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@lsi.upc.es

Abstract

This work presents a 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 serves as a unification cover for many of the existing neural models, including those classically used for the MultiLayer Perceptron (MLP) and most of those used in Radial Basis Function Neural Networks (RBF). These families of models are conceptually unified and their relation is clarified. The possibilities of deriving *new* instances are then explored and several neuron models –representative of their families– are proposed. In addition, the similarity view leads naturally to further extensions of the models to handle heterogeneous information, that is to say, information coming from sources radically different in character, including continuous and discrete (ordinal) numerical quantities, nominal (categorical) quantities, and fuzzy quantities. Missing data are also treated explicitly as such. A neuron of this class is called an *heterogeneous neuron* and any neural structure making use of them is an **Heterogeneous Neural Network** (HNN), regardless of the specific architecture or learning algorithm. In this work the experiments are restricted to feed-forward networks, as the initial focus of study. The learning procedures may include a great variety of techniques, basically divided in derivative-based methods (such as the conjugate gradient) and evolutionary ones (such as genetic algorithms).

Contents

1 Introduction

> "Rien n'est plus semblable à l'identique
> que ce qui est pareil à la même chose
> et réciproquement."
>
> Pierre Dac

The class of adaptive systems known as Artificial Neural Networks (ANN) were motivated by the amazing parallel processing capabilities of biological brains (especially the human brain). The main motivation, at least initially, was to re-create these abilities by constructing artificial models of the biological neuron. The actual artificial neurons –as used in the ANN paradigm– have little in common with their biological counterpart. Rather, they are primarily used as *computational devices*, clearly intended to problem solving: approximation of functions, identification of systems, classification, time-series prediction and others.

The power of biological neural structures stems from the enormous number of highly interconnected simple units. The simplicity comes from the fact that, once the complex electro-chemical processes are abstracted, the resulting computation turns out to be conceptually very simple. Artificial neurons are modeled to take profit of this by proposing simple computing devices that resemble the abstracted original function. However, in the ANN paradigm only very few elements are connected in most practical situations (on the order of hundreds, to say the most) and their connectivity is low. Therefore, it seems reasonable, once the "biological origin" is so departed, to think in compensating these low numbers by increasing the power of single units, while retaining the conceptual simplicity of seeing a neuron (a computational unit) as a pattern recognizer, a device that integrates its incoming sources with its own local information available (its weights) and outputs a single value expressing how much do they match.

One marked shortcoming of the neuron models existent in the literature is the difficulty of adding knowledge (either of the problem to be solved or of the data themselves) to the model. The neuron is used as a function (that is, a mapping) from $\mathbb{R}^n$ to $\mathbb{R}$ (possibly, though rarely, with an internal state) as a black-box. For such a blind processing element, part of the task is to find a structure in the data, to transform it from a *raw form* (possibly in a space of high dimension, only here and there covered by example cases) to a new space (the *hidden space*, or space spanned by the hidden units) in such a way that the problem is simpler when projected to this new space. The same processing is repeated for all the subsequent hidden layers. This is true at least for two of the most widespread neuron models: that used in MultiLayer Perceptrons (MLP) [3] –basically a scalar product between the input and weight vectors plus an offset followed by a squashing function– and that used in Radial Basis Function Neural Networks (RBF) [4] –some distance metric between the input and weight vectors followed by a monotonic basis function (for instance, a Gaussian). In this latter case, only a hidden layer suffices. In any case, the hidden layer's task is to find a new, more convenient representation for the problem *given* the data representation, and eventually performing a reduction in dimensionality.

The way the data is represented is very important. This issue not only includes which information is explicitly fed into the network after a *feature extraction* process (usually in the form of input vectors) and which is not. It also includes (or should include):

1. Input (and sometimes, output) preprocessing, mostly normalization of variables or cases;

2. The way non-standard information is encoded (e.g. codings for discrete ordinal or nominal variables);
3. The way missing values are handled.

Data representation can be a crucial factor for a succeeding learning process and can also have a great impact on generalization ability [17]. Therefore, the task of the hidden layer transformation is twofold:

1. Detection of data redundancy and relative relevance of variables;
2. Approximation of the actual underlying function.

A useful and many times necessary assumption is that of *smoothness* of the obtained solution, required to filter out small input variations (read noise) and beneficial since it biases the search space of functions, thereby reducing it. This smoothness requirement can be realized in practice in various ways [35].

The difficulty of this general task is exacerbated because, in the ANN paradigm, the data (and with them, the problem itself) are adapted to the neuron model (or, worse, to the learning algorithm), and not otherwise. Instead, we believe *it is the model* that has to conform to the data, by making use of domain or *prior* knowledge (traditionally called in inductive learning *background knowledge*), roughly defined as the information we possess about the problem itself *and* about the nature of its data, apart from the raw examples themselves. If this information (assumed relevant) is supplied to a neural network or included in its design, it will not have to be discovered.

1.1 Basic neural models

In ANN research, there are basically three "free parameters":

1. The neuron model;
2. The architecture, i.e., how these neurons are to be arranged;
3. The learning algorithm, i.e., how the weights are to be determined.

Undoubtedly the most active of these three subjects is the last. Once the first two are known and fixed, the problem of choosing the most suitable way of proposing a solution in weight space (fast and reliably) becomes the most important area of research. Studying new variations in architecture, on the other hand, is of interest mostly for recurrent systems.

There are two types of basic neuron models in the literature used in practice. Both decompose the overall computation of the unit (the *transfer* function) in two stages, as is classically done since the earlier model proposal of McCulloch & Pitts. The computation of a unit i is expressed as a function h_i of its input:

$$h_i(\vec{x}) = g(f(\vec{x}, \vec{w}^i)) \quad (1)$$

where $\vec{w}^i$ is the weight vector of neuron i, $\vec{x}, \vec{w}^i \in \mathbb{R}^n$, $f : \mathbb{R}^n, \mathbb{R}^n \to \mathbb{R}$ is called the *net input* or *aggregation* function, and $g : \mathbb{R} \to \mathbb{R}$ is called the *activation* function. The choice $f(\vec{x}, \vec{w}^i) = \vec{x} \cdot \vec{w}^i$ leads to the most widely used neuron model. When neurons of this type are arranged in a feedforward architecture, the obtained neural network is the MLP aforementioned. The g can either be linear or non-linear. In the latter case, a squashing function is the usual choice. The choice $f(\vec{x}, \vec{w}^i) = ||\vec{x} - \vec{w}^i||_{L_2}$ (or other distance measure) plus an activation g with localized response leads to the family of RBFs. Localized means that they give a significant response only in a neighbourhood of their centre $\vec{w}^i$.

Other types of neurons are obtained by generalizing the previous choices for f to take into account correlations between input variables (components of the input vector). In the first case, the inner product (containing no cross-product terms) can be generalized to a real quadratic form (an homogeneous polynomial of second degree with real coefficients) or even further to higher degrees, leading to so called higher-order units (or $\Sigma\Pi$ units). A higher-order unit of degree k includes all possible cross-products of at most k input variables, each with its own weight. Conversely, basic Euclidean distances can be generalized to completely weighted distance measures, where all the (quadratic) cross-products are included. However, these full expressions are not commonly used because of the high numbers of parameters they involve. Other models, like product units, replace the weighted arithmetic mean of the scalar product by its geometric counterpart, $f(\vec{x}, \vec{w}^i) = \prod_j x_j^{w_j^i}$. Although more powerful, they generate a much more difficult search space [11].

These two basic neuron models have traditionally been regarded as completely separated, both from a mathematical and a conceptual point of view. To a certain degree, this is true: the local vs. global approximation approaches to a function that they carry out make them apparently quite opposite methods. Though a RBF can be trained like a MLP, with a derivative-based technique, the learning algorithms usually applied are quite different in nature [39]. Mathematically, under certain conditions, they can be shown to be related [9]. These conditions (basically, that *both* vectors are normalized to unit norm) are rarely met. We will show that, under the unifying cover of similarity, the two models can be seen as different instances of the same general model, without the need of such assumptions.

It is notable how the majority of research attention in ANNs has been devoted to the proposal of new learning algorithms or the improvement or somehow enhancement of existing ones (much more usual). Some of the techniques have been rediscovered several times or have been recast from methods known in statistics or other disciplines. Important as it is, this aspect hides an obvious fact: there are at least *three* components influencing the task of finding a solution to a (supervised) ANN learning problem:

1. the function that is actually being presented to the ANN via input/output examples of it;
2. the repertoire of functions that the network has, in order to approximate the function in 1.
3. the algorithm the network uses to select one of the functions in 2.

By focusing only in 3., we are severely limiting the families of functions in 2. In theory, very many functions can in principle work well (because they have a universal approximation capability, for example) but *in practice* the results can be very different for different selections of 2. The cornerstone for this selection is precisely the neuron model. Final performance is also strongly influenced by 1.; how exactly is the target function presented to the network will greatly condition the results.

1.2 Scalar product as a pattern recognizer

Since the early days of pattern recognition, artificial neuron models were designed as computers of a "recognizing function" (giving rise to the name of the field). As such, one of the first activation function proposed was the Heaviside:

$$H_\theta(z) = \begin{cases} 1 & \text{if } z \geq \theta \\ 0 & \text{if } z < \theta \end{cases} \tag{2}$$

This function was used in the McCulloch & Pitts model [7], in many of the first non-linear associative memories (such as the *lernmatrix* [6]) and, fundamentally, in the Perceptron [8]. The binary output reflected the original closeness to the biological counterpart. When an input vector (or pattern) is supplied to the neuron, its response is either 1 or 0, standing for "I *do* recognize it" or "I *do not* recognize it". The recognition part is dealt with by the aggregation function, the ubiquitous scalar (or inner) product. It is worth digressing a little about the reasons why this way of aggregating vectors has been so widely used.

In the first place, there is the once appealing analogy to the biological counterpart (the original inspiration). The slow, analog integration of incoming connections of a neuron, each one acting as (strong or weak) excitation or inhibition channels was modeled (see previous remarks on the abstraction process) by an instantaneous and discrete "integration" operator: the weighted summation. This leads to one of the definitions of scalar product.

$$\vec{x} \cdot \vec{y} = \sum_i x_i y_i \tag{3}$$

In the second place, the definition of the angle between two vectors can be related to their scalar product:

Definition 1.1 *The angle θ between two non-null vectors, $\vec{x}, \vec{y}$ is defined as*

$$cos\theta = \frac{\vec{x} \cdot \vec{y}}{||\vec{x}||||\vec{y}||} \in [0, \pi] \tag{4}$$

where $||\cdot||$ is usually the L_2 norm.

The geometric interpretation of this definition is depicted in Fig. 1. Let us suppose that $\vec{x} = (x_1, x_2)$ and $\vec{y} = (y_1, y_2)$ and, without loss of generality, that $||\vec{x}|| < ||\vec{y}||$.

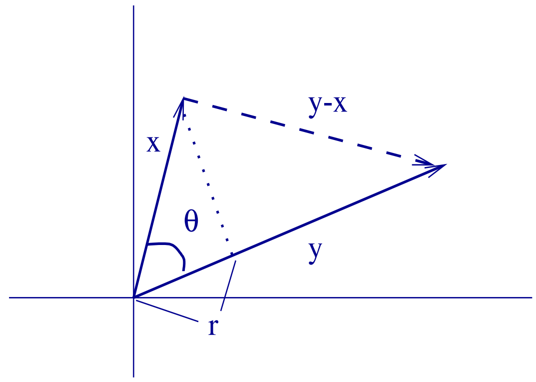


Figure 1: Two-dimensional geometric interpretation of scalar product.

From trigonometry we have $\frac{r}{||\vec{x}||} = cos\theta$ and therefore

$$r = ||\vec{x}||cos\theta = \frac{\vec{x} \cdot \vec{y}}{||\vec{y}||} \tag{5}$$

is the fraction of $||\vec{x}||$ that is pointing in the direction (with sign) of $\vec{y}$. Notice that if $\vec{y}$ is parallel to the x-axis, then $r = x_1$. In other words, $\vec{x} \cdot \vec{y}$ can be seen as $||\vec{y}||$ times the projection of $\vec{x}$ on $\vec{y}$, or viceversa ($||\vec{x}||$ times the projection of $\vec{y}$ on $\vec{x}$).

This *bounded* fraction of $||\vec{x}||$ gives an idea of how close or separate (in θ) are $\vec{x}$ and $\vec{y}$. It can also be regarded as how close are *the points* represented by $\vec{x}, \vec{y}$, as measured by $||\vec{y} - \vec{x}||$; the smaller the norm, the greater is r, and with it $\vec{x} \cdot \vec{y}$ (for constant $\vec{y}$), showing the relation between a model working on vector scalar product and one working on a norm of the vectorial difference. The value r is bounded by

$$-||\vec{x}|| \leq r \leq ||\vec{x}|| \tag{6}$$

The lower bound corresponds to the two vectors being colinear and going in opposite directions. The upper bound signals colinearity and the same direction. The magnitude is hence only a scalar factor that amplifies the value given by the angle. Therefore $\vec{x} \cdot \vec{y}$ is greater:

1. The closer (in θ) are $\vec{x}$, $\vec{y}$;
2. The longer (in $||\cdot||$) are $\vec{x}$, $\vec{y}$.

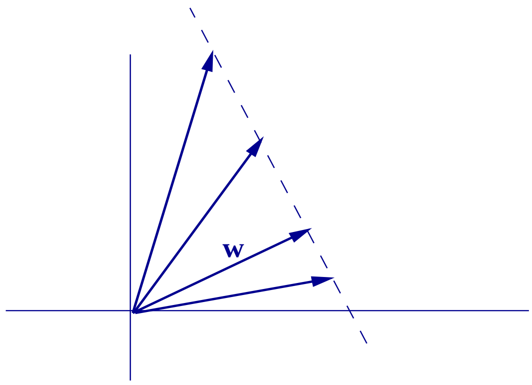


Figure 2: The scalar product between $\vec{w}$ and the other three vectors is constant.

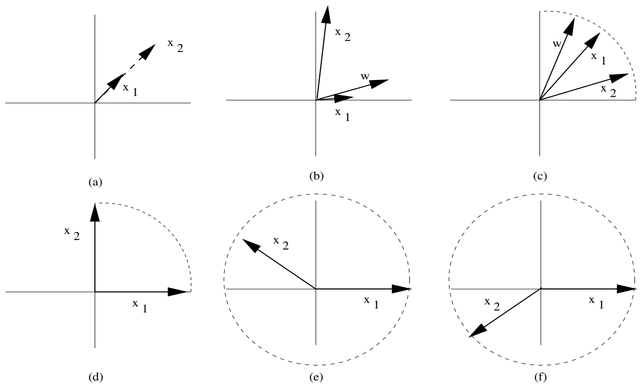


Figure 3: (a) Co-linear vectors ($\theta = 0$), $|\vec{x}\cdot\vec{y}|$ is maximum. (b) $\vec{w}\cdot\vec{x_1} < \vec{w}\cdot\vec{x_2}$ although $\vec{x_1}$ is more similar to $\vec{w}$ than $\vec{x_2}$. (c) Normalized vectors all lie on a circumference of radius one; $\vec{w}\cdot\vec{x_1} > \vec{w}\cdot\vec{x_2} > 0$ ($\theta \in [0, \frac{\pi}{2}]$). (d) Scalar product for orthogonal vectors is 0 ($\theta = \frac{\pi}{2}$). (e),(f) Scalar product is negative for $\theta \in [\frac{\pi}{2}, \frac{3\pi}{2}]$.

leading to points of constant value lying in parallel hyperplanes (Fig. 2). Note also that, being the scalar product commutative, the initial assumption about their relative magnitude is irrelevant. Some of the possible situations are depicted in Fig. 3. In general, we have $|\vec{x}\cdot\vec{y}| \leq ||\vec{x}||\,||\vec{y}||$ (the Cauchy-Buniakowski inequality). If magnitude effects are to be eliminated, normalization is necessary:

$$\vec{x}\,' = \frac{\vec{x}}{||\vec{x}||}; \qquad \vec{y}\,' = \frac{\vec{y}}{||\vec{y}||} \tag{7}$$

so that $\vec{x}\,'\cdot\vec{y}\,' = cos\theta \in [-1,+1]$ gives a measure of how *similar* $\vec{x}\,'$, $\vec{y}\,'$ are, traditionally used in statistics (Q-mode factor analysis) as a measure that preserves proportions. This makes scalar product a simple but effective *recognition* function, linear, continuous and differentiable. A neuron model $F_i(\vec{x})$ with weight vector $\vec{w}^i$ endowed with this measure will thus compute *the projection of* $\vec{x}$ *on* $\vec{w}^i$, multiplied by $||\vec{w}^i||$:

$$F(\vec{x}) = \vec{w}^i\cdot\vec{x} = ||\vec{w}^i||\,\underbrace{||\vec{x}||cos\theta}_{\text{projection}} \tag{8}$$

The problem of this measure is that, although it is *locally* bounded for a specific pair of vectors (the above inequality), the vectors themselves are not, and hence its usage as an explicit recognition device is severely limited. Variable standardization can be of help. A common input preprocessing scheme is to bring variables to zero mean and unit standard deviation. Chebyshev's inequality (see Appendix C) tells us that, in these conditions, by choosing a $k \in \mathbb{N}^+$,

$$\Pr(|x_i| > k) \leq \frac{1}{k^2}$$

and therefore we could propose a "bound" for $\vec{x}$ as $\vec{x}_{max} \approx (k, k, \ldots, k)$ so that $||\vec{x}_{max}|| = k\sqrt{n}$. A value of $k = 7$ should suffice (probability is upper-bounded by ≈ 0.02). However, the lack of an explicit control of the neuron (Is this really a good bound? How are the weights bounded?) and the dependence on the k parameter[1] still suggest that a modification of the model (not of the data!) can lead to improved general (similarity) computing devices.

[1]Note that there is a compromise between high values of k (to avoid overflowing the bound) and low values (to prevent setting an unrealistically high bound, which would make the neuron hardly sensitive).

In conclusion, although not a proper similarity measure (we will see later that it does not fulfill one of the basic requirements to be called such) the conceptually nice dual view of scalar product as a weighted summation or as an "absolute" similarity (along with its clean analytical behaviour) is the rationale behind its use from the outset until today. Besides, both interpretations are easily linked by noting that (3) is in fact the Cartesian equation of the hyperplane written in a compact form:

$$\pi : x_1 w_1 + x_2 w_2 + \ldots + x_n w_n = 0 \qquad (9)$$

with direction vector $\vec{w}$ (orthogonal to the plane) and constrained, in absence of an independent term, to pass through the origin. The situations in Fig. 3 can now be looked at again by taking one of the vectors as a reference and imagining a line perpendicular to it passing through the origin: it will be the hyperplane of which this vector is a director vector. Any other vector lying in the same side of the hyperplane as the reference will yield a positive scalar product, greater with increasing distance *to the hyperplane.* The situation is analogous for vectors lying in the opposite side and (increasingly) negative scalar products. Of course, any vector lying *on* the hyperplane will be orthogonal to the reference one, and will yield a null scalar product. This is the basic principle of the way Perceptrons work.

The basic Heaviside function was later generalized by softening it in a controlled way, leading to the family of squashing activation functions, of which the sigmoidal functions are well-known members. The main property of these functions (the "squashing property") is their asymptotic behaviour in the x-axis, which implies they are bounded functions. As an example, consider the logistic parameterized function:

$$g_{\beta,\theta}(z) = \frac{1}{1+e^{-\beta(z-\theta)}}, \qquad \beta > 0 \qquad (10)$$

It can be shown that $\lim_{\beta\to\infty} g_{\beta,\theta}(z) = H_\theta(z)$. The net effect of a squashing activation function (not considering its effects on a derivative-based learning algorithm) can be regarded as producing a softened up recognition, a normalized (because of the boundedness characteristic) *degree of recognition.* Notice that this function, being a strictly increasing monotonic one, does not alter the discussion so far.

1.3 Similarity as a basis for neural computation

In theory, ANN design should follow the principle:

Principle 1.1 *Similar patterns should yield similar outputs.*

However, what "similar patterns" means is problem-dependent, and only in counted occasions will coincide with the fixed interpretation of similarity that the network is going to perform. There is a more basic principle [10] to be satisfied:

Principle 1.2 *Similar patterns should have similar representations.*

The fulfillment of this principle easies the fulfillment of the former, although it does not directly imply it, because neuron models will make use of their own fixed "similarity measure" on the representations. Consider the example depicted in Table 1, where patterns include natural numbers (a situation much more common than not). Principle 1.2 says that their representation (the actual *input space*) has to be respectful with their nature. If these pattern components are represented as, say, binary-coded bitstrings (which only respect the characteristic of being discrete) then Principle 1.2 is being violated and, with it, Principle 1.1. This is because the similarity behind natural numbers (possibly based on some metric in $\mathbb{N}$) does not correspond at all to that used in the same binary-coded numbers actually fed to the neurons. For example, two correlative numbers in $\mathbb{N}$ as 31 and 32 (thus very similar) are far in Hamming distance. This change of representation is not part of the original task and has to be additionally discovered by the network.

pattern space	input space
$\ldots, 29, 30, 31, \ldots$	$\ldots, 011101, 011110, 011111, \ldots$
(Metric in $\mathbb{N}$)	(Hamming metric)

Table 1: Representation of natural numbers as a binary code.

This is exacerbated in the case of scalar-product driven neurons, for which magnitude will also be important, and with magnitude into account the computation is further distorted. As an example, consider two vectors in $\vec{x_1}, \vec{x_2} \in [0,1]^6$, depicted in Table (2).

	Case 1	Case 2
$\vec{x_1}$	111001	001001
$\vec{x_2}$	111111	010001
	$d_H(\vec{x_1}, \vec{x_2}) = 2$	$d_H(\vec{x_1}, \vec{x_2}) = 2$
	$d_E(\vec{x_1}, \vec{x_2}) = \sqrt{2}$	$d_E(\vec{x_1}, \vec{x_2}) = \sqrt{2}$
	$\vec{x_1} \cdot \vec{x_2} = 4$	$\vec{x_1} \cdot \vec{x_2} = 1$

Table 2: Influence of magnitude *in the representation* (d_E, d_H stand for the Euclidean and Hamming distances, respectively).

In both cases (1 and 2), scalar product yields quite different values, despite the fact that the vectors $\vec{x_1}, \vec{x_2}$ are equally near in Hamming distance. As for RBF units note that, denoting by b_2 the "base 10 to base 2 operator", it can be checked that

$$d_E(b_2(i), b_2(i')) = \sqrt{d_H(b_2(i), b_2(i'))} \; i, i' \in \mathbb{N} \quad (11)$$

This means that a neuron model based on Euclidean distance (instead of Hamming) on the bit representations will neither capture the original metric in $\mathbb{N}$ (assumed also Euclidean), because the change of representation relates all computations to the Hamming distance, thus violating Principle 1.2. Additionally, artificially augmenting the input dimension can only harm local approximation schemes like that performed by a RBF network.

All this means that, even if we devise a representation that is reasonably respectful to Principle 1.2, there is no guarantee that Principle 1.1 will be respected, because of the peculiarities induced by the neuron model. Additionally, the possibility of using a Grey code, for instance, would only fulfill Principle 1.2 locally, rendering it almost useless[2]. Even if we could make Euclidean distance behave proportionally in the representation, this would still be based on the assumption that the actual measure that should be used in general in $\mathbb{N}$ or one of its subsets is the Euclidean, certainly far from reality in some situations[3].

Therefore, the two requirements to ensure that both Principles are to be satisfied can be resumed in the following:

[2]In a Grey code, two consecutive naturals would differ in only one value. This is a useful coding wherever a small change in representation involves a small change in the represented entity (as, for example, in Genetic Algorithms). However, medium or big differences do not elicit correspondingly high Hamming distances.

[3]Think of a quantity representing a date, ranging in 1..31; the circular behaviour should be realized somehow by altering the basic distance. For a (natural) variable standing for the number of children, the distance between 7 and 9 is not the same "psychological" distance than that between 1 and 3 (which is triple). In both situations, domain knowledge is being used.

1. To identify and understand what does *similar* mean for the task at hand and propose a neuron model that best captures this functional similarity, at least for the first layer of neurons (the layer in contact with the input space). This deals with the *pattern similarity*;

2. To choose a representation –if still needed– that fulfills Principle 1.2. This has to do with the *pattern heterogeneity*;

1.4 Similarity in ANN

The appealingly simple and general "physical similarity" computation performed by a neuron working on scalar product has already been put in doubt as being so general or conceptually right in all situations [1]. The term physical similarity is coined to reflect the fact that this measure interprets the collection of input variables, whatever they are meant to represent, as the coordinates of a point in n-dimensional Euclidean space. The closer these points are (the shorter the length of the line joining them) the greater this measure will be. There are two basic flaws with this "raw" similarity computation.

The first one is the dependence on magnitude. Since the input variables are treated as vectors, their length is important. As we saw in the previous section, scalar product gives a value that is greater when *either* the angle is smaller or the product of magnitudes is greater. This implies that the measure is potentially unbounded, an undesirable feature, because if a function is to express a recognition degree, this degree has to be relative to some maximum. Of course, it can be argued that, at least in practical applications, the target function will have a compact support, and thus the "maximum scalar product" will be bounded. This remark is not useful since, on the one hand, we may not know the actual input distribution (it might, as is often the case, be covering a submanifold of the input space) and thus the theoretical maximum is an unreachable one and the actual maximum is unknown. On the other hand, what is the maximum length of a weight vector? Although in practice small weights are beneficial, there is no clear way of setting an a priori limit. Moreover, some recent experiments have shown that the range of possible weights strongly conditions the results of a learning process [2]. In conclusion, raw scalar product is an uninformative measure of similarity. In the classical neuron model (that of the MLP), the squashing activation function does part of the job: its boundedness permits to see the whole neuron computation as a bounded similarity function. However, it does not solve the problem to satisfaction: although the activation function is bounded, *its argument* is not, so that the function may be too easily brought close to its extreme values, loosing the sensibility the neuron should have. What is more, the unit is still computing the "physical similarity" which, quoting [1]:

> "... may not be the best measure of the *functional* or *psychological* similarity we would like to employ at the output." (my italics)

This is the second basic flaw, easily illustrated in the following examples. Consider the three triangles of Fig. 4. To what other figure is more similar triangle number 1? To number 2 (because, although is bigger, it keeps the same proportions), or to number 3 (because is of the same size, though with different proportions)? One could argue that number 2 is *the same* triangle as number 1, though slightly scaled up and, thus, they should be treated as similar or, better, *equal*. This first example tells us that a similarity judgement depends on what input patterns really mean to us and how should they, as a consequence, be treated.

This is an old problem in cognitive psychology and other disciplines.

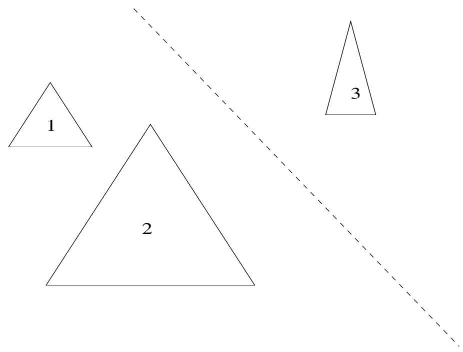


Figure 4: What triangle is more similar to triangle number 1?

As another example, consider a typical character recognition task, in which characters are encoded as binary (or continuous, for what matters) pixels in a grid, which is presented to a network as a vector in a space of very high dimension (the number of pixels in the grid) or, better, as a vector of features in the form of pixel combinations. The classification of the letters in the set $\{a, \ldots, z, A, \ldots, Z\}$ is clearly problem dependent. The precise classification the user has in mind (e.g., lower case vs. upper case or a/A vs. b/B vs. ... z/Z, or even vowels vs. consonants, etc) will surely not be the one the network "sees". In other words, patterns $\vec{x}$ that are similar to a neuron (because the inner product of $\vec{x}$ with its weight vector is high) need not (surely will not) be similar in the eyes of the user (read problem). In fact, the learning task consist precisely on this: to show the network what is the actual similarity relation. In a practical data clustering scenario reported in [13], four skulls, one of an adult *Homo Sapiens*, one of an infant *Homo Sapiens*, one of an adult gorilla, and one of an infant gorilla had to be told apart. One reasonable outcome is to have both *Homo Sapiens* skulls grouped together (that is, to ignore the differences in skull size of adults and infants). The other is to have the two adult skulls separated from the two infant ones. These are different tasks; In both cases, though, the network is using the *same* basic transformation. In these examples, the main problem is the dichotomy "what input patterns really mean to us"/"what do they mean to a neuron". Much of the learning process is devoted to narrow this gap.

What is actually needed is a transformation to another representation form in which patterns requiring similar (for the problem) responses are indeed similar to one another (for the network). This is classically performed, in theory, by the hidden layer(s) in multilayer networks (Fig. 5). Quoting again [1]:

> "... if we use a sequence of such transformations [the hidden layers], each involving certain non-linearities ... we can entirely rearrange the similarity relations among the original input vectors."

This view is conceptually right and task independent. In practice, it may be that for complex transformations several layers are needed, or very many neurons per layer if we restrict the maximum number of hidden layers to, say, one or two. In all, this is giving support to a deeper and more precise formulation of neuron models as similarity computing devices. If more information is implanted into the network design in the form of supplying a neuron model –at least for the input layer– which captures better the required similarity in the input space (which is also the space where the input-layer weights belong), the task of finding the required transformations may quite possibly be simpler to learn.

To sum up, a reasonable solution –and the one followed in this work– is to have, on the one hand, a normalized neuron computation, *instead* of having to normalize the inputs and, on the other hand, the neuron computation will be explicitly defined in the form of a similarity measure.

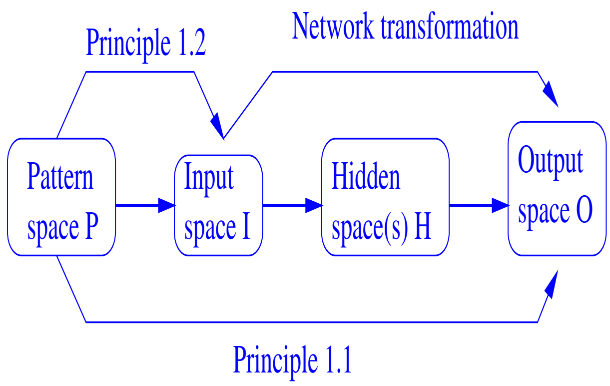


Figure 5: A Neural Network as a similarity transformer: The similarities in each space should be such that $s_I(i,i')$ captures the original likeness s_P in pattern space and $s_I(i,i') \Rightarrow s_H(h,h') \Rightarrow s_O(o,o')$ (I:input, H:hidden, O:output).

There are other, additional advantages in defining a similarity function in a explicit form by taking into account the data peculiarities:

- Practical: *understandability* of the function computed by the network (because of the domain knowledge used in its definition) and *readability* of the results (what do the weights mean).
- Theoretical: taking profit of the properties of similarity measures (e.g. boundedness, monotonicity, continuity or absence of input preprocessing).

1.5 Aggregation of local models and data heterogeneity

There is another issue in the local vs. global character of the computation involved in an ANN: the one that takes place in a neuron. Once we depart from the biological model and intend to use the artificial neuron as a computing device, there is no need to treat all the inputs in the same way, as is done in the practical totality of existing models. Scalar product and Euclidean distance are examples of a global aggregation measure, because they can be defined as a function of the weight and input vectors *as a whole*. The former can also be conveniently defined as a linear combination of components. In this case, all of the partial recognitions are performed using the same function (the correlation) and combined with a simple summation. In all, they are particular cases of performing a *combination* of *partial* recognitions. There is no local information or knowledge used to devise a particular recognition measure, *specifically tuned* to its input, nor are these partial measures combined in a special way.

However, real-world data come from many different sources (continuous or discrete processes, ...) and have their own peculiarities (imprecision, uncertainty, ...), and thus may require different treatments. Consider, for example, the common expression of Euclidean distance in an n-dimensional vectorial space:

$$d^2(\vec{x},\vec{y}) = \sum_{i=1}^{n}(x_i - y_i)^2 = \sum_{i=1}^{n} d^2(x_i, y_i) \quad (12)$$

where $d(x,y) = |x-y|$ for $x,y \in \mathbb{R}$. This is obtained by reducing (12) to the case $n = 1$. In general, a distance metric could be conceived by considering the following issues:

- Replacing $d(x,y)$ by any other distance in $\mathbb{R}$ (for example, one-dimensional Minkowskian of the type $|x-y|^q$).
- There is no need to use the *same* metric d in all real dimensions. In other words, it is possible to have specific d_i for different real components.
- In the same vein, for general input components that are *not* subsets of $\mathbb{R}$, specific measures should be devised.
- The way these partial (local) distances are combined need not be the one in (12).

Further, it may possibly be of interest for the local distances to be bounded or, better, normalized, because this way the possible combination functions could be more general (by making them independent of data particularities), and eventually easier to formulate, so that a handful of them could serve in practically all situations. This is clearly another way of adding domain knowledge in the design of the neuron model to be used and, by extension, into the function the ANN is computing.

1.6 Concluding remarks

In this introduction we have tried to provide a kind of overview and rationale for the design of neuron models cast under the common framework of similarity measures. Unification of the plethora of models existent in the literature is not pretended (e.g. non-localized RBFs, or scalar product-based neurons using non-monotonic activation functions, as the sinus are not included). Instead, the proposed framework is general enough to offer controlled means of designing models having a certain number of desirable properties.

The definition of flexible neuron models in the form of a general framework in which:

1. the computation performed by the model is defined explicitly as a similarity measure;
2. different data sources are directly and specifically treated,

permits a natural extension of existing models, and opens a way for devising new ones under the general cover of similarity computing devices. It also gives a *semantics* to the neuron computation (e.g. a psychological similarity) in contrast to the "syntactic" physical similarity. It is our hypothesis that these models will add to the *practical* computing power by enhancing the expressiveness, adapting the model to the data. Our attention will be focused primarily on defining basic concepts within this conceptual framework and explore some elementary properties or advantages.

The remainder of the paper is developed as follows. First, the basic neuron models and the way they are gathered in feed-forward architectures are introduced and formalized (§2). Next, the Section on similarity measures (§3) contains the definitions and propositions which constitute the core of the paper. Part of the more basic material can be found in a textbook on Typological Analysis (our basic reference is [13]). The description of heterogeneous variables is the topic of the next Section (§4), ending in the definition of a heterogeneous space, as well as proposals for similarity measures for it. The introduction of a s-neuron (a neuron computing a similarity and working in one such space) is dealt with in Section (§5). A worked example is therein developed. Several classes of distance to similarity transforming functions are then displayed, leading to the concepts of model and network sensitivity. Finally, a numerical experiment (§6) is preliminary carried out to illustrate the different ideas introduced and the general validity of the approach. The paper ends with some reflexions and concluding remarks.

The Appendices mainly contain proofs for the Propositions and claims of the work. It also includes additional material, relevant for the discussion but in some sense secondary.

2 Neuron Models

We consider only the basic models (not including higher-order terms). RBFs and MLPs both provide parameterized families of functions suitable to function approximation on multidimensional spaces. These two basic neuron models are sometimes described in a somewhat misleading way. A sigmoidal neuron is often explained as putting up an hyperplane that divides its input space in two halves. In other words, the points of equal neuron activation (with fixed weights) are hyperplanes. This behaviour is not caused by the sigmoidal, but by the scalar product. In the same vein, the isoactivation contours for an RBF unit (in case it is using an unweigthed Euclidean norm) are hyperspheres. This radially symmetric and centered response, again, is not caused by the activation function (Gaussian, exponential or otherwise) but by the norm. In both cases, the activation function is acting as a non-linear and monotonic distorsion of its argument – the net input– as computed by the aggregation function.

Definition 2.1 *Given a real function* $\sigma : \mathbb{R}^n \to (\sigma_{min}, \sigma_{max})$*, define* I^α_σ *as the set of isoactivation points* $I^\alpha_\sigma = \{\vec{x} \in \mathbb{R}^n / \sigma(\vec{x}) = \alpha\}$.

Definition 2.2 (p-neuron) *A neuron model of the form:*

$$F_i(\vec{x}) = \{g(\vec{w}^i \cdot \vec{x} + \theta), \vec{w}^i \in \mathbb{R}^n, \theta \in \mathbb{R}\} \quad (13)$$

with g *a bounded, non-linear, strictly monotonic function for which* $\lim_{z\to\infty} g(z) = g_{max} \in \mathbb{R}$ *and* $\lim_{z\to-\infty} g(z) = g_{min} \in \mathbb{R}$ *will be denoted P-neuron (from Perceptron). For these neurons, the sets* $I^\alpha_{F_i}$ *are (n-1)-dimensional hyperplanes for constant values of* α*, parallel with one another for different* α*. This response comes from the use of the scalar product as the aggregation function.*

In practice, the g are usually the well-behaved sigmoidals (logistic, hyperbolic tangent, etc), though other activation functions are sometimes found in the literature (e.g., sinusoidals). These are not included in the above Definition.

Definition 2.3 (r-neuron) *A neuron model of the form:*

$$F_i(\vec{x}) = \{g(||\vec{w}^i - \vec{x}||_{L_q}), \vec{w}^i \in \mathbb{R}^n, q \geq 1 \in \mathbb{R}\} \quad (14)$$

with g *a symmetric function such that* $g(|z|)$ *is monotonic, and with a maximum* g_{max} *at* $F_i(\vec{w}^i)$ *and a (possibly asymptotically reached)* $g_{min} = 0$ *will be denoted R-neuron (from Radial). For these neurons, the sets* $I^\alpha_{F_i}$ *are (n-1)-dimensional hypersurfaces (centered at* $\vec{w}^i$*) for constant values of* α *(e.g., hypercubes for* $q = 1$*, hyperspheres for* $q = 2$*) concentric with one another for different* α*. The radial response comes from the use of the norm as the aggregation function.*

The norm used can be any Minkowskian norm of the form:

$$||\vec{z}||_{L_q} = \left(\sum_{i=1}^{n} |z_i|^q\right)^{\frac{1}{q}} \quad (15)$$

In practice, typical choices are $q = 2$ and g a Gaussian function.

Definition 2.4 *A single-output feed-forward neural network*[4] *denoted 1-FFNN[c] and consisting of c hidden layers is a function* $\mathcal{F} : \mathbb{R}^n \to \mathbb{R}$ *made up of pieces of the form* $\vec{y}^{(l)} = (F^l_1(\vec{y}^{(l-1)}), \ldots, F^l_{h_l}(\vec{y}^{(l-1)}))$*, representing the output of layer* l*, for* $1 \leq l \leq c$*. The* F^l *denote the neuron model of layer* l *and* $h_l \in \mathbb{N}^+$ *their number, and each neuron* F^l_i *has its own parameters as defined in (13) and (2.3).*

The first output is defined as $\vec{y}^{(0)} = \vec{x}$*. For the last (output) layer,* $h_{c+1} = 1$ *and the single* F^{c+1} *is a P-neuron. The final outcome for* $\mathcal{F}(\vec{x})$ *is the value of* $\vec{y}^{(c+1)}$ *(a vector of a single component).*

[4] For simplicity, we are not considering connections between non-adjacent layers. The input layer is *not* counted as a layer.

Definition 2.5 (MLPNN)
A MultiLayer Perceptron Neural Network is a 1-FFNN[c] for which $c \geq 1$ *and all the* F^l *are P-neurons,* $1 \leq l \leq c$.

Definition 2.6 (RBFNN) *A Radial Basis Function Neural Network is a 1-FFNN[c] for which* $c = 1$ *and all the* F^c *are R-neurons.*

We will concentrate in this work on one-hidden-layer networks. For convenience, we provide with the following proposition, as a particular case of the previous definitions:

Proposition 2.1 *A one-hidden-layer neural network 1-FFNN[1] computes a function* $\mathcal{F} : \mathbb{R}^n \to \mathbb{R}$ *of the form:*

$$\mathcal{F}[h_1; F](\vec{x}) = \sigma(\sum_{i=1}^{h_1} c_i F_i(\vec{x}) + \theta) \qquad (16)$$

where $\theta \in \mathbb{R}$ *is a possibly null offset term,* $c_i \in \mathbb{R}$ *and* σ *will be set appropriately, including the choice* $\sigma(z) = z$.

Proof: The expression is obtained by setting $c = 1$ in Definition (2.4), and making use of Definition (13).

3 Similarity Measures

3.1 Distance measures

Definition 3.1 (Distance) *Given a set* X*, any function* d*:*

$$d : X \times X \to \mathbb{R}^+ \cup \{0\}$$

such that $\forall x, y, z \in X$*:*

i) $d(x, y) = 0 \Leftrightarrow x = y$

ii) $d(x, y) = d(y, x)$

iii) $d(x, y) \leq d(x, z) + d(z, y)$

is a distance in X.

A set X endowed with a distance d will be denoted (X, d) and referred to as a *metric space*. The following is a collection of useful propositions and properties about metric spaces and distance functions. The proofs are given in Appendix B.

Proposition 3.1 *Let* (X, d) *be a metric space and* $S \subset X$*; then* (S, d) *is a metric space.*

Proposition 3.2 $(\mathbb{R}, d)$ *is a metric space for* $d(x, y) = |x - y|$*. This distance function will be referred to as the standard metric in* $\mathbb{R}$ *[12].*

Proposition 3.3 $(\mathbb{Q}, d)$*,* $(\mathbb{Z}, d)$ *and* $(\mathbb{N}, d)$ *are metric spaces for the standard metric in* $\mathbb{R}$.

We will denote the set of distance measures of a set X as $D(X)$. The standard metric in $\mathbb{R}$ can be extended in various ways to $\mathbb{R}^n, n \in \mathbb{N}^+$. We study such ways as particular cases of *aggregation* operators.

Definition 3.2 (Aggregation operator)
Consider a collection of n *non-negative real quantities. For convenience, we group these quantities as a vector* $\vec{z} = \{z_1, z_2, \ldots, z_n\}$*. An aggregation operator* Θ *is a function:*

$$\Theta : (\mathbb{R}^+ \cup \{0\})^n \to \mathbb{R}^+ \cup \{0\}$$

fulfilling:

i) Minimality. $\Theta(\vec{z}) = 0 \Leftrightarrow \forall i \; z_i = 0$.

ii) Symmetry. $\Theta(\vec{z}) = \Theta(\vec{z}\,')$ *for any* $\vec{z}\,'$ *permutation of* $\vec{z}$.

iii) Strict monotonicity. $\Theta(\vec{z}) > \Theta(\vec{z}\,')$ *whenever* $\exists i \; z_i > z'_i \wedge \forall j : j \neq i : z_j = z'_j$.

Remark 3.1 *These conditions imply that* $\Theta(\vec{z}) > 0$ *whenever there exists a* $z_i > 0$.

The use of the norm defined in (15) as an aggregation operator leads to general families of distances.

Definition 3.3 (Minkowskian aggregation) *Define the "Minkowskian aggregation" operator* $\Theta_q^M(\cdot), \forall q \geq 1 \in \mathbb{R}$, *for an* n*-dimensional vector* $\vec{d}$, *as*

$$\Theta_q^M(\vec{d};\vec{v}) = \left\{ \frac{1}{n'} \sum_{i=1}^{n} \left(\frac{|d_i|}{v_i} \right)^q \right\}^{\frac{1}{q}} \tag{17}$$

where $\vec{v}$ *is a weight vector and* n' *an averaging factor, either* $n' = n$ *or* $n' = 1$. *The choice* $n' = n$ *eliminates the dependence on the number of components. It is required that all the* v_i *be positive, in order to fulfill condition (iii) in Definition (3.2). Certain choices of* v_i *can make the terms* $\frac{|d_i|}{v_i}$ *to be normalized so as to lie in* $[0, d_{max}]$. *In these cases, the choice* $n' = n$ *is mandatory so that this characteristic is kept.*

Proposition 3.4 *The aggregation function defined in (3.3) is a norm in* $\mathbb{R}^n$.

On top of this general kind of aggregation whole classes of distance measures can be defined.

Proposition 3.5 *The functions* $d_q^{(n)}$ *defined as:*

$$d_q^{(n)}(\vec{x},\vec{y}) = \Theta_q^M(\vec{x}-\vec{y};\vec{v}) \tag{18}$$

make $(\mathbb{R}^n, d_q^{(n)})$ *a metric space.*

The first family is specifically known as (weighted) Minkowskian distances. The measure obtained for $q = 1$ is generally known as the *City-block*. For $q = 2$ we have the Euclidean measure. This class of functions is characterized by an increasing attention to the bigger partial components d_i with increasing values of q, to the point that, in the limit:

$$\lim_{q\to\infty} d_q^{(n)}(\vec{x},\vec{y}) = \sup_{i=1,\ldots,n} d(x_i, y_i)$$

only the biggest value is taken into account. This last measure is known (for obvious reasons) as the *supremum* or *Chebyshev*. In general, though, we are not restricted to such a family of distances, and can make use of many possibilities of combining the individual distances that stem from the combinations of $\vec{v}$, n' and q. Additionally, several other classes of distances can be derived as, for example:

$$d_q^{(n)}(\vec{x},\vec{y}) = \Theta_q^M(\frac{\vec{x}-\vec{y}}{\vec{x}+\vec{y}};\vec{v}) \tag{19}$$

$$d_q^{(n)}(\vec{x},\vec{y}) = \Theta_q^M(\sqrt[q]{\vec{x}} - \sqrt[q]{\vec{y}};\vec{v}) \tag{20}$$

where all the arithmetic operations on vectors are taken to be component-wise. These two classes are "self-normalizing" and require the vectors not having negative components or being null vectors. The following are classical examples of specific distance measures that can be derived from (18), (19) and (20). Denoting by $\vec{\mu}$, $\vec{\sigma}$ and $\vec{\sigma}_{dev}$, the vector of means, standard deviations and maximum deviation of the variables,

1. (18) with $n' = 1, q = 2, \vec{v} = \vec{1}$ (*basic unweighted Euclidean distance*)
2. (18) with $n' = n, q = 1, \vec{v} = \vec{\mu}$ (*City-block distance weighted by the inverse of the mean*)
3. (18) with $n' = 1, q = 2, \vec{v} = \vec{\sigma}$ (*Pearson distance*)
4. (19) with $n' = n, q = 2, \vec{v} = \vec{1}$ (*Coefficient of divergence or Clark distance*)
5. (19) with $n' = 1, q = 1, \vec{v} = \vec{1}$ (*Canberra distance*)

6. (18) with $n' = n, q = 1, \vec{v} = \vec{1}$ (*unweighted mean City-block distance*)

7. (18) with $n' = n, q = 2, \vec{v} = \vec{\sigma}_{dev}$ (*mean Euclidean distance weighted by the inverse of the maximum deviation*)

8. (20) with $n' = n, q = 2, \vec{v} = \vec{1}$ (*mean Duran-Odell distance*)

It can be noted that (4.) and (7.) above are normalized distances, with $d_{max} = 1$. It is not our intention to devise new distance measures, albeit by combining these factors many variations can be obtained. For convenience, the following simple ways for distance aggregation are derived from (3.3), having the additional advantage of not being restricted to $\mathbb{R}^n$.

Definition 3.4 (Linear operator) *An unary operator π defined on a set X is linear if and only if, for every $x, y \in X$:*

1. $\pi(cx) = c\pi(x), \qquad c \in \mathbb{R}$
2. $\pi(x + y) = \pi(x) + \pi(y)$

Definition 3.5 (n-Linear operator) *A n-ary operator is n-linear if it is linear on each of its arguments.*

Proposition 3.6 *Given $n \in \mathbb{N}^+$ and a collection of sets $X_1, X_2, \ldots, X_n$ for which corresponding distance functions $\vec{d} = \{d_1, d_2, \ldots, d_n\}$ have been defined, $d_i \in D(X_i)$, we denote $X = X_1 \times X_2 \times \ldots \times X_n$ (the usual Cartesian product). Then, for any n-linear Θ and non-negative $\vec{v}$, the pair $(X, \Theta(\vec{d}; \vec{v}))$ is a metric space.*

Proposition 3.7 *Given a non-negative weighting $\vec{v}$, the following is a valid n-linear Θ operator:*

$$\Theta(\vec{d}; \vec{v}) = \sum_{i=1}^{n} v_i d_i \qquad (\textit{weighted summation}) \tag{21}$$

In the common situation where the partial distances are all normalized to the same range of values, it is possible to devise specific aggregation operators. For example, let us suppose that, in the same conditions of Proposition (3.6), it holds $z_i = \frac{d_i}{v_i}$ such that $\forall i : 1 \leq i \leq n$: $z_i : X_i \times X_i \rightarrow [0, z_{max}]$. Then, any Θ is a function of the form

$$[0, z_{max}]^n \rightarrow [0, z_{max}]$$

Proposition 3.8 *For normalized distances $\vec{z} = \{z_1, z_2, \ldots, z_n\}, z_i \in [0, z_{max}]$, the following is a valid Θ operator:*

$$\Theta(\vec{z}) = \frac{1}{\sqrt[q]{n}} ||z||_{L_q}, \;\; q \geq 1 \in \mathbb{R}$$

(normalized modulus)

Proposition 3.9 *For normalized distances $\vec{z} = \{z_1, z_2, \ldots, z_n\}, z_i \in [0, z_{max}]$, the following is a valid n-linear Θ operator:*

$$\Theta(\vec{z}) = \frac{1}{n} \sum_{i=1}^{n} z_i \qquad (\textit{arithmetic mean})$$

These are very simple and effective possibilities for aggregation families and shall be used through-out. Notice that an additional weight can be assigned to individual components of distance measures (not for normalization purposes but to express relative *relevance* of components), in the form $\Theta(\vec{d}; \vec{v}; \vec{w})$ where each w_i multiplies z_i *before* the application of the aggregation operator. The condition is again that all the w_i be positive to preserve monotonicity. Normalized distances require also the condition

$$\sum_{i=1}^{n} w_i = 1$$

Although it can be an interesting feature, these relevance weightings are not considered here, because relative relevance assessment is assumed to be part of the task of the ANN. Normalization should suffice to equal all variables to a common scale before learning takes place; in any case, the previous Propositions are still valid. We now turn our attention to the problem of defining similarity measures.

3.2 The definition of similarity

In classical AI systems, the notion of similarity appears in learning by analogy and case-based reasoning (CBR), where learning by analogy and instance-based learning (IBL) fuse. In particular, similarity coefficients have had a long history in the literature of cluster analysis and data clustering algorithms [19]. However, similarity has been studied in its own right more thoroughly in psychology –and, in particular, in cognitive psychology– than in AI. Firstly, there is a well established school rooted in *psychophysics* which takes the form of geometric models. These models represent the objects as points in a high-dimensional coordinate space where a distance between points is to be defined based on the given data. This distance measure is to reflect the observed similarities and should correspond as closely as possible to the *objective* similarity between the objects that are represented by the points, which could be any kind of data with information about objective or empirical similarities (proximities) between pairs of objects. Secondly, we find the general notion of similarity defined in set-theoretic terms –generally known as the *contrast* model– discussed in the work of Tversky [14], which summarizes a large portion of relevant literature and introduces a formal theory of similarity. According to it, a similarity judgement needs the existence of a essentially unique continuous real-valued function s that expresses a degree of similarity in a clearly defined sense. Such a function is presumably bounded and, taking two objects a and b represented by the feature sets A and B, respectively, expresses the (bounded) degree to which a and b are regarded as similar, written $s(a, b)$. This way, one can state comparisons of the type $s(a, b) > s(c, d)$ to say that object a is more similar to object b than object c is to object d. This endows s with a *semantics* expressing how much two objects resemble each other[5].

In cognitive psychology, formulation and representation of similarity has been a key issue. Understanding how (and whether) biological perceptual systems represent and use similarities has been notoriously difficult to formalize, mainly because of the feature selection problem, relevance assessment, integration of a context into the definition, transitivity considerations and others. In any case, there is an agreement: it is easier to respond intelligently to a stimulus if one can recall previous responses made under *similar* circumstances. The key point for a succeeding perceptual system (biological or not) is thus its ability to represent internally the similarities between the different stimuli [15]. We would like to bring to attention the analogy of this reasoning to the Principles formulated in the Introduction.

In the ANN paradigm, these aspects are somewhat mitigated because, either they are not addressed (like transitivity) or assumed discovered by the network (like the assignment of relevance to input features). As for context, a basic assumption in ANN is that all the patterns are measured along the same fixed set of features. This means that some form of prior knowledge is implicit in the feature selection process and the neuron models are designed taking this into account. Nonetheless, in some recurrent architectures there is an explicit provision for context.

[5]Resemblance is the term used by Hume [16].

3.3 Similarity measures

Let us represent objects, patterns, input stimuli, entities, or however we originally call them, belonging to a space X (of which nothing is assumed, apart that $X \neq \emptyset$) as a vector $\vec{x_i}$ of n components, where each component x_{ij} represents the value of a particular feature (descriptive variable) a_j for object i, from a predefined set of features $A = \{a_1, a_2, \ldots, a_n\}$, judged by the investigator as relevant to the problem. These "vectors" can be thought of as points in a n-dimensional vectorial space; this is possible provided that all the components have a dimension (i.e., there is a total order $\leq_{a_j}$ for all $a_j \in A$) and is irrelevant for the following definitions. It is assumed that the precise order of the $a_j \in A$ is of no importance, provided, of course, that is kept for all the $\vec{x_i}$. This is the usual representation system in which the components x_{ij} of a given object are the values that this object takes for the variables a_i in A. Now, the main question that arises is: how *like* are two $\vec{x_i}, \vec{x_j}$ in X?

We start by calling this vague notion of "likeness" a *proximity* index, one expressing the overall degree of either likeness or resemblance –the *similarity*– or the opposite –the *dissimilarity*. That is, a proximity index (or proximity measure) is a unique number expressing how *like* two given objects are, by only taking into account the selected set of features characterizing them.

For simplicity, let us denote by p_{ij} the proximity between $\vec{x_i}$ and $\vec{x_j}$, that is, $p : X \times X \rightarrow \mathbb{R}^+ \cup \{0\}$ and $p_{ij} = p(\vec{x_i}, \vec{x_j})$. To begin with, we differentiate among two kinds of proximity indexes: similarity indexes s_{ij} and dissimilarity indexes δ_{ij}, defined as p above.

Definition 3.6 (Similarity, dissimilarity) *A proximity index*[6] *has to fulfill the following properties [13]:*

1. Non-negativity. *The p_{ij} cannot be negative.*

$$s_{ij} \geq 0 \tag{22}$$
$$\delta_{ij} \geq 0 \quad \forall \vec{x_i}, \vec{x_j} \in X \tag{23}$$

2. Symmetry. *The p_{ij} do not depend on the order of i, j.*

$$s_{ij} = s_{ji} \tag{24}$$
$$\delta_{ij} = \delta_{ji} \quad \forall \vec{x_i}, \vec{x_j} \in X \tag{25}$$

3. Boundedness. *This is where s_{ij} and δ_{ij} diverge. There is a maximum similarity (namely, that of an object with itself) and a minimum dissimilarity (idem).*

$$s_{ij} \leq s_{max} \tag{26}$$
$$\delta_{ij} \geq \delta_{min} \quad \forall \vec{x_i}, \vec{x_j} \in X \tag{27}$$

4. Minimality *(Reflexivity in the strong sense). The extreme values are attained for equal objects, and only for them.*

$$s_{ij} = s_{max} \quad \Leftrightarrow i = j \tag{28}$$
$$\delta_{ij} = \delta_{min} \quad \Leftrightarrow i = j \quad \forall \vec{x_i}, \vec{x_j} \in X \tag{29}$$

5. Semantics. *The semantics of $s_{ij} > s_{ik}$ is that object i is more similar to object j than is to object k. Conversely, the semantics of $\delta_{ij} > \delta_{ik}$ is that object i is more dissimilar to object j than is to object k.*

Two possible (and, otherwise, arbitrary) values are $s_{max} = 1$ and $\delta_{min} = 0$. An index s_{ij} satisfying properties (1.) through (5.) is a *similarity index*. An index δ_{ij} satisfying properties (1.) through (5.) is a *dissimilarity* index. If, in property (4.), we replace the $\Leftrightarrow$ by just $\Leftarrow$, so that two different objects can be assigned s_{max} or δ_{min} (as if they were the same

[6] In this context, the terms *index* and *function* will be used interchangeably.

object) then we obtain, respectively, a *pseudo-similarity* or *pseudo-dissimilarity* index. Note that by *different* objects $\vec{x_i}, \vec{x_j}$ we mean that $\exists k : 1 \leq k \leq n : x_{ik} \neq x_{jk}$. A space X where a proximity index p_{ij} has been defined forms a semi-metric space, denoted (X, p), being p either s or δ. Other, more restrictive properties can be further demanded to such a space:

6. *Triangular inequality.* This property asks that any three different objects i, j, k are either aligned or forming a triangle, and can only be in general fulfilled by a dissimilarity.

$$\delta_{ij} \leq \delta_{ik} + \delta_{kj} \quad \forall \vec{x_i}, \vec{x_j}, \vec{x_k} \in X \tag{30}$$

   This is equivalent to say that, given two different objects $\vec{x_i}, \vec{x_j}$, their dissimilarity must always be less or equal than the sum of their respective dissimilarities to a third object $\vec{x_k}$. In other words, it is impossible for the way from objects i to j via a third object k to be less than the direct way from i to j. A dissimilarity or distance index fulfilling this property is called a *metric distance*, which will be denoted as d. Notice that this ultimately corresponds to the definition of distance given in (3.1). In what regards to similarities, it is perfectly possible for a similarity to either fulfill it or violate it (see Fig. 6).

7. *Ultrametric inequality.*

$$\delta_{ij} \leq max(\delta_{ik}, \delta_{kj}) \quad \forall \vec{x_i}, \vec{x_j}, \vec{x_k} \in X \tag{31}$$

   This is equivalent to say that, given two different objects $\vec{x_i}, \vec{x_j}$, their dissimilarity (now metric distance) must always be less or equal than the maximum of their respective dissimilarities to a third object $\vec{x_k}$, asking that any three different objects $\vec{x_i}, \vec{x_j}, \vec{x_k}$ must be always forming an isosceles or equilateral triangle.

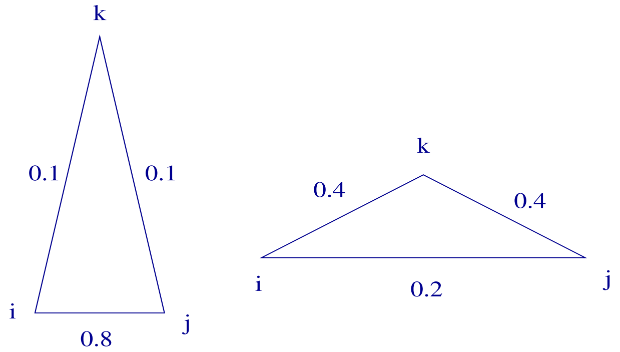


Figure 6: Two scenarios for the same similarity measure s_{ij}. Left: i and j are very similar between each other and both are very dissimilar to k ($s_{ij} \geq s_{ik} + s_{kj}$). Right: i is moderately similar to k and k to j, and thus i is very dissimilar to j ($s_{ij} \leq s_{ik} + s_{kj}$).

The last two properties are particular cases of the more general statement:

$$\delta_{ij} \leq \{(\delta_{ik})^r + (\delta_{kj})^r\}^{\frac{1}{r}} \tag{32}$$

$\forall \vec{x_i}, \vec{x_j}, \vec{x_k} \in X, r \geq 1$ for $r = 1$ and $r = \infty$, respectively.

3.4 Aggregation of similarities

The way different partial similarities should be combined to give an overall score is by no means clear. The aggregation operator fulfills also a *semantic* role. Semantics are very important and is a means to express psychological aspects of similarity; for example, the presence of a particular attribute is neither a necessary nor a sufficient condition for the overall similarity judgement [36]. In this vein, a clearly defined semantics has to be stated. In our case, the adopted posture follows:

1. Even small contributions can only *add* something in favour for the overall measure;

2. The eventually missing pieces are regarded as *ignorance* and do not contribute in favour nor against the overall measure.

The following is a proposal of similarity aggregation formalization –an elusive topic [47]– through a compilation of desirable properties. Also, the definition is made easier by the fact that the partial values are, by definition, normalized. Incidentally, in comparison to Definition (3.2), the minimality condition can be relaxed and the other two conditions are still useful. Notice that monotonicity reflects point (1.) above.

For $z \in \mathbb{R}, n \in \mathbb{N}^+$ and an operator T, define:

$$z^n[T] = \begin{cases} z & \text{if } n = 1 \\ T(\underbrace{z, z, \ldots, z}_{n \text{ times}}) & \text{if } n > 1 \end{cases} \tag{33}$$

Definition 3.7 (Θ_s aggregation operator) *Consider a collection of n quantities, grouped by convenience as a vector $\vec{s} = \{s_1, s_2, \ldots, s_n\}$, where $s_i \in [0, s_{max}] \subset \mathbb{R}$. A similarity aggregation operator Θ_s is a function*

$$\Theta_s : [0, s_{max}]^n \to [0, s_{max}]$$

fulfilling:

i) *Minimality.* $\Theta_s(\vec{s}) = 0 \Rightarrow \exists i : \; s_i = 0$ *and* $(\forall i : \; s_i = 0) \Rightarrow \Theta_s(\vec{s}) = 0$. *This formulation allows (though not ensures) the 0 as an absorbing element.*

ii) *Symmetry.* $\Theta_s(\vec{s}) = \Theta_s(\vec{s}\,')$ *for any* $\vec{s}\,'$ *permutation of* $\vec{s}$.

iii) *Monotonicity.* $\Theta_s(\vec{s}) > \Theta_s(\vec{s}\,')$ *whenever there exists a* s_i, $1 \leq i \leq n$ *such that*

$$s_i > s'_i \wedge \forall j : 1 \leq j \leq n \wedge j \neq i : s_j = s'_j$$

iv) *Idempotency. For an (arbitrary)* s_i, $s_i^n[\Theta_s] = s_i, \forall n \geq 1$. *Note that this specifically includes the boundary values* $s_{max}^n[\Theta_s] = s_{max}$ *and* $0^n[\Theta_s] = 0$. *Idempotency implies* $s_i^{n+1}[\Theta_s] = s_i^n[\Theta_s], \forall n \geq 1$. *A less restrictive condition is to require:*

   (a) *Idempotency for boundary values;*

   (b) *The terms* $\{s_i^n[\Theta_s]\}_{n \geq 1}$ *form a monotonic succession.*

   This property will be denoted relaxed *idempotency.*

v) *Cancellation law.* $\Theta_s(\{s_1, s_2\}) = \Theta_s(\{s_1, s_3\})$ *and* $s_1 > 0$ *implies* $s_2 = s_3$.

vi) *Continuity.* Θ_s *should be continuous in all its arguments so the reactions to small changes in an* s_i *are not "jumpy". This property also ensures that all possible values of* $\Theta_s(\vec{s})$ *can in principle be generated:*

$$\forall s : s \in [0, s_{max}] : (\exists \vec{s}^{\,*} : \; s = \Theta_s(\vec{s}^{\,*}))$$

vii) *Compensativeness.* Θ_s *should be a Cauchy mean, i.e.,*

$$\min_i s_i \leq \Theta_s(\vec{s}) \leq \max_i s_i$$

   A Cauchy mean[7] is such that a good (bad) score s_i *can be compensated by a bad (good) score on another* s_j.

This last condition is a way of introducing a *specific* semantics in the aggregation, and others should be possible. Note that some measures, like the product with arguments in $[0, 1]$, would violate it. It will thus be considered as an optional requirement, although the majority of measures used in practice are to comply with it. Notice also that the *max* and *min* functions would not fulfill strict monotonicity.

Remark 3.2 *Weighted operators* $\Theta(\vec{s}; \vec{v})$ *are also valid, provided they make the operator fulfill the above conditions.*

Proposition 3.10 *A measure* s *obtained from any such aggregation operator* $s = \Theta_s(\vec{s})$, *provided* $\{s_1, \ldots, s_n\}$ *are also similarities* s_i *in* X^i, *fulfills the properties of Definition (3.6), making* s *a similarity measure in* $X = X_1 \times X_2 \times \ldots \times X_n$.

[7]This definition has been borrowed from [5].

Proposition 3.11 *The following are valid families of similarity aggregation functions:*

1. *Additive measures of the form*

$$\Theta(\vec{s};\vec{v}) = f^{-1}\left(\sum_{i=1}^{n} v_i f(s_i)\right)$$

where f is a strictly increasing function $f:[0,s_{max}] \to [0,s_{max}]$ such that $f(0) = 0$ and $f(s_{max}) = s_{max}$, and $\sum_{i=1}^{n} v_i = 1$, with $v_i \geq 0$. As an example, consider $f(z) = z^m, m \in \mathbb{Z}, m \neq 0$. For $m = 1$ we get the weighted arithmetic mean, which further reduces to a pure arithmetic average setting $v_i = 1/n$. For $m = -1$ a weighted harmonic mean is obtained.

2. *Multiplicative measures of the form*

$$\Theta(\vec{s};\vec{v}) = f^{-1}\left(\prod_{i=1}^{n} f(s_i)^{v_i}\right)$$

with f in the same conditions and $\prod_{i=1}^{n} v_i = 1$, with $v_i \geq 0$. Consider again $f(z) = z^m, m \in \mathbb{Z}, m \neq 0$. The choice $m = n$ and $v_i = 1/n$ leads to the geometric mean, whereas $m = v_i = 1$ yields a pure correlation[8] for $s_{max} = 1$. Note that pure correlation fulfills the relaxed idempotency.

3. *The normalized modulus*

$$\Theta(\vec{s}) = \frac{1}{\sqrt[q]{n}}||\vec{s}||_{L_q},\ q \geq 1 \in \mathbb{R}$$

This is an interesting special case of an additive measure by taking $v_i = \frac{1}{n}$ and $f(z) = z^q, q \in \mathbb{N}^+$, fulfilling all the properties in (3.7).

Notice that nothing is said about the degree of linearity of the resulting measure. It is useful to introduce a function that –regardless of its linear or non-linear character– maintains the similarity properties when applied to one such measure.

[8] For $s_{max} = 1$ there are no restrictions on $\vec{v}$.

Definition 3.8 (Similarity keeping) *A similarity keeping function (denoted $\breve{s}$) is a strictly increasing monotonic and continuous function $\breve{s}:[0,s_{max}] \to [0,\breve{s}_{max}]$ such that $\breve{s}(0) = 0$ and $\breve{s}(s_{max}) = \breve{s}_{max} > 0$.*

Note that these functions keep (strict) monotonicity when composed to another (strictly) monotonic function and, since every strictly monotonic function is injective, when applied to a measure obtained by aggregation, the original properties that the aggregation operator as a whole could have –of particular interest, (vi)– are kept.

Proposition 3.12 *Let s be a similarity function in X defined on $[0,s_{max}]$ and $\breve{s}$ a similarity keeping function (either linear or non-linear). Then $\breve{s} \circ s$ is a similarity function in X.*

The final piece for constructing general similarity measures is a transformation operator to obtain a similarity out of a metric distance.

Definition 3.9 (Similarity transforming) *A similarity transforming function (denoted $\hat{s}$) is a strictly decreasing monotonic and continuous function $\hat{s}:[0,+\infty) \to [0,s_{max}]$ such that $\hat{s}(0) = s_{max}$ and $lim_{z\to+\infty}\hat{s}(z) = 0$.*

Proposition 3.13 *Let d be a distance function defined on X and $\hat{s}$ a similarity transforming function (either linear or non-linear). Then $s(x,y) = \hat{s}(d(x,y))$ is a similarity function in X, for any two $x,y \in X$.*

Proposition 3.14 *Let $\breve{s},\hat{s}$ be two similarity keeping and transforming functions, respectively. Then $\breve{s} \circ \hat{s}$ is a similarity transforming function.*

In case the original distance is normalized, the preceding statements can be further refined.

Proposition 3.15 *Let $d : X \to [0, d_{max}]$ be a normalized distance function defined on X and $\hat{s}$ a similarity transforming function (either linear or non-linear). Then, for an $\hat{s}$ such that $\hat{s}(d_{max}) = 0$, the function $s(x, y) = \hat{s}(d(x, y))$ is a similarity in X.*

Proposition 3.16 *For every $\hat{s}$ function defined for a non-normalized distance, there exists another $\hat{s}$ function defined for a normalized one in $[0, d_{max}]$, of the same shape, and adhering to the conditions of Proposition (3.15).*

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

- Linear: $\hat{s}_0(z) = d_{max} - z$, for $z \in [0, d_{max}]$ $(s_{max} = d_{max})$.
- Inverse function: $\hat{s}_1(z) = \frac{a}{1+z}$ for $z \in [0, \infty), a > 0$ $(s_{max} = a)$.
- Cauchy function: $\hat{s}_2(z) = \frac{a}{1+z^2}$ for $z \in [0, \infty), a > 0$ $(s_{max} = a)$.
- Exponential function: $\hat{s}_3(z) = e^{-z}$, for $z \in [0, \infty)$ $(s_{max} = 1)$.
- Inverted logistic: $\hat{s}_4(z) = a(1 - \sigma(z))$, for $z \in [0, \infty), a > 0$ $(s_{max} = a/2)$.

  where $\sigma(z)$ is the logistic function.

Some of these functions (like $\hat{s}_0, \hat{s}_1$ and $\hat{s}_2$) are found through-out in data analysis literature [13, 19] and will later on be generalized conveniently (Section 5.2).

3.5 The design of similarity measures

In general, we can devise a similarity measure either directly or by conversion from another proximity measure such as a dissimilarity, a distance or a pre-existent similarity. Measures in more than one dimension are constructed by making use of the aggregation operators defined in (3.2) and (3.7).

$$\text{Similarity} \begin{cases} \text{from distance} \begin{cases} \text{globally} & (a) \\ \text{locally} & (b) \end{cases} \\ \text{directly } (c) \\ \text{from dissimilarity } (d) \end{cases}$$

The first two types (a) and (b) are distance-induced similarity functions. The difference stands in their local/global character, as described in Section (1.5). In (a), similarity is globally defined from a distance which is in turn decomposed in partial distances. In (b), similarity is built out of the aggregation of local (partial) similarities, each in turn coming from a (partial) distance.

In general, using a distance as a measure of the similarity between two objects expresses the idea that if two points are close together then they are relatively similar in terms of the features represented by their coordinates. Let X^n be a space of dimension n built out of the Cartesian product of one-dimensional spaces $X = \prod_{i=1}^{n} X^i$, in which a similarity relation has to be defined and let $\vec{x}, \vec{y} \in X^n$. The case (a) stands for a similarity s defined by means of a transformation from a metric distance in X^n, as follows:

$$s(\vec{x}, \vec{y}) = \hat{s}(\Theta(\vec{d})) \qquad (34)$$

where $\vec{d} = \{d_1, \ldots, d_n\}$ is the vector of partial distance measures $d_i \in D(X^i)$ so that $\Theta(\vec{d}) \in D(X^n)$ and thus s is a similarity in X^n.

The case (b) stands for a similarity s defined by aggregation of partial similarities on one-dimensional spaces, each one of them coming from a transformation from a metric distance in those spaces, as follows:

$$s(\vec{x}, \vec{y}) = \breve{s}(\Theta_s(\vec{s})) \qquad (35)$$

where $\vec{s} = \{s_1, \ldots, s_n\}$ is the vector of partial similarity measures s_i defined on X^i as $s_i = \hat{s}(d(x_i, y_i))$ with $d \in D(X^i)$ and Θ_s is a similarity aggregation operator, so that s is a similarity in X^n. In the most general situation, the partial distance measure need not be equal, with $s_i = \hat{s}(d_i(x_i, y_i))$. Eventually, a further similarity keeping function can be applied to produce the overall similarity $\breve{s}(s)$.

The case (c) stands for a *direct* measure s, either by aggregation of partial direct measures,

$$s(\vec{x}, \vec{y}) = \breve{s}(\Theta_s(\vec{s})) \tag{36}$$

–where $\vec{s} = \{s_1, \ldots, s_n\}$ is the vector of direct partial similarity measures $s_i(x_i, y_i)$ defined on X^i– or else as a globally defined similarity that takes relations between components into account, as the one in example 2 below. An example of the former and perhaps the most widely used direct similarity, the scalar product is the more basic measure of linear dependency. It fits this scheme by taking equal $s_i(x_i, y_i) = x_i y_i$ and Θ_s to be a simple summation. Other, more flexible functions, are angular similarity, the covariance, and the correlation coefficient, which have seen a widespread and polemic use as similarity measures in classical data analysis techniques [13].

The last situation (d) depicts the case in which a similarity measure is defined by transformation from a dissimilarity, in the form:

$$s(\vec{x}, \vec{y}) = \breve{s}(\hat{s}(\delta(\vec{x}, \vec{y}))) \tag{37}$$

where δ is a dissimilarity measure as defined in (3.6), conveniently transformed onto a similarity. In general, there are multiple possibilities for devising direct similarity or pseudo-similarity indexes, even in cases for which an Euclidean (or Minkowskian) metric could in principle have been used. Consider the following two scenarios.

1. Given two vectors $\vec{x}, \vec{y} \in \mathbb{R}^n$ the similarity we wish to apply should reflect the degree to which the two vectors have all their respective components different (i.e., to what degree the number of different $x_1, \ldots, x_n$ matches the number of different $y_1, \ldots, y_n$). Euclidean distance (or any member of its family, weighted or not) will not capture this simple relation. Even by normalizing the vectors to unit norm, the problem is still unsolved. Denoting

   $$N(\vec{z}) = n - \#i : 1 \le i \le n : (\exists j : i < j \le n : x_i = x_j) \tag{38}$$

   we have

   $$d(\vec{x}, \vec{y}) = \frac{|N(\vec{x}) - N(\vec{y})|}{n-1}$$

   This is not a distance in $\mathbb{R}^n$, because it violates the first condition. However, since we have $d_{max} = 1$, the transformation $s(\vec{x}, \vec{y}) = 1 - d(\vec{x}, \vec{y})$ is a pseudo-similarity measure in $\mathbb{R}^n$ (with $s_{max} = 1$)

2. Given two vectors $\vec{x}, \vec{y} \in \mathbb{R}^n$, possibly coding some form of a temporal data series, we regard them as more similar the higher is their number of common elements. The number of such points can be defined as:

   $$N(\vec{x}, \vec{y}) = \#i : 1 \le i \le n : (\exists j : 1 \le j \le n : x_i = y_j) \wedge (\forall j : 1 \le j < i : x_j \ne x_i) \tag{39}$$

   and a direct measure can be defined:

   $$s(\vec{x}, \vec{y}) = \frac{N(\vec{x}, \vec{y})}{n}$$

   that is a (pseudo) similarity in $\mathbb{R}^n$, with $s_{max} = 1$.

In the first example, although the measure $d(\vec{x}, \vec{y})$ proposed *fulfills* the triangular inequality –the hardest of the three conditions in (3.1)– the violation of the first condition prevents us to define a proper metric; therefore, on the one hand, scalar product or Euclidean measures will have a hard time discovering the similarity relation, which is in fact a simple one, *once it is known.* On the other, we are not able to devise an alternative distance measure that correctly captures the similarity in pattern space. In the second example, the underlying relation is not decomposable componentwise. In both situations, the relation can be defined in terms of a (in this case, pseudo) similarity and still get a working model.

An immediate derivation of the four defined types of similarity is that they can be combined, as long as a similarity (or pseudo-similarity) measure is devised, so as to keep within the introduced framework, leading to potentially more general and useful measures.

3.6 Proper scalar product-based similarity

In previous Sections we have seen how distance and similarity are opposite concepts (the more the distance, the less the similarity, and viceversa). In order to cast the classical neuron models defined in (2.3) and (13) as similarity computing devices, it is necessary to design a proper scalar product-based measure – by proper we mean fulfilling the required properties in Definition (3.7). To this end, it is useful to examine their precise relationship. Let us assume, for simplicity, an unweighted Euclidean distance d in n-dimensions. We have:

$$d^2(\vec{x}-\vec{y}) = ||\vec{x}-\vec{y}||^2 = \sum_i^n (x_i - y_i)^2$$

$$= \sum_i^n x_i^2 + \sum_i^n y_i^2 - 2\sum_i^n x_i y_i$$

$$= ||\vec{x}||^2 + ||\vec{y}||^2 - 2\vec{x}\cdot\vec{y} =$$

At this point we see that they are effectively inverses. Now, assuming normalized vectors, as in (7):

$$= 2 - 2\vec{x}\,'\cdot\vec{y}\,' = 2(1-\vec{x}\,'\cdot\vec{y}\,')$$

Thus, *if* both vectors are normalized, then scalar product is a direct function of the distance, with no other factor involved:

$$\vec{x}\,'\cdot\vec{y}\,' = 1 - \frac{d^2}{2} \in [-1,1] \tag{40}$$

where $d \in [0,2]$. Therefore,

$$s(\vec{x}\,',\vec{y}\,') = \frac{\vec{x}\,'\cdot\vec{y}\,'+1}{2} \tag{41}$$

is a neat similarity function of type (a) in $[0,1]$, easily obtained by substituting (40) in (41) and applying Proposition (3.13) to Euclidean distance using $\hat{s}(z) = 1-(z/2)^2$ for $z = d(\vec{x}\,',\vec{y}\,')$ as:

$$s(\vec{x}\,',\vec{y}\,') = \hat{s}(d(\vec{x}\,',\vec{y}\,'))$$

This is a translated and scaled form of a measure usually known as *angular similarity* [13]:

$$s(\vec{x}\,',\vec{y}\,') = s\left(\frac{\vec{x}}{||\vec{x}||},\frac{\vec{y}}{||\vec{y}||}\right) =$$

$$\left(\frac{\vec{x}\cdot\vec{y}}{||\vec{x}||\,||\vec{y}||}+1\right)\frac{1}{2} = \frac{\cos(\theta)+1}{2} \tag{42}$$

where $\theta \in [0,\pi]$ is the angle between $\vec{x}\,'$ and $\vec{y}\,'$. However, the normalization condition for both vectors is rarely found in ANNs, and renders the measure expressed in (41) useless in practice. Unfortunately, if this condition is not met, the expression in (42) is *not* a proper similarity measure, since it does not fulfill condition (4) in Definition (3.6). It is, instead, a pseudo-similarity. This is caused by the fact that *different* vectors can be considered as maximally similar: it suffices that they are co-linear and point in the same direction.

To get a full compliant scalar product-based similarity, let

$$\begin{cases} \pi^+_{(K)} = & \sup_{\vec{x},\vec{y}\in K\subset\mathbb{R}^n} \; \vec{x}\cdot\vec{y} \\ \pi^-_{(K)} = & \inf_{\vec{x},\vec{y}\in K\subset\mathbb{R}^n} \; \vec{x}\cdot\vec{y} \end{cases} \tag{43}$$

to denote $\forall \vec{x}, \vec{y} \in K: \ \vec{x} \cdot \vec{y} \in [\pi^-_{(K)}, \pi^+_{(K)}]$. It is then assumed that the target function (the function to be approximated by the neural network) has a compact support K, being $\mathbb{R}^n$ the space where input patterns are drawn from, so that there exists a $\vec{z}^*$ such that

$$\vec{z}^* = argmax\, \vec{z}: \ \vec{z} \in K \subset \mathbb{R}^n : ||\vec{z}||^2$$

Since (43) is a continuous function defined in a closed interval, we have $\pi^+_{(K)}, \pi^-_{(K)} \in \mathbb{R}$, and the measure:

$$s(\vec{x}, \vec{y}) = \frac{\vec{x} \cdot \vec{y} - \pi^-_{(K)}}{\pi^+_{(K)} - \pi^-_{(K)}} \in [0,1], \ \vec{x}, \vec{y} \in K \quad (44)$$

is a similarity in K. For simplicity, let $K = [m, M]^n \subset \mathbb{R}^n, m \leq 0, M \geq 0$. In these conditions, define $\vec{\beta} = (\beta, \cdots, \beta)$ of dimension n for any scalar β and denote $\mu = max(|m|, M)$. The maximum and minimum scalar products in K are given by:

$$\begin{cases} \pi^+_{(K)} = & ||\vec{\mu}||^2 \\ \pi^-_{(K)} = & -||\vec{m}||||\vec{M}|| \end{cases} \quad (45)$$

A further simplification can be obtained by considering the particular case $m = -M$ (a symmetric zero-centered interval). In this situation, $\vec{x}, \vec{y} \in K, \ \vec{x} \cdot \vec{y} \in [-||\vec{M}||^2, ||\vec{M}||^2]$, and

$$s(\vec{x}, \vec{y}) = \frac{\vec{x} \cdot \vec{y} + ||\vec{M}||^2}{2||\vec{M}||^2} \in [0,1] \quad (46)$$

is a similarity measure in K.

3.7 Neural models of similarity

We are now in a position to summarize the casting of neuron models as similarity computers, in the following Propositions:

Proposition 3.17 *A r-neuron model of the kind defined in (2.3) computes a similarity measure s in $\mathbb{R}^n$ of the form $F_i(\vec{x}) = s(\vec{x}, \vec{w}^{\,i})$.*

Proof: these models fall within type (a) similarities $s(\vec{x}, \vec{w}^{\,i}) = \hat{s}(d^{(n)}_q(\vec{x}, \vec{w}^{\,i}))$, as described in Section 3.5, with $d^{(n)}_q(\vec{x}, \vec{w}^{\,i}) = \Theta^M_q(\vec{x} - \vec{w}^{\,i}; \vec{1})$ and $n' = 1$ (18) defined in $\mathbb{R}^n$, followed by a similarity transforming function $\hat{s}(z) = g(z)$. By Proposition (3.13), this is a similarity in $\mathbb{R}^n$.

Proposition 3.18 *For normalized vectors, a p-neuron model of the kind defined in (13) computes a similarity measure s in $\mathbb{R}^n$ of the form $F_i(\vec{x}) = s(\vec{x}, \vec{w}^{\,i})$.*

Proof: in these conditions, this model behaves as a r-neuron in (2.3), by taking $q = 2$ and $\hat{s}(z) = g(1 - (z/2)^2)$. By Propositions (3.14), (3.17), this is a similarity in $\mathbb{R}^n$.

Proposition 3.19 *For non-normalized vectors, a p-neuron model of the kind defined in (13) computes a pseudo-similarity measure s in $\mathbb{R}^n$ of the form $F_i(\vec{x}) = s(\vec{x}, \vec{w}^{\,i}) = \breve{s}(\vec{x} \cdot \vec{w}^{\,i})$.*

Proof: the general scalar-product based model[9] falls within type (c) or correlation-based similarities, plus an additional similarity keeping function $\breve{s}(z) = g(z)$, as described in Section 3.5. We prove it by fulfillment of the conditions in Definition (3.6).

1. $s(\vec{x}, \vec{w}^{\,i}) \geq 0$. This will depend on the exact form of g. Many of the most common activation functions are non-negative –such as the logistic in (10). In general, a transformation $g \to g'$ of the form

$$g'(z) = \frac{g(z) - g_{min}}{g_{max} - g_{min}}$$

[9] The bias term θ is considered as included in the summation.

makes $g'_{min} = 0$ and does not no alter the characteristics of g.

2. $s(\vec{x}, \vec{w}^{\,i})$ is symmetric because scalar product is.

3. $s(\vec{x}, \vec{w}^{\,i})$ is bounded (asymptotically) by g_{max}.

4. Minimality is not fulfilled, as illustrated above for (42).

5. The semantics of this similarity is described in Section (1.2).

Since the failing condition is minimality, this is a pseudo-similarity measure. Notice that, in all these cases, the partial measures used are all equal. This enables to express the overall computation in the usual vectorial form.

Proposition 3.20 *For non-normalized vectors, the measure defined in (44) falls within the class of models defined in (13) and is a similarity measure in $K \subset \mathbb{R}^n$.*

Proof: First, we prove that s in (44) is a similarity. Since $s \in [0,1]$, the non-negativity and boundedness conditions (with $s_{max} = 1$) are easily met. Symmetry and semantics also hold as in the previous Proposition. Minimality is now met because

$$s(\vec{x}, \vec{w}^{\,i}) = 1 \Longleftrightarrow \vec{x} = \vec{w}^{\,i} = \vec{z}^{\,*}$$

Second, note that the composition of a similarity keeping function $\check{s}$ to $s(\vec{x}, \vec{y})$ in (44) makes the result a similarity, by Proposition (3.12). Third, every function g –of a given parameterized family– of the required form in (13) can be transformed to a g' by choosing a different set of parameters (acting as constants) so that $g'(\vec{x} \cdot \vec{y}) = g(s(\vec{x}, \vec{y}))$.

For example, letting $g = g_{\beta,\theta}$ the classical logistic function as defined in (10):

$$g_{\beta,\theta}(z) = \frac{1}{1 + e^{-\beta(z-\theta)}}, \qquad \beta > 0 \qquad (47)$$

a model of the form:

$$g(s(\vec{x}, \vec{y})) = g(\frac{\vec{x} \cdot \vec{y} + s_{max}}{2 s_{max}})$$

is a p-neuron because the above expression can be equivalently written:

$$g(s(\vec{x}, \vec{y})) = g'(\vec{x} \cdot \vec{y})$$

where $g' = g_{\beta',\theta'}$ is just the same logistic, for

$$\beta' = \frac{\beta}{2 s_{max}}; \theta' = s_{max}(2\theta - 1)$$

This measure offers the additional advantage of being much cheaper to compute than (42). As an illustrative example, consider a collection of data properly normalized to lie in $D^n \subset \mathbb{R}^n = [0,1]^n$ (a common situation in practice, see [32, 34]). Then $\pi^+_{(D^n)} = n$, $\pi^-_{(D^n)} = 0$ in (44), and thus:

$$s(\vec{x}, \vec{y}) = \frac{\vec{x} \cdot \vec{y}}{n} \in [0,1]$$

is a similarity in D^n. In later parts of the work (Section 5), the assumption (43) will be additionally justified.

4 Heterogeneous Spaces

A process can be generally described by means of a set of variables that are thought to exert an influence on it. The values of these variables are either collected by the user or generated by a model. In the case of ANNs, they are grouped in the form of input vectors, although the term input *tuple* would seem to be more accurate. Among the data collected there is information of very different nature.

4.1 Definition of heterogeneous variables

We consider in this work the following types of variables:

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.

Continuous : numerical and crisp variable for which a linear order relation has been defined on a continuum of values.

Set : variable whose values are sets.

Fuzzy Number : obtained out of a continuous variable expressing uncertainty in the form of *imprecision*.

Linguistic : obtained out of an ordinal variable expressing uncertainty in the form of *vagueness*.

The values of the last two variables are obtained by converting each of an existing set of crisp values into a fuzzy quantity.

Nominal variables

The most basic similarity measure for these variables is the overlap. Let $\mathcal{N}$ be a *nominal* space and $x, y \in \mathcal{N}$.

$$s(x,y) = \begin{cases} 1 & \text{if } x = y \\ 0 & \text{if } x \neq y \end{cases} \tag{48}$$

Proposition 4.1 *The function defined in (48) is a similarity measure in* $\mathcal{N}$.

Ordinal variables

It is assumed that a *linear order* exists such that the values of the variable form a total linearly ordered space. Let $\mathcal{O}$ be an *ordinal* space and $x, y \in \mathcal{O}$, with the notation $\#\{\cdot\}$ to denote cardinality. Let $\preceq$ be the linear order in $\mathcal{O}$. An equality relation $=$ is also assumed so that strict inequality

$$x' \prec x \equiv x' \preceq x \wedge \neg(x' = x)$$

is defined in the usual way. Let us define now:

$$\eta : \mathcal{O} \to [1, m] \subset \mathbb{N}^{+}$$

as

$$\eta(x) = \#\{x' \in \mathcal{O} : x' \prec x\} + 1, \qquad x \in \mathcal{O}$$

Since the order $\preceq$ is linear, this is a bijection. Since every finite set is well-ordered, $\mathcal{O}$ has a first (x_f) and a last element (x_l), i.e., elements such that $\nexists x' \in \mathcal{O} : x' \prec x_f$ and $\nexists x' \in \mathcal{O} : x_l \prec x'$, respectively. Therefore, $\eta(x_f) = 1$, $\eta(x_l) = \#\{\mathcal{O}\} = m$ and:

$$\eta(x') = \eta(x) + 1 \Leftrightarrow x' = succ(x)$$

where the $\Leftarrow$ holds by definition of η and $\Rightarrow$ by the linear character of $\preceq$. By doing this we can devise a distance in $\mathcal{O}$ resorting to a metric in $\mathbb{R}$, using Propositions (3.1) and (3.3) for $S = \mathcal{O}$, and making use of the fact that η is a bijection. Hence, for $x, y \in \mathcal{O}$, define d as:

$$d(x,y) = |\eta(x) - \eta(y)| \in D(\mathcal{O})$$

A normalized distance can be obtained as:

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

and, by selecting, $\hat{s}(z) = 1 - z$, $s(x,y) = \hat{s}(d(x,y))$ is a similarity measure in $\mathcal{O}$ (though others are possible).

Proposition 4.2 *Provided $\mathcal{O}$ is finite and linearly ordered:*

$$s(x,y) = 1 - \frac{|\eta(x) - \eta(y)|}{\#\{\mathcal{O}\} - 1} \qquad (49)$$

is a similarity measure in $\mathcal{O}$, with $s_{max} = 1$.

Finally note that, in a working implementation, the set $\mathcal{O}$ can be safely replaced by $[1, m] \subset \mathbb{N}^+$ as long as the elements are not arithmetically operated beyond (49).

Continuous variables

Let $x, y \in \Gamma = [r^-, r^+] \subset \mathbb{R}, r^+ > r^-$. By Propositions (3.1) and (3.2), the standard metric d in $\mathbb{R}$ is a metric in Γ. Therefore, any $s(x,y) = \hat{s}(d(x,y))$ is a similarity measure in Γ. A normalized distance can be obtained by setting:

$$d(x,y) = \frac{|x-y|}{\sup_{x,y\in\Gamma} |x-y|} \in [0,1], d \in D(\Gamma) \qquad (50)$$

which is the standard distance weighted by the maximum deviation.

Proposition 4.3 *Given $\Gamma \subset \mathbb{R}$, the function:*

$$s(x,y) = 1 - \frac{|x-y|}{\sup_{x,y\in\Gamma} |x-y|} \qquad (51)$$

is a similarity measure in Γ, with $s_{max} = 1$.

Notice that this is equivalent to the ordinal case, where normalization is performed dividing by the number of elements in the space.

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 $x, y \in \mathcal{S}$.

Proposition 4.4 *Provided $\mathcal{S}$ is finite (so that all its subsets are),*

$$s(x,y) = \frac{\#\{x \cap y\}}{\#\{x \cup y\}}, \qquad x, y \neq \emptyset \qquad (52)$$

is a similarity measure in S, with $s_{max} = 1$.

Note that this measure reduces to (48) for singleton sets.

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 an existing theoretically crisp object. The object is crisp, but its observation or realization is vague [20]. 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 us begin by defining a basic concept [26]:

Definition 4.1 (Fuzzy number)
A fuzzy number in X is a convex and normalized fuzzy set F, with piecewise continuous μ_F, where X is the reference set, and such that $\exists! x \in X : \mu_F(x) = 1$. Symmetry of μ_F is a useful simplifying assumption although is not required.

Let $\mathbb{F}_n(X)$ be the (crisp) set of all the fuzzy numbers in X:

$$\mathbb{F}_n(X) = \{F \subset_f X / \ F \text{ is a fuzzy number}\}$$

where X is assumed a continuum. Here the notation $A \subset_f B$ means that A is a fuzzy subset of B. Among the several possible forms such a fuzzy set can take, two of the most popular are:

Triangular

Correspond to non-symmetric fuzzy sets of the form:

$$\mu_{[m]}^{[\alpha_1,\alpha_2]}(z) = \begin{cases} \frac{z-(m-\alpha_1)}{\alpha_1} & \text{if } z \in [m-\alpha_1, m] \\ \frac{(m+\alpha_2)-z}{\alpha_2} & \text{if } z \in [m, m+\alpha_2] \\ 0 & \text{otherwise} \end{cases} \quad (53)$$

In both situations, m is the mode or center and α_1, α_2 the left and right fuzzyness, respectively. In case the number is symmetric, it may be written in a more compact form:

$$\mu_{[m,\alpha]}(z) = \begin{cases} 1 - \frac{|z-m|}{\alpha} & \text{if } |z-m| \le \alpha \\ 0 & \text{otherwise} \end{cases} \quad (54)$$

Gaussian

Correspond to symmetric fuzzy sets of the form:

$$\mu_{[m,\alpha]}(z) = exp(\frac{z-m}{\alpha})^2 \quad (55)$$

Since it is unlikely that collected data come already in this fuzzified form, there has to be a *fuzzyfication* process, much as in Fuzzy Controllers [28]. In these systems, crisp values u_0, in absence of other information, are transformed onto a fuzzy number of the form:

$$\mu_{u_0}(u) = \begin{cases} 1 & \text{if } u = u_0 \\ 0 & \text{otherwise} \end{cases} \quad (56)$$

In this work, however, a numerical variable will be considered a fuzzy number (and treated as such) *only* if additional information is available; otherwise it is considered as continuous (crisp). One of the forms this information can take is about the reliability or error of measured values, which can be translated into the fuzzy number's left/right fuzzyness.

For every $x \in \mathbb{R}$, let $\tilde{x}$ be a fuzzy number of a given form centered at x and with fuzzyness generally defined as a function of x. The transformation:

$$\tilde{\ } : \quad x \in \mathbb{R} \xrightarrow{\sim} \tilde{x} \in \mathbb{F}_n(\mathbb{R}) \quad (57)$$

associates a fuzzy number $\tilde{x}$ to x where $\mu_{\tilde{x}} = \mu_{[x,\alpha(x)]}$ is a membership function expressing a fuzzy number –as (53) or (55)– and $\alpha(x)$ stands for as many fuzzyness parameters as required. The resulting fuzzy number is to express the *degree of uncertainty*. This interpretation was proposed by Zadeh when introducing possibility theory [21]. A membership expression like $\mu_{\tilde{x}}(u)$ represents the degree of possibility that x has value u. Domain knowledge can be added to refine the transformation. For instance, let us suppose that the value x comes from a measuring device known to be biased towards underestimating the actual value at most a 2%, and overestimating it at most a 1%. A (non-symmetric) fuzzy number –depicted in Fig. (7)– could be derived, where $\mu_{\tilde{x}} = \mu_{[x,\alpha(x)]}$, $\alpha(x) = (\alpha_1, \alpha_2) = (k_1 x, k_2 x)$ and $k_1 = 0.02, k_2 = 0.01$.

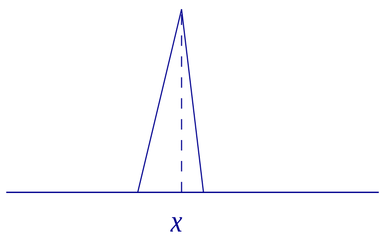


Figure 7: Example form for $\mu_{\tilde{x}}$ (not to scale).

Given two fuzzy numbers, the question is: how similar are they? For variables representing fuzzy sets, similarity relations from the point of view of fuzzy theory have been defined elsewhere [24, 25], and different choices are possible. In possibility theory two functions called *possibility* (Π) and *necessity* (N) measures rank events by the degree of *unsurprisingness* and *acceptance*, respectively [23].

In particular, the first of these functions expresses the possibility of co-occurrence or simultaneity of two vague propositions, with a value of 1 standing for absolute certainty. For two fuzzy sets $\tilde{A}, \tilde{B}$ possibility is defined as:

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

where $\mu_{\tilde{A} \cap \tilde{B}}(u) = \min\left(\mu_{\tilde{A}}(u),\, \mu_{\tilde{B}}(u)\right)$. Notice that this measure is symmetric, in the sense $\Pi_{\tilde{A}}(\tilde{B}) = \Pi_{\tilde{B}}(\tilde{A})$.

Proposition 4.5 *Given $X, Y \in \mathbb{F}_n(\Gamma), \Gamma \subset \mathbb{R}$, and making use of the transformation in (57), the function*

$$s : \mathbb{F}_n(\Gamma), \mathbb{F}_n(\Gamma) \to [0, 1]$$

defined as:

$$s(x, y) = \Pi^*(\tilde{x}, \tilde{y}) \tag{58}$$

where $\Pi^*(\tilde{x}, \tilde{y}) = \Pi_{\tilde{x}}(\tilde{y})$, is a similarity measure in $\mathbb{F}_n(\Gamma)$, with $s_{max} = 1$.

Proof: Since $s \in [0, 1]$, the symmetry, non-negativity and boundedness conditions (with $s_{max} = 1$) are easily met. There is also a clear semantics. Minimality is met because, given $\alpha(x)$ is the same function for all $x \in \mathbb{R}$, it holds that $x = y \Leftrightarrow \tilde{x} = \tilde{y} \Leftrightarrow \Pi^*(\tilde{x}, \tilde{y}) = 1$.

This is a simple and effective similarity measure under the context of *fuzzy numbers*. For more general fuzzy sets, other measures should be defined.

Linguistic variables

The integration of numeric and qualitative information —the latter in the form of fuzzy sets— has been pointed out to have several advantages in the framework of pattern recognition [27], either enriching the basic ideas or giving rise to completely new concepts. The addition of linguistic features for expressing 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. The linguistic approach considers the variables by means of linguistic terms [22] (words in the most basic situation). Since words are less precise than numbers, this approach enables the use of vague information, in cases where a precise quantity is unknown or there is a need or convenience to abstract it out.

Let *supp*(F) denote the *support* of a fuzzy set F and let $[\mathbb{R}]$ denote the set of all finite and closed real intervals:

$$[\mathbb{R}] = \{[a, b] \subset \mathbb{R} \;/\; a, b \in \mathbb{R}, a \leq b\}$$

We set the following definitions:

Definition 4.2 (Fuzzy quantity) *A fuzzy quantity in $\mathbb{R}$ is a normalized fuzzy set F with continuous μ_F such that supp$(F) \in [\mathbb{R}]$.*

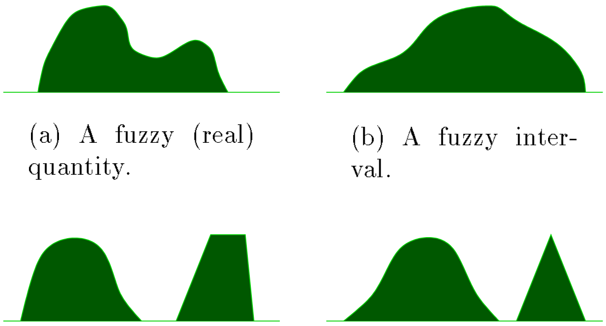
Definition 4.3 (Fuzzy interval) *A fuzzy interval in X is a convex fuzzy quantity in X. Symmetry of μ_F is a useful simplifying assumption although is not required.*

Definition 4.4 (Fuzzy p-quantity) *A fuzzy p-quantity in X is a fuzzy interval in X, 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*[10].

Note that a fuzzy number is a unimodal fuzzy p-quantity. These quantities (depicted in Fig. 8) are useful for modeling the linguistic terms of a linguistic variable, and should be

[10]This is a special case of a fuzzy interval of the LR-type [50], defined for convenience.

accurate enough in most situations. In particular, a p-quantity with support in $[a, b]$ models the vague proposition "roughly between a and b". Among the several possible forms such a fuzzy set can take, the most popular are probably the trapezoidal and bell-shaped.



(a) A fuzzy (real) quantity.

(b) A fuzzy interval.

(c) Two fuzzy p-quantities.

(d) Two fuzzy numbers.

Figure 8: Several specific forms for a continuous fuzzy set in $\mathbb{R}$.

Trapezoidal

Correspond to fuzzy sets of the form:

$$\mu_{[m_1,m_2]}^{[\alpha_1,\alpha_2]}(z) = \begin{cases} \frac{z-(m_1-\alpha_1)}{\alpha_1} & \text{if } z \in [m_1-\alpha_1, m_1] \\ \frac{(m_2+\alpha_2)-z}{\alpha_2} & \text{if } z \in [m_2, m_2+\alpha_2] \\ 1 & \text{if } z \in [m_1, m_2] \\ 0 & \text{otherwise} \end{cases} \tag{59}$$

where $m_2 \geq m_1$ and $\alpha_1, \alpha_2 > 0$ are offsets.

Bell

Correspond to symmetric fuzzy sets of the form:

$$\mu_{[m,\alpha,\beta]}(z) = \frac{1}{1+(\frac{|z-m|}{\alpha})^{2\beta}} \tag{60}$$

Let $\mathbb{F}_q(X)$ be the (crisp) set of all the fuzzy p-quantities in X:

$$\mathbb{F}_q(X) = \{F \subset_f X / \ F \text{ is a fuzzy p-quantity}\}$$

Among the possible similarity measures between these elements, given two fuzzy p-quantities $\tilde{A}, \tilde{B} \in \mathbb{F}_q(\Gamma), \Gamma \in \mathbb{R}$, with support sets $\Gamma_{\tilde{A}}, \Gamma_{\tilde{B}} \in \Gamma$, respectively, define their ratio as:

$$r(\tilde{A}, \tilde{B}) = \frac{\int_{\Gamma_{\tilde{A}} \cup \Gamma_{\tilde{B}}} \mu_{\tilde{A}\cap\tilde{B}}(u)du}{\int_{\Gamma_{\tilde{A}} \cup \Gamma_{\tilde{B}}} \mu_{\tilde{A}\cup\tilde{B}}(u)du} \tag{61}$$

which is the continuous version of one of the classical measures between fuzzy sets [31]. Being these measures general ones, they are not discriminative enough to capture all possible forms two fuzzy sets can take. The context in which they are applied here (fuzzy p-quantities) allows for an accurate assessment of similarity. Note that (61) can be regarded as a generalization of a measure for crisp sets representing subintervals of $\mathbb{R}$, in which a value either belongs or not to the interval. The step to fuzzy sets makes the integral operator necessary. Notice also that, although (58) could in principle be applied, its behaviour in some cases –namely, whenever $\#\{\mu_{\tilde{A}\cap\tilde{B}} = 1\} > 1$– makes it unsuitable as a similarity index (it does not fulfill minimality).

Proposition 4.6 *Given* $\tilde{A}, \tilde{B} \in \mathbb{F}_q(\Gamma), \Gamma \subset \mathbb{R}$*, the function* $s(\tilde{A}, \tilde{B})$

$$s : \mathbb{F}_q(\Gamma), \mathbb{F}_q(\Gamma) \to [0, 1]$$

as defined in (61), for $(\cap = min, \cup = max)$*, is a similarity measure in* $\mathbb{F}_q(\Gamma)$*, with* $s_{max} = 1$.

Proof: Symmetry, non-negativity and boundedness conditions (with $s_{max} = 1$) hold. There is also a clear semantics. Minimality is met because, as in crisp sets, the property

$\tilde{A} \cap \tilde{B} = \tilde{A} \cup \tilde{B} \Leftrightarrow \tilde{A} = \tilde{B}$ holds (provided $\{\cap = min, \cup = max\}$ are used).

These variables are to be used whenever enough domain knowledge is available so as to define linguistic terms instead of simple ordinals. It is assumed that the ordering $\preceq$ in $\mathcal{O}$ is respected, since this is a basic knowledge. Denoting $\tilde{X}$ the linguistic term associated with a given ordinal $x \in \mathcal{O}$, and $[\tilde{X}]$ its center of gravity —for example, $\frac{m_1+m_2}{2}$ in (59) and m in (60)— it holds that:

$$\forall x, y \in \mathcal{O}: \quad [\tilde{X}] \leq [\tilde{Y}] \Leftrightarrow x \preceq y$$

Additionally, a complete coverage and a correct degree of overlapping should be ensured. It is a design decision —involving the existence of a continuous substrate and enough domain knowledge— which variables are amenable to be treated as linguistic and which as ordinal.

Note that the so-constructed fuzzy quantities $\tilde{x}$ and $\tilde{X}$ are respectful with the respective orderings in $\Gamma \in \mathbb{R}$ and $\mathcal{O}$. All these similarity measures are conveniently defined such that $s_{max} = 1$, so to allow a simpler and more general design of aggregation functions.

4.2 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, as follows. Consider 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 of 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 [49].

Consider now the collection of n_f extended fuzzy sets of the form $\hat{\mathcal{F}}_1, \ldots, \hat{\mathcal{F}}_{n_f}$, where $\hat{\mathcal{F}}_i = \mathcal{F}_i \cup \{\mathcal{X}\}, \mathcal{F}_i \subset \mathbb{F}_q(\Gamma_i), \Gamma_i \in [\mathbb{R}]$, and $1 \leq i \leq n_f$. Notice that $\mathbb{F}_n(\Gamma_i) \subset \mathbb{F}_q(\Gamma_i)$.

The respective Cartesian products can now be constructed:

$$\begin{array}{rcl} \hat{\mathcal{R}}^{n_r} & = & \hat{\mathcal{R}}_1 \times \ldots \times \hat{\mathcal{R}}_{n_r} \\ \hat{\mathcal{O}}^{n_o} & = & \hat{\mathcal{O}}_1 \times \ldots \times \hat{\mathcal{O}}_{n_o} \\ \hat{\mathcal{N}}^{n_n} & = & \hat{\mathcal{N}}_1 \times \ldots \times \hat{\mathcal{N}}_{n_n} \\ \hat{\mathcal{F}}^{n_f} & = & \hat{\mathcal{F}}_1 \times \ldots \times \hat{\mathcal{F}}_{n_f} \end{array} \tag{62}$$

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, the space formed by the Cartesian aggregation of the previous sets:

$$\hat{\mathcal{H}}^n \equiv \hat{\mathcal{R}}^{n_r} \times \hat{\mathcal{O}}^{n_o} \times \hat{\mathcal{N}}^{n_n} \times \hat{\mathcal{F}}^{n_f}$$

is such that $n = n_r + n_f + n_o + n_n$, with $\hat{\mathcal{R}}^0 = \hat{\mathcal{O}}^0 = \hat{\mathcal{N}}^0 = \hat{\mathcal{F}}^0 = \emptyset$ and $n > 0$. According to this definition, elements of $\hat{\mathcal{H}}^n$ are general tuples of n components, among which there may be real numbers, fuzzy sets (representing fuzzy numbers and p-quantities), ordinals, nominals and missing data. We will call such a structure an *heterogeneous space.*

Proposition 4.7 *The Definitions and Propositions in Section (3), referring to distance and similarity measures, apply also to an heterogeneous space $\hat{\mathcal{H}}^n$.*

Proof: in all these statements, there is no assumption about the structure of the space itself, provided basic distance and similarity functions are definable. The existence of aggregation operators is only conditioned to the fulfillment of their respective properties.

From Propositions (3.10) and (4.7), we derive the following Corollary:

Corollary 4.1 *Let Θ_s be an aggregation operator as defined in Proposition (3.7). Given a group of partial similarities $\vec{s} = \{s_1, \ldots, s_n\}$, where:*

$$s_i : \hat{\mathcal{H}}^i, \hat{\mathcal{H}}^i \to [0, s_{max}]$$

Then, the measure:

$$s : \hat{\mathcal{H}}^n, \hat{\mathcal{H}}^n \to [0, s_{max}]$$

defined as:

$$s(\vec{x}, \vec{y}) = \Theta_s(s_1(x_1, y_1), \ldots, s_n(x_n, y_n))$$

for $\vec{x}, \vec{y} \in \hat{\mathcal{H}}^n$, is a similarity measure in $\hat{\mathcal{H}}^n$, where $\hat{\mathcal{H}}^n$ is the Cartesian product of the $\hat{\mathcal{H}}^i, 1 \leq i \leq n$.

Analogously, to build an heterogeneous distance $d \in D(\hat{\mathcal{H}}^n)$ two basic steps are needed:

1. Define the partial measures $d_i \in D(\hat{\mathcal{H}}^i), 1 \leq i \leq n$;
2. Define a distance d by an aggregation operator $\Theta(\vec{d}; \vec{v})$ where $\vec{d} = \{d_1, \ldots, d_n\}$.

However, with such kinds of heterogeneity, uncertainty or missing information, only in some cases it will be possible to define a distance measure in the pattern space. The absence of a metric can in general be caused by:

- Presence of categorical variables (absence of an order);
- Presence of missing values (regardless of the data type);
- Uncertainty and imprecision in some variables.

Moreover, we have seen that in some occasions, although we *could* in principle define a metric (such as an Euclidean), it is preferable to devise a direct similarity measure incorporating domain theory.

5 A Framework for General Neuron Models

Definition 5.1 (h-neuron) *Given $\vec{x} \in \hat{\mathcal{H}}^n$, an heterogeneous neuron or h-neuron is a function of the form:*

$$F_i(\vec{x}) = \{h(\vec{x}, \vec{w}^{\,i}), \vec{w}^{\,i} \in \hat{\mathcal{H}}^n\} \tag{63}$$

with $h : \hat{\mathcal{H}}^n, \hat{\mathcal{H}}^n \to \hat{\mathcal{H}}$, where the domains $\hat{\mathcal{H}}^n$ are equal heterogeneous spaces of n components, and the codomain $\hat{\mathcal{H}}$ is a single heterogeneous space.

This definition accounts for general mappings from an (heterogeneous) input space to an (heterogeneous) output space. Therefore, h-neurons are classified according to the nature of their codomain (which must not be necessarily restricted to a subset of the reals). In the present study, a model with a codomain given by $\hat{\mathcal{H}} = \mathbb{R}$ is called of the *real kind*.

The reason to consider in a first place neuron models of the real kind is twofold. On the one hand, there is their natural coupling with other classical neuron models (i.e. accepting only real inputs), thus leading to hybrid networks in a straightforward way. On the other, in this work, the emphasis has been put on the definition of scalar similarity functions acting on real subintervals. None the less, the definition makes provision for neurons, either not working on similarity or computing similarity functions in other image sets $s : \hat{\mathcal{H}}^n, \hat{\mathcal{H}}^n \to \hat{\mathcal{H}}$ where in general the range of possible values of s is a subset $R_s \subset \hat{\mathcal{H}}$, and $\hat{\mathcal{H}}$ is one of the subspaces in $\hat{\mathcal{H}}^n$.

Definition 5.2 (S-neuron) *A similarity-based neuron or s-neuron (of the real kind) is an h-neuron computing device for which $h(\cdot, \cdot)$ is a similarity or pseudo-similarity in $\hat{\mathcal{H}}^n$ and $\hat{\mathcal{H}} = \Gamma_s \cup \{\mathcal{X}\}, \Gamma_s = [0, s_{max}] \in [\mathbb{R}]$.*

It is now straightforward to see that the neuron models defined in (2.3) and (13) are included in the above definition, by setting $\hat{\mathcal{H}}^n = \mathbb{R}^n$, $n = n_r$ (and thus $n_f = n_o = n_n = 0$, and no missing data at all) and making use of Propositions (3.17) and (3.19). The four types of similarity described in Section (3.5) are also included, since nothing is required about the precise form of the function $h(\cdot,\cdot)$.

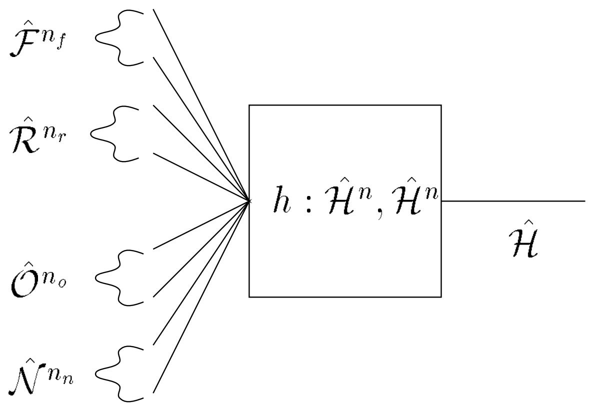


Figure 9: The h-neuron model.

Within this framework, as described in Section (3), all possible variations of *localized* RBF units of the kind described in (2.3) are included in similarity measures of type (a). It suffices to take real-valued and complete inputs, and a metric followed by a $\hat{s}$ function. In theory, there is no need for a RBF unit to have a localized response. Choices as $g(z) = z^2 lnz$ (thin-plate spline), $g(z) = \sqrt{z^2+c^2}, c \in \mathbb{R}$ (multiquadric), or even simpler functions as $g(z) = z^3$ (cubic) are in the list of valid radial basis functions for data interpolation [4], although they require a polynomial to be added to the general expression of the network (a linear term for the above functions). However, non-localized functions are scarcely used in RBF networks applications because –albeit they are still radially centered– their unbounded response departs from the clear conceptual view of setting local models that respond strongly to a small (input) region of attraction and gently (or rapidly) fade away outside this region. This view is in the basis of RBFs and any conceivable distance-based localized model fits nicely in the similarity framework.

Factorizable RBF networks (F-RBFNN), an idea introduced in [4], are specifically included as particular cases of type (b) similarity measures. These networks are conceptually seen as three layered ones. With respect to a standard RBF network, an extra layer is placed before the hidden layer. In this extra layer, there is a neuron per each input dimension for every neuron in the hidden layer. These neurons compute a one-dimensional distance between their (scalar) input and weight, followed by a localized function. Their outputs are then seen as "factors" and grouped by a hidden neuron. This grouping is a product operator. This scheme is intuitively appealing for Gaussian units working on Euclidean distance, since in this case the Gaussian can be factorized in individual terms as follows:

$$e^{-||\vec{x}-\vec{y}||^2} = e^{-\sum_{i=1}^{n}|x_i-y_i|^2} = \prod_{i=1}^{n} e^{-|x_i-y_i|^2} \tag{64}$$

As in our framework the partial similarity measures s_i can be seen as representing a single unidimensional unit specifically detecting similarity in one input dimension, equation (64) is cast as

$$\prod_{i=1}^{n} e^{-|x_i-y_i|^2} = \Theta_s(\vec{s}) \tag{65}$$

with $\vec{s} = \{s_1, \ldots, s_n\}$, $s_i = e^{-|x_i-y_i|^2}$ (thus $s_{max} = 1$ for all i) and Θ_s as in Proposition (3.11, 2.) with $m = v_i = 1$.

In general, single units need not be Gaussian nor their combination needs to be done in this way. Any valid set of unidimensional similarity measures –not necessarily distance-based– possibly designed in accordance with the nature of their respective inputs, can be grouped by means of a similarity aggregation operator.

Measures of type (c) include scalar product as a special case and, by simple extension, the sigmoidal unit defined in (13). As for measures of type (d), similarities from dissimilarities, they do not correspond or include any neuron model as used in ANNs. The situation makes sense whenever it is sensible to define a pseudo-distance, that is, a distance not fulfilling triangular inequality. The transformation functions $\hat{s}$ and $\breve{s}$ are valid and can be applied.

The flexibility of this framework carries with it several advantages:

1. There is no strict order between linear/non-linear functions, as long as the overall (similarity) measure is non-linear.

2. Domain knowledge can be put into work in the design of the model.

3. The functions $\breve{s}$ and $\hat{s}$ and the aggregation operators play analogous roles regardless of the specific similarity type.

Last but not least, Definition (5.2) explicitly states that the weight w_i associated to a given input line x_i is an element of the same space than x_i. This in turn implies that, whenever $\hat{\mathcal{H}}^n = \mathbb{R}^n$ and, more precisely, $\hat{\mathcal{H}}^i = \Gamma_i \in [\mathbb{R}]$, then *both* $x_i, w_i \in \Gamma_i$, thereby bounding the weight to be within the same limits than the input it is weighting, giving a strong basement to the assumption (43). Notice that, as a further consequence of the definition, the presence of missing values is allowed in the weights. This can be beneficial to a learning algorithm, since it adds for more flexibility by enabling the units to ignore some of their inputs.

Among the many different architectures, we consider in the first place hybrid feed-forward ones, consisting of a hidden layer of s-neurons and an output layer of linear p-neurons. Although, in essence, this architectures are not fully compliant with the ideas exposed in the Introduction[11], a simple parsimony principle leads to begin by choosing the simplest architectures, among those with enough theoretical computing capacity. The output p-neurons can be regarded as performing a linear combination of the similarities reported by the hidden layer. More elaborated architectures, where outputs are also computed by s-neurons, would constitute the immediate step. From a conceptual point of view, these units would be computing a similarity measure between vectors of similarities, and yielding a scalar quantity in account of the input vector fed to the network. A possibly interesting choice for the aggregation of such similarity vectors would be given by a normalized modulus (Proposition 3.11, 3.). In addition, networks with more than one hidden layer of s-neurons are also a clear choice and subject of study, albeit it is conjectured that one hidden layer should suffice for most applications, especially if the network is acting as an (heterogeneous) RBF network.

Definition 5.3 (HNN) *A feed-forward Heterogeneous Neural Network or HNN is a 1-FFNN[c] for which $c \geq 1$, where the F^l are s-neurons, $1 \leq l \leq c$, and the output units are either s-neurons or p-neurons.*

The general aspect of an HNN is depicted in Fig. (10) for $c = 1$. The computation performed by each hidden s-neuron is:

$$F_i(\vec{x}) = s(\vec{x}, \vec{w}^{\,i})$$

If the output unit is a p-neuron, thanks to Proposition (2.1) we have:

$$y(\vec{x}) = \mathcal{F}[h_1; F](\vec{x})$$

[11]Note that, though the inputs of the p-neurons are bounded, their weights are not.

In case it is a s-neuron, computing a similarity measure s_0:

$$y(\vec{x}) = s_0(\vec{w}, \vec{F}(\vec{x}))$$

where $\vec{w} = (w_1, \ldots, w_{h_1})$ and $\vec{F}(\vec{x}) = (F_1(\vec{x}), , \ldots, F_{h_1}(\vec{x}))$.

5.1 An example of a s-neuron

A basic but very useful heterogeneous measure can be devised by making use of a Gower-like similarity index [29]. This coefficient has its values in the real interval $[0, 1]$ and for any two objects $\vec{x_i}$, $\vec{x_j}$ given by tuples of cardinality n, is given by the expression

$$s_G(\vec{x_i}, \vec{x_j}) = \frac{\sum_{k=1}^{n} g_{ijk}\, \delta_{ijk}}{\sum_{k=1}^{n} \delta_{ijk}} \tag{66}$$

where:

- g_{ijk} is a similarity *score* for objects $\vec{x_i}$, $\vec{x_j}$ 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.
- δ_{ijk} is a binary function expressing whether the two objects are comparable or not according to their values w.r.t. variable k, as follows:

$$\delta_{ijk} = \begin{cases} 1 & \text{if } x_{ik} \neq \mathcal{X} \wedge x_{jk} \neq \mathcal{X} \\ 0 & \text{otherwise} \end{cases} \tag{67}$$

In particular, in Gower's original work, the partial similarity measures used are those defined in (48) for nominal and (51) for continuous variables. For ordinal, set, fuzzy and linguistic variables, their scores can be computed by using the measures in (49), (52), (58) and (61), respectively. The expression in (66) can be cast as an additive similarity aggregation operator with $s_{max} = 1$ and using Proposition (3.11, 3.) by taking $f(z) = z$ and a weighting vector $\vec{v} = (v_1, \ldots, v_n)$ defined as:

$$v_k = \begin{cases} \frac{1}{n'} & \text{if } x_{ik} \neq \mathcal{X} \wedge x_{jk} \neq \mathcal{X} \\ 0 & \text{otherwise} \end{cases} \tag{68}$$

where n' is the number of actually performed comparisons. Note how the treatment of missing values adopted respects the philosophy described in Section (3.4). To see this, let $\vec{s}^{\,1}$ be a vector of partial similarities. Suppose now that a further feature is added such that, *for a particular* two objects, this feature is missing for one of the two. That is, we have $\vec{s}^{\,2}$, which is $\vec{s}^{\,1}$ plus a $\mathcal{X}$. Then, $\Theta_s(\vec{s}^{\,1}) = \Theta_s(\vec{s}^{\,2})$. For example, let:

$$\vec{s}^{\,1} = \{0.3, \mathcal{X}, 0.1\}$$
$$\vec{s}^{\,2} = \{0.3, \mathcal{X}, 0.1, \mathcal{X}\}$$

This aggregation policy yields $\Theta_s(\vec{s}^{\,1}) = \Theta_s(\vec{s}^{\,2}) = 0.2$. The possibility that none of the partial similarities can be performed has to be handled with special care. Formally, given $\vec{x_i}, \vec{x_j}$, if it holds that:

$$\forall k : 1 \leq k \leq n : \delta_{ijk} = 0$$

the result of (66) is $\frac{0}{0}$. As in Gower's original work, the measure would remain undefined. This is unfeasible from the point of view of a neural network computation. There are at least two immediate solutions:

1. To set $s_G(\vec{x_i}, \vec{x_j}) = 0$, denoting that they are incomparable (corresponding to a *pessimistic* interpretation of similarity).
2. To set $s_G(\vec{x_i}, \vec{x_j}) = 1$, denoting that they are indistinguishable (corresponding to an *optimistic* interpretation of similarity). This option carries with it the redefinition of when two objects are equal.

In our framework, however, there is a third possibility, namely, being $\vec{s}^{\,\mathcal{X}} = \{\mathcal{X}, \ldots, \mathcal{X}\}$, define $\Theta_s(\vec{s}^{\mathcal{X}}) = \mathcal{X}$, expressing that a specific unit's computation of similarity is missing.

An added advantage stems from the fact that (66) is not affected by any linear normalization (see Appendix C) thus rendering this usual preprocessing step unnecessary. Furthermore, being a linear aggregation operator, a non-linear component can be added to in the form of a similarity keeping function $\breve{s}$ to form an s-neuron of the type defined in (5.2), as follows:

$$h_G(\vec{x}, \vec{w}^{\,i}) = \breve{s}(s_G(\vec{x}, \vec{w}^{\,i})) \qquad \vec{x}, \vec{w}^{\,i} \in \hat{\mathcal{H}}^n \tag{69}$$

The widespread logistic function can be used to work as a non-linear $\breve{s}$ by adapting it so as to map the real interval $[0, 1]$ on $(0, 1)$, by taking $\breve{s} = g_{11,0.5}$ in (10). None the less, other families of logistic activation functions can be especially designed to operate within the $[0, 1]$ interval [49], such as $\breve{s} = f(\cdot, k)$, where:

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

$$a(k) = \frac{-0.5 + \sqrt{0.5^2 + 4 * k}}{2} \tag{70}$$

being $a(k)$ an auxiliary function with $k > 0$ a real-valued parameter controlling the global curvature, usually set in the experiments to $k = 0.1$. This family of functions is displayed in Fig. (11).

Especially noteworthy is the following situation. If the heterogeneous space $\hat{\mathcal{H}}^n$ includes only elements of $\mathbb{R}, \mathcal{O}$ or $\mathcal{N}$, which are the similarities coming from distances, *and* there is no missing information, then the overall measure in (66) can be viewed as coming from a distance $d_G \in D(\mathcal{H}^n)$, as follows:

$$h_G(\vec{x}, \vec{w}^{\,i}) = \breve{s}(1 - d_G(\vec{x}, \vec{w}^{\,i})) \qquad \vec{x}, \vec{w}^{\,i} \in \hat{\mathcal{H}}^n \tag{71}$$

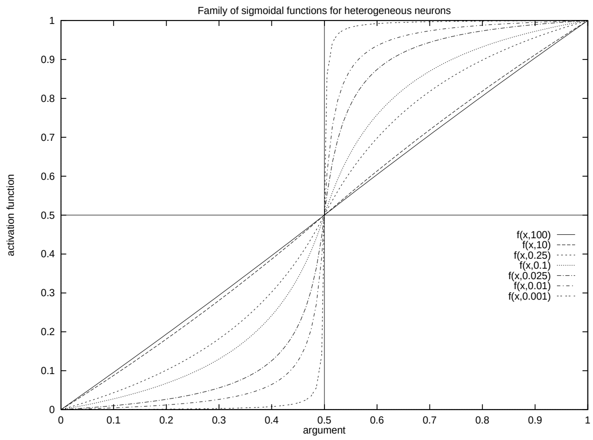


Figure 11: The family of sigmoidal functions $f(x, k)$, for different values of k.

The above expression fits the RBF unit scheme since it is a type (a) similarity measure of the form (34) with $\hat{s}(z) = 1 - z$ applied to an aggregated distance d_G, built as in (18) with $n' = n$, $q = 1$ and $v_i = 1$. Additionally, the expression can be further simplified by making use of Proposition (3.14) and collapsing the activation function into a single similarity transforming one:

$$\hat{s}'(z) = \breve{s}(1 - z)$$

where either $\breve{s} = g_{11,0.5}(\cdot)$, $\breve{s} = f(\cdot, k)$ or other suitable function. In any case, the neuron computation corresponds to a localized distance-based –the response of which is shown in Fig. (12)– thereby making (69) an instance of a model acting as a extended RBF (because it accepts ordinal data) in this particular situation, and in all cases behaving as a similarity-driven model. In general, though, in presence of fuzzy, nominal or missing information in the components of the heterogeneous input space, the function d_G in (71) cannot be guaranteed to be a distance in $D(\mathcal{H}^n)$ and the RBF philosophy is departed.

Additionally, it is noteworthy to observe how the sigmoidal-like functions are by no means tied to the MLPNN (or, in general, to be used as $\breve{s}$-functions). They can most conve-

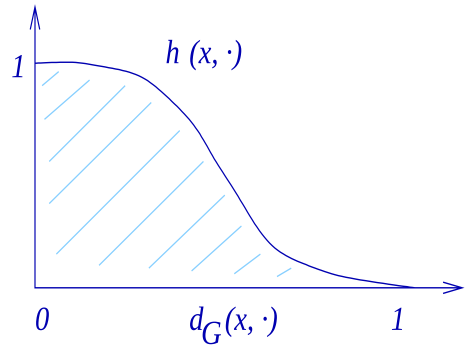


Figure 12: General response of the example s-neurons (for fixed weights).

niently be used as an $\hat{s}$-function by inverting its behaviour. For instance, by setting $a = 2$ in the Example $\hat{s}_4$ in page (23), the function $\sigma_0(z) = 2(1 - \sigma(z))$ is a similarity whenever z is a distance, with a response much like that in Fig. (12). This transformation is well-known although scarcely used in the ANN context. In case $z \in [0, 1]$, the relation is more compactly written as $\sigma_0(z) = 2\sigma(-z)$. Notice that in all these situations only the upper half of the functions is being used.

5.2 Classes of $\hat{s}$ functions

In this section, several classes of functions that can act as similarity transformers are presented, obtained by generalizing and extending the functions listed in page (23). Some of their instances are well known and have been proposed in the context of RBF networks as theoretically valid radial basis functions [4]. In the proposed framework, the requirement for inclusion in the list is the fulfillment of Definition (3.9). Without loss of generality, all the functions are set to yield $s_{max} = 1$. The following represent flexible classes of $\hat{s}$ functions:

$$\hat{s}_0(z) = (1-z^d)^\alpha,\ d \leq 1, \alpha \geq 1, z \in [0,1] \quad (72)$$

$$\hat{s}_1(z) = \frac{1}{1 + (az)^\alpha},\ \alpha > 0, a > 0, z \in [0, \infty) \quad (73)$$

$$\hat{s}_2(z) = \frac{1}{1 + e^{(az-b)^\alpha}},\ \alpha > 0, a > 0, z \in [0, \infty) \quad (74)$$

$$\hat{s}_3(z) = e^{-(az)^\alpha},\ \alpha \geq \frac{1}{2}, a > 0, z \in [0, \infty) \quad (75)$$

$$\hat{s}_4(z) = 1 - g_{11,0.5}(cz),\ c \geq 1, z \in [0,1] \quad (76)$$

The variety of such classes of functions is enhanced by the introduction of parameters to control their precise shape within a specific analytic expression. In practical terms, this variety translates in to how *sensitive* a neuron endowed with a given function will be w.r.t. its argument. Since this argument is a distance, the sensitivity can be thought of as a means to control how far apart (in heterogeneous distance) two entities (in this case, an input and its weight) should be to be considered as, for example, "half" similar (recall the similarity is always expressing a degree).

The first family of functions is designed so as to explicitly take this controlled sensitivity into account. The basic distance to similarity transformation $\hat{s}_0(z) = 1 - z$ being linear, this generalization achieves also the purpose of making it non-linear in a controlled way. First, let us show how (72) is a valid generalization of the standard transformation $t(z) = 1 - z$, assuming z is a normalized distance in $[0, 1]$.

Proposition 5.1 *Given z a normalized distance, $z \in [0, 1]$, the family of functions*

$$\hat{s}_0(z) = (1 - z^d)^\alpha, d \leq 1, \alpha \geq 1$$

is a similarity in $[0, 1]$.

Proof. First, since z is a distance, $z^d (d \leq 1)$ so is, by Lemma (A.4) in the Appendix. Second, $\hat{s}_0(z) = 1 - z$ is a valid similarity transforming function, one of those in the list at page (23), for $d_{max} = 1$. And third, $\breve{s}(z) = z^\alpha, \alpha \geq 1$ is clearly a valid similarity keeping function.

A logistic non-linearity $g_0 = g_{11,0.5}$ can be further applied, as a $\breve{s}$ function, like in Section (5.1). In (76), the same function is used in a parameterized way, this time as a $\hat{s}$ function. Note that this last class is a particular case of (74), by making use of the equality $1 - g_{\beta,\theta}(cz) = g_{\beta,\theta}(-cz)$ and setting $a = 11c, b = 6$ and $\alpha = 1$ in (74). The particular response of this class is the reason why it is included in the list; although its general appearance is obviously much like that of (75), its conception as a generalized and inverted logistic can be conveniently instantiated into more specific functions, for which the centered and symmetric behaviour over a diagonal axis is no longer needed. Instead, its utility stems from its smooth and monotonically decreasing response, controlled by parameter c.

The response of functions (72) to (76) is displayed in Figs. (13 to 17), where their diverse behaviour is plainly observed, reacting quite differently to distance values within the same input range. Therefore, it is of interest to establish a means to compare their relative response and a mechanism to determine which of them are more suited to be used as similarity transformers in practice.

5.3 Sensitivity relations

The idea behind the sensitivity of a $\hat{s}$ function can be informally stated as follows: given a similarity value $s_0 \in [0, s_{max}]$, what is the required distance value d_0 to yield s_0? Since $s_0 = \hat{s}(d_0)$, we have $d_0 = \hat{s}^{-1}(s_0) \in [0, \infty)$ (note that the inverse $\hat{s}^{-1}$ is guaranteed to exist by the properties of $\hat{s}$). The smaller d_0, the less sensible $\hat{s}$ is, since it takes a small distance value to reach a given similarity. Accumulating all this values over all $s_0 \in [0, s_{max}]$, and applying a proper normalization, we have a measure $\odot$ of sensitivity for a given $\hat{s}$ function.

The accumulation is nothing more than the integral:

$$\odot \hat{s} = \int_{[0,s_{max}]} \hat{s}^{-1}(z) dz$$

Assuming $\hat{s}^{-1}(z) \in [0, d_{max}]$ –which implies that $\hat{s}(d_{max} = 0)$, by Proposition (3.15)– the above integral is guaranteed to exist (that is, to be finite) and $\odot \hat{s} \in [0, s_{max} d_{max}]$; therefore, the sensitivity can be normalized. In addition, by making use of the following equality:

$$\int_{[0,s_{max}]} \hat{s}^{-1}(z) dz = \int_{[0,d_{max}]} \hat{s}(z) dz \qquad (77)$$

the inverse can be avoided. Denoting $\hat{S}$ the set of all $\hat{s}$ functions satisfying Definition (3.9), we end by defining the *sensitivity* of a function:

$$\hat{s} : [0, d_{max}] \to [0, s_{max}]$$

as a function

$$\odot : \hat{S} \to [0, 1]$$

such that

$$\odot \hat{s} = \frac{\int_{[0,d_{max}]} \hat{s}(z) dz}{s_{max} d_{max}} \qquad (78)$$

This measure gives a value closer to 1 the more sensible a function is; in other words, the easier is for a distance to elicit a given similarity value, accumulated over all the possible distance values. It is non-negative since the integrated functions are; being all the $\hat{s} \in \hat{S}$ monotonic and continuous, they are Riemann-integrable in $[0, d_{max}]$ [40]. Additionally, (78) induces an ordering relation $\sqsubseteq$ on $\hat{S}$, defined as:

$$f_1 \sqsubseteq f_2 \Longleftrightarrow \odot f_1 \leq \odot f_2, \qquad f_1, f_2 \in \hat{S} \quad (79)$$

expressing that f_1 is *less* sensible than f_2. It is straightforward to see that $\sqsubseteq$ is a preorder (a reflexive and transitive relation) and hence $(\hat{S}, \sqsubseteq)$ is a *preordered* set. The relation is neither symmetric (obviously) nor antisymmetric. This is illustrated in Fig. (18), in which it is assumed that $\odot f_1 = \odot f_2$.

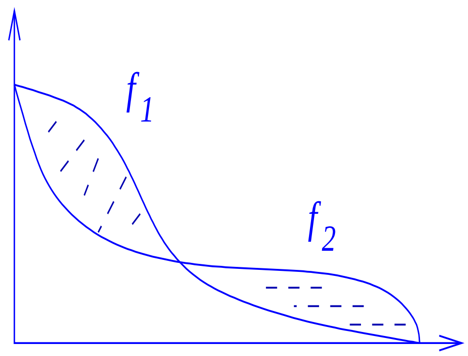


Figure 18: An example of non-antisymmetry. Assuming both marked areas are equal, it follows that $f_1 \sqsubseteq f_2 \wedge f_2 \sqsubseteq f_1$ but $f_1 \neq f_2$.

Mathematically, the comparison between different $\hat{s}$ makes sense because, in general:

$$\forall \hat{s} \in \hat{S}[a,b],\ \exists \xi(\hat{s}) / \int_{[a,b]} \hat{s} = \xi(\hat{s})(b-a) \quad (80)$$

In our case, $[a,b] = [0, d_{max}]$ and is constant for all the functions so that, being (77) in $[0, s_{max} d_{max}]$, the number:

$$\xi(\hat{s}) = \frac{1}{d_{max}} \int_{[0,d_{max}]} \hat{s}$$

represents the *fraction* (w.r.t. s_{max}) of area filled by $\hat{s}$ in $[0, d_{max}]$, depicted in Fig. (19). Dividing by s_{max}, we get the normalized measure (78).

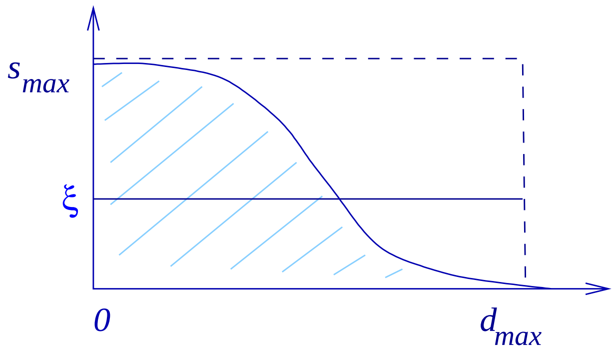


Figure 19: Sensitivity as a similarity ratio.

5.4 Sensitivity in practice

To clarify the relationship of different instances of the proposed families of $\hat{s}$ functions and to assess their usefulness as parts of recognition devices, a numerical experiment is performed with a twofold purpose. On the one hand, to check whether sensitivity is an important quantity and to what degree it exerts an influence on neuron performance. On the other, to ascertain what is the best range of values for the sensitivity. Too sensitive a neuron would make it useless as a recognizer because of the loss in discrimination ability, while too unsensitive would translate in a very specialized neuron model, one yielding high outputs only for a specific pattern (its weight vector) and vigorously decaying moving away from it. This has to do in general with the *degree of overlap* between neurons, a topic already found in RBF networks [39]. Furthermore, Notice that the application of a $\hat{s}$ function is not limited to the case in which the neuron as a whole is computing a distance –as in type (a) similarities–, but can most conveniently be applied to individual neuron input lines –specifically to those that are distance-based– as in type (b) similarities. The general form of the function computed by the neuron allows the aggregation of partial similarities with no assumption to their distance-based character. Therefore, in those situations where the similarity measure is not completely of either type (a) or (b) the overall sensitivity of a neuron has to be assessed in a different way.

To this end, we define the β-sensitivity of a s-neuron i with weight vector $\vec{w}^i$, computing a similarity measure $s : K, K \to [0,1]$ over a support set $K \subset \hat{\mathcal{H}}^n$ as:

$$\beta_i(K, s) = E\{s(\vec{x}, \vec{w}^i)\}_{\vec{x} \in K}, \qquad \vec{w}^i \in K \tag{81}$$

where $E\{\cdot\}$ denotes the statistical expectation operator. One way to evaluate the sensitivity of s is by assessing its initial response in K. This can be achieved by computing the β-sensitivity as:

$$E\{s(\vec{x}, E\{\vec{w}_0^i\})\} \tag{82}$$

where $\vec{w}_0^i$ stands for the initial weight vector of neuron i in a training process. Expanding (81) we get, for $(\vec{x}, \vec{w}_0^i) \in K \times K \subset \hat{\mathcal{H}}^{2n}$:

$$\int_{K \times K} s(\vec{x}, E\{\vec{w}_0^i\}) p(\vec{x}, \vec{w}_0^i) \; d\vec{x} \; d\vec{w}_0^i \tag{83}$$

The joint distribution can be decomposed in two, for the two variables are independent:

$$\int_{K \times K} s(\vec{x}, E\{\vec{w}_0^i\}) p_x(\vec{x}) p_w(\vec{w}_0^i) \; d\vec{x} \; d\vec{w}_0^i \tag{84}$$

Let us denote by $D[X, p, \vartheta](\vec{x})$ a set of ϑ samples of $\vec{x}$ drawn from a set X with probability density function (pdf) $p(\vec{x})$. The pdf p_w for the initial weights is known albeit to be determined; we can generally assume it to be uniform though it is not necessary. In any case, we have a (finite) set $W_0^i = D[K, p_w, \vartheta_w](\vec{w}_0^i)$ of available samples at the outset of the training process. The pdf p_x from which the input patterns are drawn is not known. Instead, in general all we have is a set of samples $K_0 = D[K, p_x, \vartheta_x](\vec{x})$. Therefore, the integral (84) has to be estimated by evaluating it on the (finite) set $K_0 \times W_0^i$ so that:

$$\tilde{\beta}_i(K, s) = \frac{1}{\vartheta_x \vartheta_w} \sum_{\vec{x} \in K_0} \sum_{\vec{w}_0^i \in W_0^i} s(\vec{x}, \vec{w}_0^i) \tag{85}$$

yields an estimate of the true $\beta_i(K, s)$ in (81), where p_w is taken to be uniform. In practice, the measure (85) is likely to be more accurate the bigger the two sets of samples. As for W_0^i, we have in principle an unlimited (in the sense that it can be as big as needed) set of initial weights. With regard to K_0, having a training set as large and, most important, representative of the underlying distribution as possible is a key condition for acceptable performance on novel data. In addition, considering a network with a layer of h_1 hidden neurons allows for a supplementary source of independent estimates (at least initially), which can in turn be averaged over. The final quantity, denoted *mean initial similarity* is computed as:

$$\tilde{\beta}(K, s) = < \beta_i(K, s) >_i, \; 1 \le i \le h_1 \tag{86}$$

The value yielded by (86) gives an idea of how initially sensitive the network is to a precise data set. The question arises: what is the most convenient value for it? There surely is no value generally best, although, in practice, lower and upper bounds are likely to exist. It also depends on the way partial measures (be distances or similarities) are aggregated, and on the way these are computed. Therefore, the answer traces back to the sensitivity of the $\hat{s}$ functions as defined in (78). Since s_{max} has been set to 1, we could expect $\tilde{\beta}(K, s)$ to be about 0.5, signaling a middle initial average similarity. We hypothesize that this will certainly not be the case in general, for reasons that will soon be introduced. It will indeed be the case for the standard p-neurons as defined in Definition (13) and working on pseudo-similarities –see Proposition (3.19). As a matter of fact, it is general advice to set the initial weights of these neurons so to elicit an activation about 0.5 (that is, in the mid part of the sigmoid linearity) so that the learning process moves the weights towards saturation, thus committing the neuron to the extreme bounds of the non-linearity [10].

However, for general neuron models working on similarity, especially those using a combination of *different* partial measures, either distance-based or direct, for each input variable, the answer is not so obvious. Looking at it from the point of view of a similarity leads to think that 0.5 over 1.0 corresponds to a high degree of overall similarity. It means that, by guessing initially blind, untrained weight vectors, the expected similarity (that is, the degree of recognition) is yet half the maximum possible. It it thus reasonable to presume that a lower (possibly much lower, depending on the precise model) initial value would be more tenable.

6 Preliminary Experimental Comparison between Similarity Measures

6.1 Experiment description

To study the validity of the general approach and, at the same time, to preliminary assess how well each of the proposed neuron models (acting as basis functions) is able to approximate a sampled function, they are experimentally compared on a two-dimensional benchmark function [38], the non-linear system identification problem:

$$y(k) = y_1(k-1) + y_2(k-1)$$

where

$$\begin{aligned} y_1(k) &= 2.5y(k)sin[\pi e^{-u^2(k)-y^2(k)}] \\ y_2(k) &= u(k)[1+u^2(k)] \end{aligned} \qquad (87)$$

The output $y(k)$ depends on the previous input $u(k-1)$ and on the previous output $y(k-1)$. A learning set of $\vartheta_x = 100$ input/output pairs is constructed by setting $y(0) = 0$ and randomly exciting the system using a signal $u(k)$ uniformly drawn from $[-2.0, 2.0]$.

In this task there is no data heterogeneity –the two variables are continuous– and there is no missing information. It has been chosen as a preliminary case study precisely because of that: the primary interest being to compare the relative usefulness of several instances of the parameterized functions (72) to (76) as bricks for constructing basic distance-based similarity models on continuous variables, as well as to have them compared to the more classical neural models. A fuzzy model is also investigated. The following are the explored models:

1. The standard scalar-product neuron ($SclProd$) plus a bias, and the measure defined in (44) ($SclProd2$), both using a logistic $g_{1,0}$.

2. Two RBF models, using Euclidean distance as aggregation, and a Gaussian ($\hat{s}_3$ for $\alpha = 2$) as activation function. The first model, $StdRBF$ has a different variance (to be learned) for every hidden unit. The second, $NormRBF$, computes a normalized distance –that in Proposition (3.8, $q = 2$)– followed by a Gaussian adapted to an argument in $[0, 1]$, by setting $a = 6$.

3. A representative set of $\hat{s}$ functions –displayed in Table (3)– chosen from (72) to (76), applied to a distance computation, as explained below.

4. A fuzzy model, obtained by considering the original crisp data as triangular fuzzy numbers.

The partial similarity measures between the two variables in point 3. above are computed using a distance-based similarity function, obtained by application of the different $\hat{s}$ to the basic distance (50) on continuous variables. Notice that S0_0 –the first obtained function– in Table (3) corresponds to (51). They are all aggregated using a simple arithmetic mean:

$$s(\vec{x}, \vec{w}^{\,i}) = g_{11,0.5}\left(\frac{\hat{s}(d(x_1, w_1^i)) + \hat{s}(d(x_2, w_2^i))}{2}\right) \tag{88}$$

thus being type (b) similarities, whilst the two RBF models are examples of type (a). Note how the similarity transforming function $\hat{s}_3$ is being used for both types, though with a different parameter set. Also noteworthy is the fact that partial non-linear measures are linearly aggregated to form a non-linear similarity so that, in principle, there is no need to apply a further non-linearity (in the form of a similarity keeping function). The behaviour of (88) is illustrated in Fig. (20) for several choices of $\hat{s}_0$. All of these variations are to be tested experimentally.

The fuzzy model uses (58) for the partial similarities, the same aggregation and $\breve{s} = g_{11,0.5}$ as above. The imprecision is roughly estimated as follows. Data is generated from (87) by rounding to four significant digits (three decimal), with the application of equation (87) magnifying the errors up to a factor of 10. Therefore, symmetric triangular fuzzy numbers $\tilde{x}$ centered at the original value x are associated to x as $\mu_{\tilde{x}} = \mu_{[x,\alpha(x)]}$, $\alpha(x) = (\alpha_1, \alpha_2) = (kx, kx)$, $k = 5 \cdot 10^{-3}$.

6.2 Experimental setup

All of the tested models use exactly the same experimental environment, in which everything is kept constant except the neuron model itself. The two input variables are normalized to lie in $[0, 1]$. This is not needed by the heterogeneous neurons because they compute a normalized measure, but is beneficial for the standard models. The output is not normalized. A fixed architecture composed of $h_1 = 8$ neurons arranged in a hidden layer plus an output linear neuron is used. In general, the number of free parameters for the standard models in 1. and 2. is given by:

$$(n+1)h_1 + h_1 + 1 = h_1(n+2) + 1 = 33 \tag{89}$$

where $n = 2$ is the number of inputs. For the heterogeneous neurons in 3. it is given by:

$$nh_1 + h_1 + 1 = h_1(n+1) + 1 = 25 \tag{90}$$

while for the fuzzy model is:

$$2nh_1 + h_1 + 1 = h_1(2n+1) + 1 = 41 \tag{91}$$

The weights (including biases and standard deviations) for the models in 1. and 2. are let to vary in $[-10, 10]$, a sufficiently wide range given the normalization chosen. The same interval is used for the hidden-to-output weights in all cases.

The training procedure used is a *Breeder Genetic Algorithm* (BGA) [41] an Evolutionary Algorithm in mid position between Genetic Algorithms (GAs) [42] and Evolution Strategies[12] (ES) [43]. While in GA selection is stochastic and meant to mimic —to some degree— Darwinian evolution, BGA selection is named *truncation* selection, a deterministic procedure driven by a *breeding* mechanism, an

[12]Notice that the training method used is not an studied aspect here. It is implicitly assumed that different procedures are not to alter the *relative* behaviour of the models.

Function Index	Parameters	Expression
S0_0	$\hat{s}_0$ for d=1, α=1	$1-z$
S0_1	$\hat{s}_0$ for d=1, α=2	$(1-z)^2$
S0_2	$\hat{s}_0$ for d=1, α=4	$(1-z)^4$
S0_3	$\hat{s}_0$ for d=0.5, α=1	$1-\sqrt{z}$
S0_4	$\hat{s}_0$ for d=0.5, α=2	$(1-\sqrt{z})^2$
S0_5	$\hat{s}_0$ for d=0.5, α=4	$(1-\sqrt{z})^4$
S0_6	$\hat{s}_0$ for d=0.25, α=1	$1-\sqrt[4]{z}$
S0_7	$\hat{s}_0$ for d=0.25, α=2	$(1-\sqrt[4]{z})^2$
S0_8	$\hat{s}_0$ for d=0.25, α=4	$(1-\sqrt[4]{z})^4$
S1_0	$\hat{s}_1^*$ for a=1, α=0.25	$\frac{2}{1+\sqrt[4]{z}}-1$
S1_1	$\hat{s}_1^*$ for a=1, α=0.5	$\frac{2}{1+\sqrt{z}}-1$
S1_2	$\hat{s}_1^*$ for a=1, α=1.0	$\frac{2}{1+z}-1$
S1_3	$\hat{s}_1^*$ for a=1, α=2.0	$\frac{2}{1+z^2}-1$
S3_0	$\hat{s}_3$ for a=6, α=0.5	$exp\{-6\sqrt{z}\}$
S3_1	$\hat{s}_3$ for a=6, α=1.0	$exp\{-6z\}$
S3_2	$\hat{s}_3$ for a=6, α=2.0	$exp\{-6z^2\}$
S4_0	$\hat{s}_4$ for c=1.0	$1-g_{11,0.5}(1.0z)$
S4_1	$\hat{s}_4$ for c=1.5	$1-g_{11,0.5}(1.5z)$
S4_2	$\hat{s}_4$ for c=2.0	$1-g_{11,0.5}(2.0z)$
S4_3	$\hat{s}_4$ for c=2.5	$1-g_{11,0.5}(2.5z)$

Table 3: The different $\hat{s}(z)$ used in the experiment for partial similarity computations on continuous variables. The symbol $(*)$ denotes the adaptation to an argument $z \in [0,1]$, so that $\hat{s}_1^*(z) = 2\hat{s}_1(z) - 1$. In all cases, $s_{max} = 1$.

artificial selection method in which only the best individuals —usually a fixed percentage τ of total population size μ— are selected and enter the gene pool to be recombined and mutated, as the basis to form a new generation. A detailed description can be found in [45].

The BGA is used here with the following parameters: $\mu = 60, \tau = 25$, EIR recombination with $\delta = 0.45$, and continuous mutation with $\rho = 0.5, k = 8$. This setting has been proven very useful in the context of ANN optimization [44, 46]. A number of `NRuns`=10 independent runs are performed, each of which is allowed 10,000 error function evaluations, a moderate amount of computation for an ANN optimization task (a total of $\lfloor 10,000/60 \rfloor = 166$ generations are to be carried out).

This means that $\vartheta_w = h_1 \mu \texttt{NRuns} = 4,800$ is the sample size of initial weights as they are generated to form the various initial populations of individuals. For each tested model, the *mean square error* is reported (as MSE) plus/minus the normalized standard deviation in the usual form. The best MSE found is also reported. The last column is the mean initial similarity, a measure of network sensitivity, computed as in (86). The results of the training process are displayed in Tables (4) to (7) for the different models explored. The following remarks are in order:

1. The most interesting family is the S0, be-

Function	Mean MSE $\pm\ \sigma/\sqrt{n}$	Min MSE	$\tilde{\beta}(K,s)$
S0_0	1.228 $\pm$ 0.045	1.010	0.795
S0_1	0.752 $\pm$ 0.053	0.521	0.541
S0_2	0.371 $\pm$ 0.037	0.223	0.289
S0_3	0.448 $\pm$ 0.041	0.328	0.475
S0_4	0.356 $\pm$ 0.033	0.266	0.176
S0_5	0.840 $\pm$ 0.074	0.456	0.053
S0_6	0.273 $\pm$ 0.015	0.211	0.172
S0_7	1.312 $\pm$ 0.060	0.959	0.040
S0_8	5.315 $\pm$ 0.176	4.333	0.008

Table 4: Performance of the heterogeneous models using $\hat{s}_0$.

Function	Mean MSE $\pm\ \sigma/\sqrt{n}$	Min MSE	$\tilde{\beta}(K,s)$
S0_0	0.745 $\pm$ 0.050	0.633	0.692
S0_1	0.500 $\pm$ 0.029	0.350	0.524
S0_2	0.235 $\pm$ 0.017	0.133	0.352
S0_3	0.362 $\pm$ 0.044	0.176	0.487
S0_4	0.258 $\pm$ 0.045	0.151	0.280
S0_5	0.309 $\pm$ 0.023	0.173	0.124
S0_6	0.220 $\pm$ 0.015	0.145	0.302
S0_7	0.324 $\pm$ 0.053	0.149	0.117
S0_8	3.050 $\pm$ 0.090	2.704	0.028

Table 5: Same as Table (4), without using the $\breve{s} = g_{11,0.5}$ non-linearity.

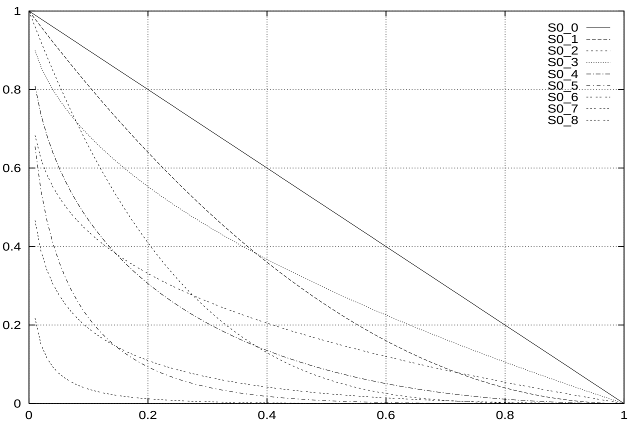


Figure 21: The functions named S0_0 to S0_8.

cause of its richer set of functions, displayed in Fig. (21). As it can be derived, the parameters d, α are in control of the overall sensitivity of the model. Obviously, for increasing α or decreasing d, high similarities are obtained only for really similar vectors, so that the neuron becomes less sensitive. More precisely, the $\odot$ sensitivities for S0_0 to S0_8 are, respectively,

$$\{\frac{1}{2},\frac{1}{3},\frac{1}{5},\frac{1}{3},\frac{1}{6},\frac{1}{15},\frac{1}{5},\frac{1}{15},\frac{1}{70}\}$$

Note how $\odot$S0_0=1/2 is half the maximum (perhaps too high) and $\odot$S0_8=1/70 $\approx$ 0.014 is a small value, leading to an almost unsensitive neuron. This is reflexed in Table (4), where there is a relation between the factor $\tilde{\beta}(K,s)$ and the obtained performance, also related to the above $\odot$ sensitivities. The networks using these two extreme models S0_0, S0_8

Function	Mean MSE $\pm \sigma/\sqrt{n}$	Min MSE	$\tilde{\beta}(K,s)$
S1_0	0.491 $\pm$ 0.050	0.311	0.072
S1_1	0.313 $\pm$ 0.042	0.143	0.245
S1_2	0.768 $\pm$ 0.057	0.534	0.613
S1_3	1.110 $\pm$ 0.022	0.954	0.882
S3_0	1.676 $\pm$ 0.095	1.276	0.036
S3_1	0.470 $\pm$ 0.088	0.173	0.212
S3_2	1.397 $\pm$ 0.134	0.751	0.630
S4_0	0.597 $\pm$ 0.025	0.444	0.838
S4_1	1.514 $\pm$ 0.083	1.207	0.607
S4_2	0.920 $\pm$ 0.076	0.596	0.450
S4_3	0.895 $\pm$ 0.109	0.504	0.355

Table 6: Performance of the heterogeneous models using $\hat{s}_1$, $\hat{s}_3$ and $\hat{s}_4$.

achieve the worst results (much worse for S0_8) for different reasons, displaying the highest (0.795) and lowest (0.008) $\tilde{\beta}$, respectively. For this family, the relations are the following:

$$\text{S0\_8} \sqsubseteq \begin{matrix}\text{S0\_7}\\ \text{S0\_5}\end{matrix} \sqsubseteq \text{S0\_4} \sqsubseteq \begin{matrix}\text{S0\_2}\\ \text{S0\_6}\end{matrix} \sqsubseteq \begin{matrix}\text{S0\_1}\\ \text{S0\_3}\end{matrix} \sqsubseteq \text{S0\_0} \tag{92}$$

This translates, on the one hand, in the mean initial similarity $\tilde{\beta}$, higher for higher $\odot$ sensitivities, as expected. On the other, in performance, which is markedly better in the middle of the relation scale (92), for functions S0_4, S0_2 and S0_6, corresponding to a $\tilde{\beta}$ in the range $[0.176 - 0.289]$. Notice also how the sensitivity of these three functions is very similar (1/6, 1/5, 1/5) and different from the rest. The standard deviation of the errors reported for these functions are also the lowest, signaling less variability, as is the best result found (Min MSE).

2. In Table (5), the same experiment is repeated, this time not using the non-linear logistic as activation function. It can be seen that the results have generally improved those in the previous experiment, the possible reason being that the non-linearities in the proposed measures for partial similarities are enough to capture the relation (87). Notoriously, S0_0 has also improved, although it is now a linear model. The $\tilde{\beta}$ estimations have suffered a slight increase, due to the absence of the logistic, which tended to lower similarity values under 0.5 and raise those above 0.5. The results are in general accordance with Table (4), being again S0_2, S0_4 and S0_6 the best performing models.

3. Results for the other $\hat{s}$ functions tested are shown in Table (6), related as follows (inter-family relations not shown):

$$\text{S1\_0} \sqsubseteq \text{S1\_1} \sqsubseteq \text{S1\_2} \sqsubseteq \text{S1\_3} \tag{93}$$

$$\text{S3\_0} \sqsubseteq \text{S3\_1} \sqsubseteq \text{S3\_2} \tag{94}$$

$$\text{S4\_3} \sqsubseteq \text{S4\_2} \sqsubseteq \text{S4\_1} \sqsubseteq \text{S4\_0} \tag{95}$$

The only acceptable performance (compared to the already found results) is that of S1_1 (the gaussian function) yielding a network with estimated $\tilde{\beta} = 0.245$, inside the previous range. As for the S3 family, only S3_1 shows a fair performance

Function	Mean MSE $\pm \sigma/\sqrt{n}$	Min MSE	$\tilde{\beta}(K,s)$
SclProd	0.549 $\pm$ 0.055	0.314	0.507
SclProd2	1.936 $\pm$ 0.015	1.826	0.135
StdRBF	1.659 $\pm$ 0.035	1.496	0.228
NormRBF	1.974 $\pm$ 0.052	1.628	0.522
Fuzzy	0.482 $\pm$ 0.041	0.252	0.064

Table 7: Performance of the two scalar-product models, the two RBF and the fuzzy model.

($\tilde{\beta} = 0.212$), while the logistic itself (S4 family) does not seem to be appropriate, possibly due to its characteristic S-shape.

4. The classical models are displayed in Table (7). The first remarkable point is the poor results achieved by the two RBF models, possibly caused by an insufficient number of basis functions, although factors as input normalization, different standard deviation per unit, a representative training set and the presence of only two variables are generally beneficial. Incidentally, notice how performance is comparable to that of S3_2.

5. The mean result for the scalar-product (0.549) can be taken as a reference for the similarity-driven models. For those in Table (4), all the models from S0_2 to S0_6 (except S0_5) yield a better performance. For those in Table (5), all the models from S0_1 to S0_7 —that is, all but the two extreme ones— outperform it, in various degrees. The standard deviation of the errors and the Min MSE are also lower. Notice how the estimated $\tilde{\beta} = 0.507$ for $SclProd$ is close to the expected value 0.500. Interestingly, the nearest model in performance (S0_1) yields a similar $\tilde{\beta} = 0.524$.

6. The measure $SclProd2$ reports also poor results, though in the line of those for the RBF. We conjecture that its estimated $\tilde{\beta} = 0.135$ is too low for a scalar-product, possibly caused by the input distribution. More work should be devoted to study this measure by controlling its sensitivity.

7. Regarding the fuzzy model, the first notable point is its very low sensitivity ($\tilde{\beta} = 0.064$), due to the way the partial similarities are computed. The fairly good obtained results (Mean MSE=0.482) indicate that explicitly considering uncertainty in a network leads to valid models enjoying an added flexibility –because they work on a superset of the reals. This compensates for the increase in the number of free parameters, less likely to be properly constrained by a limited size data set [17].

Perhaps the most noteworthy conclusion is the relation between model sensitivity, estimated $\tilde{\beta}(K,s)$ and network performance. In particular, sensitivity has been shown to have a deep impact on performance. In a working model, its value is to depend on two factors: the training set K_0 and the actual measure s being used. Holding the first factor constant, varying the shape of s reflects in the resulting performance of the system, and this is in turn related to the $\tilde{\beta}(K,s)$ estimation. In the presented experiment, the overall results are only indicative because of the simplicity of this preliminary study –only two variables (both of them significant), no missing values and no complex interactions present– but they are in general accordance with the discussion presented.

7 Concluding Remarks

One of the main contributions of the approach presented in this work consists in providing a conceptual framework for the study of neuron models as similarity computing devices. The intention is not to embody all the existing neuron models –a difficult topic about which some excellent material exists [30]. Rather, it is to provide a precise though comprehensive framework that unifies the most common models under the general cover of a similarity measure, thereby exploiting the various advantages stemming from this. We use the term **Heterogeneous Neural Network** for the broad class of architectures making use of neuron models derived from the framework and show the relation between neuron models of the kind used in MLP and RBF networks. The rationale behind the approach is that, by replacing, whenever convenient, the traditional scalar-product by an *explicit*, *normalized* and *heterogeneous* similarity measure, it is expected that the resulting model is to capture more satisfactorily the underlying process (in terms of quality of the approximation, generalization ability or readability of the solution found).

Perhaps a more important prospect is that it offers the possibility to develop *new* models, eventually taking into account data heterogeneity, a further extension that fits nicely in the framework. For example, standard (localized) RBF models are seen as particular cases whenever all the partial similarity computations are distance-based and continuous, using a specific aggregation measure and in the absence of missing values. The scheme stresses also the fact that certain activation functions are by no means tied to specific neuron models. Instead, what counts is the properties they fulfill and its capacity to adapt to different roles; in this sense, the sigmoidal family of functions is a flexible means to do so.

In general, the similarity relation is to be data dependent, and possibly problem dependent, since the same data could be treated differently for different tasks. In any case, it is ensured that any derived instance is to behave as a (soft) pattern recognizer and that similar patterns for the input units are to elicit similar responses in the hidden units. This may add an extra insight in the processing carried out in an artificial neuron (especially in the hidden ones) because of the known and controlled response characteristic and the possibility to add prior knowledge in its design. The $\hat{s}$ functions proposed, being parameterized families, allow to have a control on the exact shape of the neural response, leading to the idea of *sensitivity*. In this vein, the specific analysis of the space spanned by the h_1 hidden neurons as $\subset [0, s_{max}]^{h_1}$ and the way it is covered is to give a deeper insight in the overall process and opens a possibility for further refinement.

All this suggests that there are useful alternatives to classical neuron models. The task of selecting the most appropriate is however an open question. A given measure may maximize the accuracy of a learner in an experiment but perform poorly as compared to others in another one. This in turn relates to the number of examples needed to represent a specific function, bringing to the conclusion that there is no "universal" similarity measure that efficiently represents any target function [48].

Clearly, applications on real-world size problems as well as thorough studies in ANN benchmark data (as [32, 33]) are needed to assess the validity experimentally. These problems and benchmarks should specially include rich forms of data, as discrete, nominal or linguistic variables, and missing information, traditionally resistant to classical models.

Acknowledgements. This work is supported by the Spanish CICYT grant TAP99-0747.

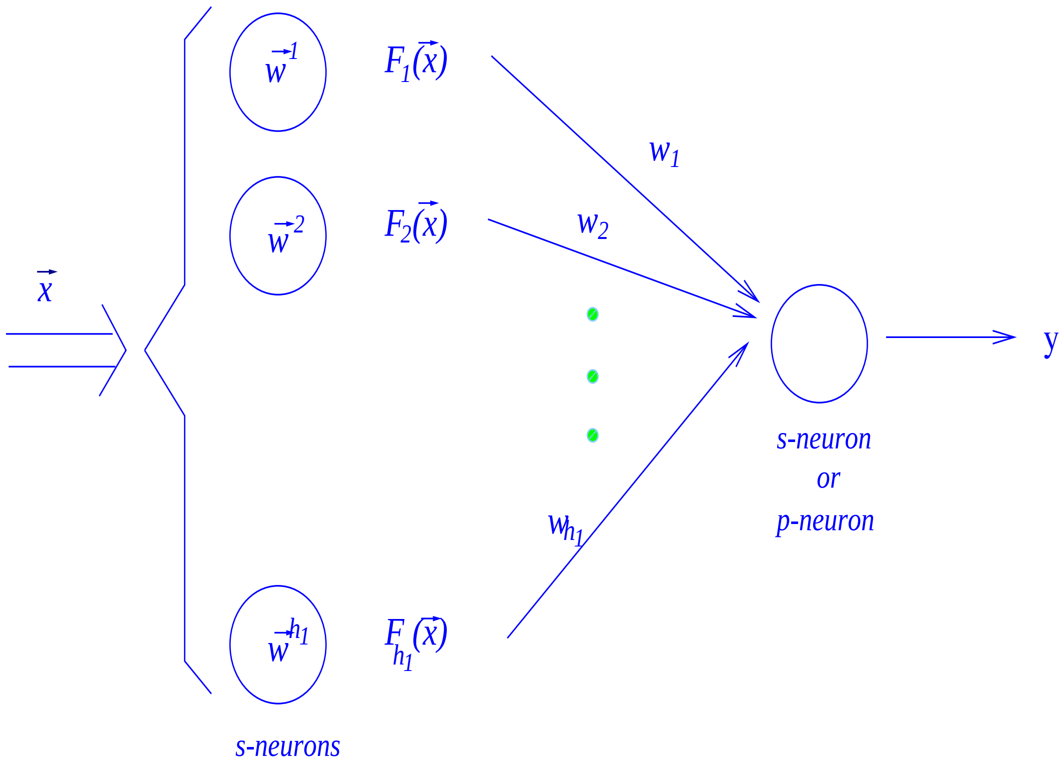
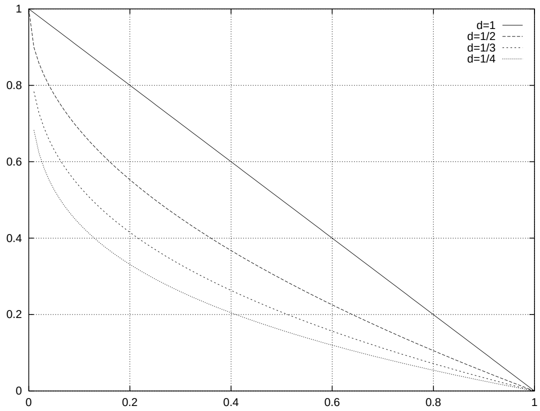
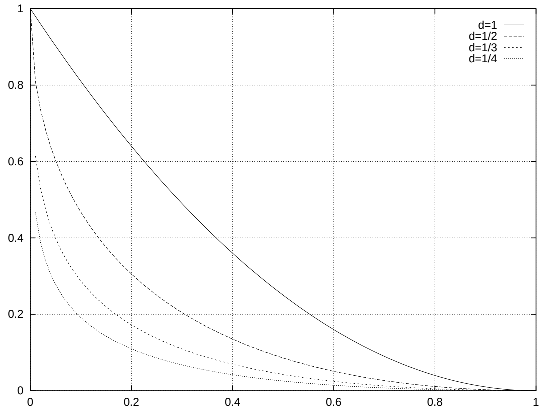


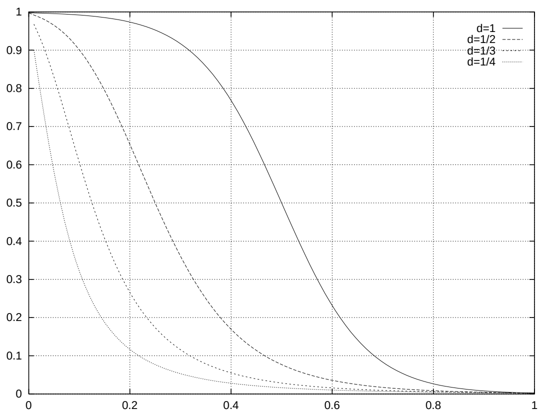
Figure 10: A one-hidden-layer HNN with h_1 hidden s-neurons and a single output unit, either a s-neuron or a p-neuron, where $F_j(\vec{x}) = s(\vec{x}, \vec{w}^j), 1 \leq j \leq h_1$.



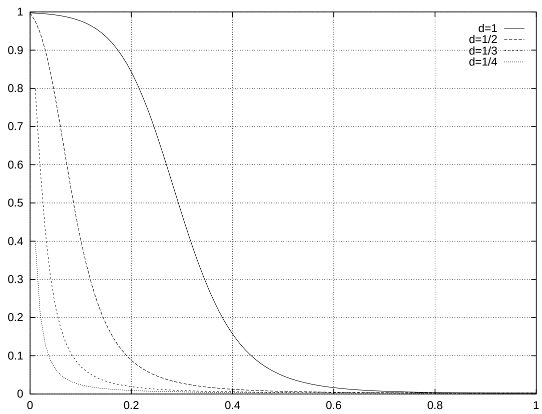
(a) Varying d for $\alpha = 1$



(b) Varying d for $\alpha = 2$

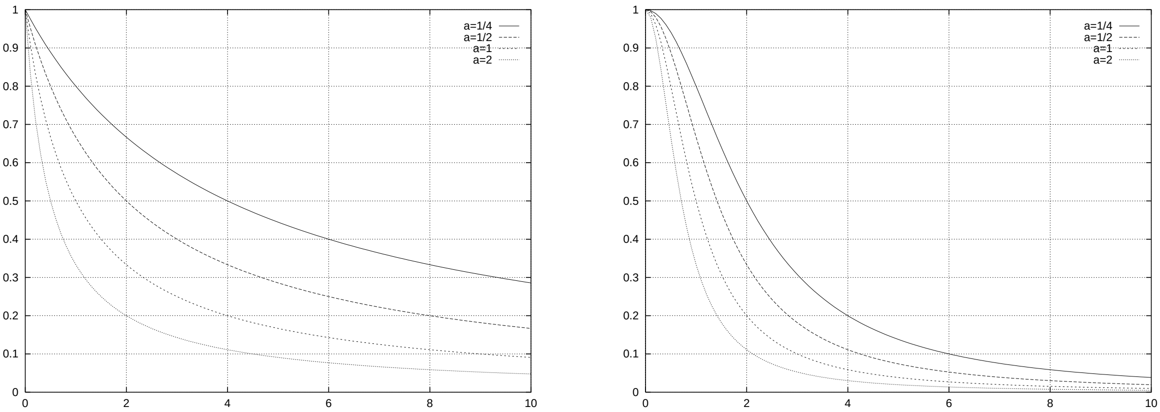


(c) Same as (a) after applying $g_0 = g_{11,0.5}$



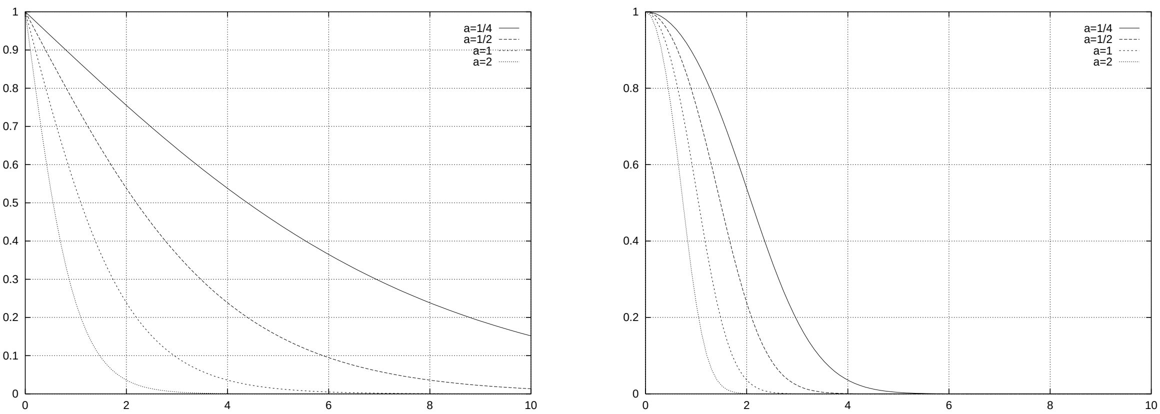
(d) Same as (b) after applying $g_0 = g_{11,0.5}$

Figure 13: Response of the class $\hat{s}_0(z)$.



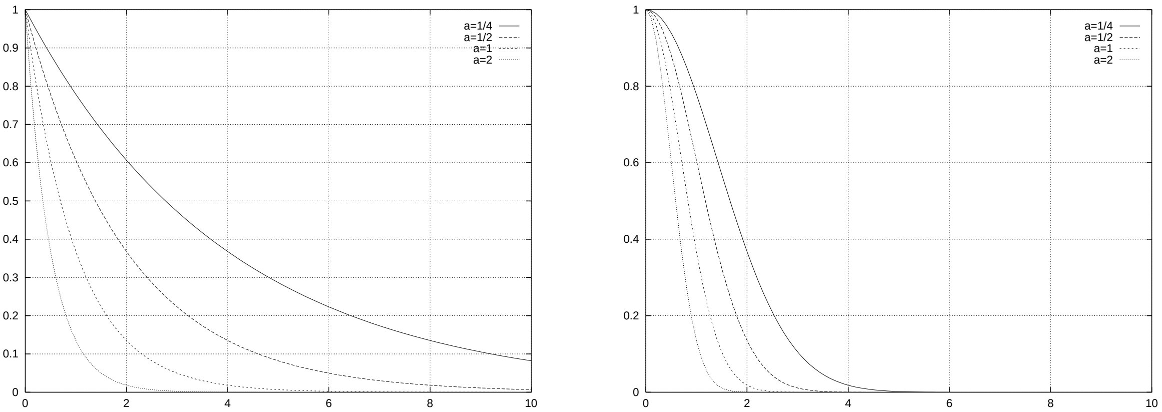
(a) Varying a for $\alpha = 1$ (b) Varying a for $\alpha = 2$

Figure 14: Response of the class $\hat{s}_1(z)$.



(a) Varying a for $\alpha = 1, b = 0$ (b) Varying a for $\alpha = 2, b = 0$

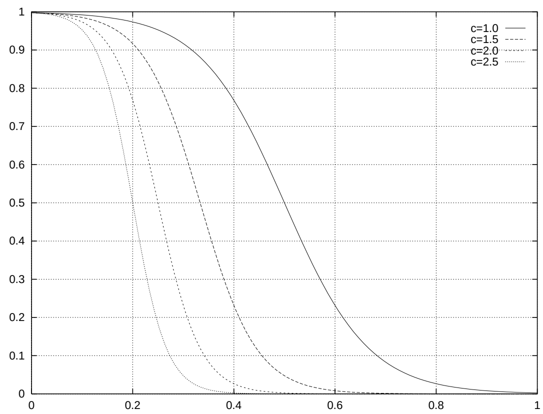
Figure 15: Response of the class $\hat{s}_2(z)$.



(a) Varying a for $\alpha = 1$

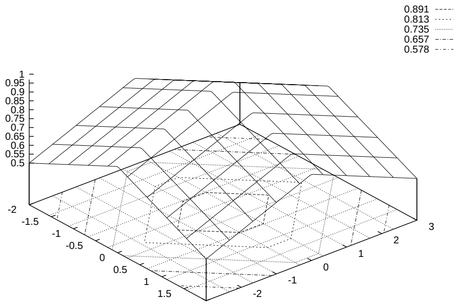
(b) Varying a for $\alpha = 2$

Figure 16: Response of the class $\hat{s}_3(z)$.

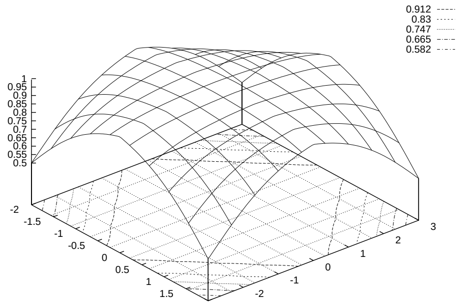


(a) Plots of $\hat{s}_4(z)$ for varying c.

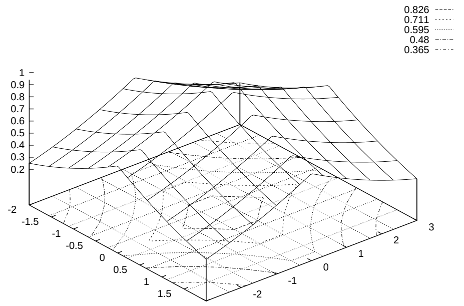
Figure 17: Response of the class $\hat{s}_4(z)$.



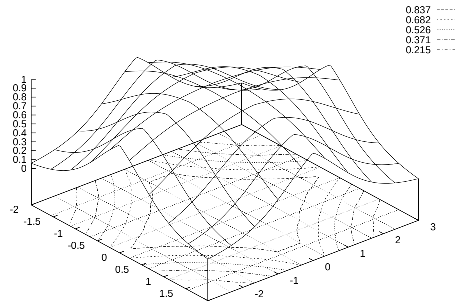
(a) $d = 1, \alpha = 1$, no $\breve{s}$



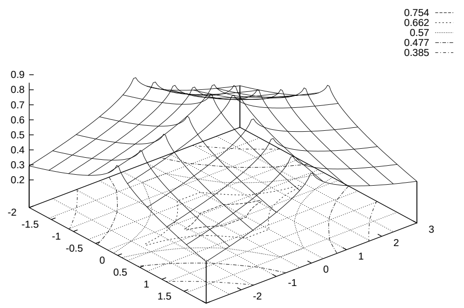
(b) $d = 1, \alpha = 1$, with $\breve{s}$



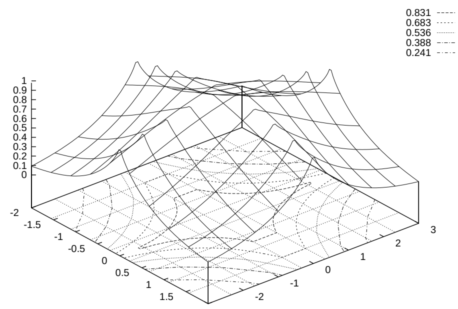
(c) $d = 1, \alpha = 2$, no $\breve{s}$



(d) $d = 1, \alpha = 2$, with $\breve{s}$



(e) $d = 0.5, \alpha = 1$, no $\breve{s}$



(f) $d = 0.5, \alpha = 1$, with $\breve{s}$

Figure 20: Example response of (88) for a reference weight vector $\vec{w}^i = (0, 0)$ and $\vec{x} \in [-2, 2] \times [-3, 3]$, with and without a further smoothing $\breve{s} = g_{11,0.5}$ non-linearity. Note how the response in (c) and (e) is less sensitive than in (a). The same happens for (d) and (f) against (b).

Bibliography

[1] D.E. Rumelhart, R. Durbin, R. Golden, Y. Chauvin. Backpropagation: The Basic Theory. In *Mathematical Perspectives of Neural Networks*. Smolensky, Mozer, Rumelhart (eds.), Lawrence Erlbaum, 1993.

[2] J.M. Sopena, E. Romero, R. Alquézar. Neural networks with periodic and monotonic activation functions: a comparative study in classification problems. In Procs. of the 9th International Conference on Artificial Neural Networks (ICANN'99), Edinburgh, Scotland, 1999.

[3] D.E. Rumelhart, G.E. Hinton, R.J. Williams. Learning internal representations by error propagation. In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition* (Vol. 1: Foundations). Rumelhart, McClelland (eds.), MIT Press, Cambridge, MA, 1986.

[4] T. Poggio, F. Girosi. A theory of networks for approximation and learning. AI Memo No. 1140, AI Laboratory, MIT, 1989.

[5] M. Roubens. Fuzzy sets and decision analysis. *Fuzzy sets and systems*, Vol. 90, No. 2, 1997.

[6] K. Steinbuch. Die lernmatrix. *Kybernetik*, 1: 36-45, 1961.

[7] W. McCulloch, W. Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5: 115-133, 1943.

[8] F. Rosenblatt. *Principles of neurodynamics*. Spartan Books, NY, 1962.

[9] G. Dorffner. A generalized view on learning in feedforward neural networks. In Cromme et al. (eds.), CoWAN'94, Technische Universität Cottbus, Reihe Mathematik M-01/1995, pp.34-54, 1995.

[10] S. Haykin. *Neural Networks: A Comprehensive Foundation*. MacMillan, 1994.

[11] R. Durbin, D.E. Rumelhart. Product units: A computationally powerful and biologically plausible extension to backpropagation networks. *Neural Computation*, 1: 133-142, 1989.

[12] M. Queysanne. *Álgebra básica*. Vicens Vives, Barcelona, 1985.

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

[14] A. Tversky. Features of Similarity. *Psychological Review*, Vol. 84, No. 4, 327-352, 1977.

[15] S. Edelman. Representation, Similarity and the Chorus of Prototypes. Dept. of Applied Mathematics and Computer Science. The Weizmann Institute of Science (1993). Also in *Minds and Machines*, 5:45-68, 1995.

[16] D. Hume. An enquiry concerning human understanding. In *The World's Great Thinkers. Man and Spirit: the Speculative Philosophers.* pp. 341-423. Random House, New York, 1947.

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

[18] V. Tresp, S. Ahmad, R. Neuneier. Training Neural Networks with Deficient Data. In Cowan, Tesauro and Alespector (eds.). Advances in Neural Information Processing Systems 6, San Mateo, CA, Morgan Kaufman, 1994.

[19] B. Everitt. *Cluster Analysis.* Heinemann Books, 1977.

[20] D. Nauck, F. Klawonn, R. Kruse. Fuzzy Sets, Fuzzy Controllers and Neural Networks. In Wissenschaftliche Zeitschrift der Humboldt-Universität zu Berlin, Reihe Medizin 41, Nr. 4: 99-120, 1992.

[21] L. Zadeh. Fuzzy Sets as a basis for a theory of possibility. *Fuzzy Sets and Systems*, 1: 3-28, 1978.

[22] L. Zadeh. The concept of a linguistic variable and its application to approximate reasoning. Parts 1, 2. *Information Science*, 8: 199-249, 301-357, 1975. Part 3, *op. cit.*, 9: 43-80, 1976.

[23] D. Dubois, H. Prade. The three semantics of fuzzy sets. *Fuzzy sets and systems*, Vol. 90, No. 2, 1997.

[24] D. Dubois, F. Esteva, P. García, Ll. Godo, H. Prade. A logical approach to interpolation based on similarity relations. Research Report IIIA 96/07. Instituto de Investigación en Inteligencia Artificial. España, 1996.

[25] D. Dubois, H. Prade, F. Esteva, P. García, L. Godo, R. López de Mántaras. Fuzzy set modelling in case-based reasoning. In Lecture Notes in Artificial Intelligence, LNAI-1266, pp. 599-610, Springer, 1997.

[26] G.J. Klir, T.A. Folger.: Fuzzy Sets, Uncertainty and Information. Prentice Hall, 1988.

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

[28] J.S. Roger Jang, C.T Sun. Neuro-Fuzzy Modeling and Control. Procs. of the IEEE, March 1995.

[29] J.C. Gower. A General Coefficient of Similarity and some of its Properties. *Biometrics*, 27: 857-871, 1971.

[30] L.M. Reyneri. Unification of Neural and Wavelet Networks and Fuzzy Systems. *IEEE Trans. on Neural Networks*, Volume 10, Number 4, 1999.

[31] C. Pappis, N. Karacapilidis. A comparative assessment of measures of similarity of fuzzy values. *Fuzzy sets and systems*, Vol. 56: 171-4, 1993.

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

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

[34] W.S. Sarle. (ed.) Neural Network FAQ. Usenet *comp.ai.neural-nets.* URL: *ftp://ftp.sas.com/pub/neural/FAQ.html*

[35] J. Moody. Prediction Risk and Architecture Selection for Neural Networks. In *From Statistics to Neural Networks: Theory and Pattern Recognition Applications.* Cherkassky, Friedman and Wechsler (eds.), NATO ASI Series F, Springer-Verlag, 1994.

[36] M.A. Wallach. On Psychological Similarity. *Psychological Review*, Vol. 65, No. 2, 103-116, 1958.

[37] A.N. Kolmogorov, S.V. Famin. Elementos de la teoría de funciones y del análisis funcional. Ed. Mir, Moscú, 1975.

[38] Y.T. Su, Y.T. Sheen. Neural Networks for System Identification. *Intl. Journal of Systems Science*, 23(12): 2171-2186, 1992.

[39] M.J.L. Orr. Introduction to Radial Basis Function Networks. Technical Report of the Centre for Cognitive Science, Univ. of Edinburgh, 1996.

[40] E. Jarauta. Anàlisi matemàtica d'una variable. Edicions UPC, Barcelona, 1993.

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

[42] D.E. Goldberg. *Genetic Algorithms for Search, Optimization & Machine Learning.* Addison-Wesley, 1989.

[43] Th. Bäck. Evolution Strategies: An alternative Evolutionary Algorithm. Technical Report of the Informatik Centrum, Dortmund.

[44] I. De Falco, A. Della Cioppa, P. Natale, E. Tarantino. Artificial Neural Networks Optimization by means of Evolutionary Algorithms. In *Soft Computing in Engineering Design and Manufacturing*, Springer-Verlag, London, 1997.

[45] Ll. Belanche. A Study in Function Optimization with the Breeder Genetic Algorithm. LSI Research Report LSI-99-36-R. Universitat Politècnica de Catalunya, 1999.

[46] Ll. Belanche. A Case Study in Neural Network Training with the Breeder Genetic Algorithm. LSI Research Report LSI-00-07-R. Universitat Politècnica de Catalunya, 2000.

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

[48] C. Globig, S. Lange, Case-based representability of classes of boolean functions. In ECAI-96: Procs. of Twelfth European Conf. on Artificial Intelligence, pp. 117-121, 1996.

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

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

A Additional material on distances

Definition A.1 *Let E be a vector space over $K = \mathbb{R}$ or $\mathbb{C}$. Every function ρ of E:*

$$\rho: \ E \to \mathbb{R}^{+} \cup \{0\}$$

such that, $\forall x, y \in E, \lambda \in K$:

1. $\rho(x) = 0 \Rightarrow x = 0$
2. $\rho(x+y) \leq \rho(x) + \rho(y)$
3. $\rho(\lambda x) = |\lambda| \rho(x)$

is a norm on E. A space E with a norm ρ is a normed space (E, ρ).

Proposition A.1 *Every normed space (E, ρ) can be converted in a metric space (E, d) by defining $d(x, y) = \rho(x - y)$.*

Proof: see [37].

Lemma A.1 *Let $E = E_1 \times E_2, \ldots, \times E_n$ and $d_1, \ldots, d_n$ such that $d_i \in D(E_i)$ (these functions are not necessarily equal). Then,*

$$d_\Sigma = \sum_{i=1}^{n} d_i \qquad \in D(E)$$

Proof: By fulfiment of the conditions in Definition (3.1). Forall $\vec{x}, \vec{y}, \vec{z} \in E$,

1. $d_\Sigma(\vec{x}, \vec{y}) = 0$
 $\equiv$ (def. of d_Σ)
 $\sum_{i=1}^{n} d_i(x_i, y_i) = 0$
 $\equiv$ ($d_i(x_i, y_i) \geq 0$)
 $\forall i : 1 \leq i \leq n : d_i(x_i, y_i) = 0$
 $\equiv$ (d_i are all distances)
 $\forall i : 1 \leq i \leq n : x_i = y_i$
 $\equiv$
 $\vec{x} = \vec{y}$ $\square$

2. $d_\Sigma(\vec{x}, \vec{y})$
 $\equiv$ (def. of d_Σ)
 $\sum_{i=1}^{n} d_i(x_i, y_i)$
 $\equiv$ (d_i are all distances)
 $\sum_{i=1}^{n} d_i(y_i, x_i)$
 $\equiv$ (def. of d_Σ)
 $d_\Sigma(\vec{y}, \vec{x})$ $\square$

3. $d_\Sigma(\vec{x}, \vec{y}) \leq d_\Sigma(\vec{x}, \vec{z}) + d_\Sigma(\vec{z}, \vec{y})$
 $\equiv$ (def. of d_Σ)
 $\sum_{i=1}^{n} d_i(x_i, y_i) \leq \sum_{i=1}^{n} d_i(x_i, z_i) + \sum_{i=1}^{n} d_i(z_i, z_i)$
 $\equiv$
 $\sum_{i=1}^{n} d_i(x_i, y_i) \leq \sum_{i=1}^{n} \{d_i(x_i, z_i) + d_i(z_i, z_i)\}$
 $\equiv$ (d_i are all distances)
 true $\square$

Lemma A.2 *Let $d \in D(E)$. Then,*

$$\forall \alpha \in \mathbb{R}^{+}, \ \alpha d \in D(E)$$

Proof: obvious, since α can be cancelled everywhere.

Lemma A.3 *Let $E = E_1 \times E_2, \ldots, \times E_n$ and $d_1, \ldots, d_n$ such that $d_i \in D(E_i)$ (these functions not necessarily equal). Then, any linear combination of the d_i with non-negative coefficients is a distance in E.*

Proof: By making use of Lemmas (A.1) and (A.2).

Lemma A.4 *Let $d \in D(E)$. Then,*

$$\forall \alpha \in (0, 1], \ d^{\alpha} \in D(E)$$

Proof: By fulfiment of the conditions in Definition (3.1). Forall $x, y, z \in E$,

1. $d(x, y)^{\alpha} = 0$
 $\equiv$
 $d(x, y) = 0$
 $\equiv$ (d is a distance)
 $x = y$ $\square$

2. $d(x,y)^\alpha = d(y,x)^\alpha$
   $\equiv$
   $d(x,y) = d(y,x)$
   $\equiv$ (d is a distance)
   $true$ $\square$

3. $d(x,y)^\alpha \leq d(x,z)^\alpha + d(z,y)^\alpha$
   $\equiv$ (defining $\beta = \frac{1}{\alpha} \geq 1$)
   $\sqrt[\beta]{d(x,y)} \leq \sqrt[\beta]{d(x,z)} + \sqrt[\beta]{d(z,y)}$
   $\equiv$ (raising to the β power)
   $d(x,y) \leq \left(\sqrt[\beta]{d(x,z)} + \sqrt[\beta]{d(z,y)}\right)^\beta$
   $\equiv$
   $d(x,y) \leq d(x,z) + d(z,y) + G(x,y,z,\beta)$
   $\equiv$ (d is a distance and $G(x,y,z,\beta) \geq 0$)
   $true$ $\square$

B Proofs of Propositions

Those not proven in the main body.

Proof of Proposition (3.1). Obvious, since $S \subset X$ implies $x,y,z \in S \Rightarrow x,y,z \in X$ and every $x,y,z \in X$ fulfills Definition (3.1) because (X,d) is a metric space $\square$.

Proof of Proposition (3.2). Refer to [12].

Proof of Proposition (3.3). Use $\mathbb{N} \subset \mathbb{Z} \subset \mathbb{Q} \subset \mathbb{R}$ and Proposition (3.1) $\square$.

Proof of Proposition (3.4). Let $\vec{z} = \{z_1, z_2, \ldots, z_n\} \in \mathbb{R}^n$ be a vector of non-negative components. Denote $\rho = \Theta_q^M, \forall q \geq 1 \in \mathbb{R}$. We prove this ρ is a norm in $\mathbb{R}^n$ by fulfilment of Definition (A.1).

1. $\rho(\vec{z}) = 0 \Rightarrow \vec{z} = \vec{0}$

2. $\rho(\vec{z_1} + \vec{z_2}) \leq \rho(\vec{z_1}) + \rho(\vec{z_2})$ First, we see that n' is a positive constant factor and can thus be cancelled, leading to:

$$\left(\sum_{i=1}^{n}\left(\frac{|z_i^1 + z_i^2|}{v_i}\right)^q\right)^{\frac{1}{q}} \leq$$

$$\left(\sum_{i=1}^{n}\left(\frac{|z_i^1|}{v_i}\right)^q\right)^{\frac{1}{q}} + \left(\sum_{i=1}^{n}\left(\frac{|z_i^2|}{v_i}\right)^q\right)^{\frac{1}{q}}$$

which is a weighted form of Minkowski's inequality [37]. Since the denominators v_i are also positive constants they do not alter the inequality.

3. $\rho(\lambda\vec{z}) = |\lambda|\rho(\vec{z})$. We have:

$$\rho(\lambda\vec{z}) = \left(\frac{1}{n'}\sum_{i=1}^{n}\left(\frac{|\lambda z_i|}{v_i}\right)^q\right)^{\frac{1}{q}}$$

($|\cdot|$ is a norm in $\mathbb{R}$)

$$= \left(\frac{1}{n'}\sum_{i=1}^{n}|\lambda|^q\left(\frac{|z_i|}{v_i}\right)^q\right)^{\frac{1}{q}}$$

$$= |\lambda|\left(\frac{1}{n'}\sum_{i=1}^{n}\left(\frac{|z_i|}{v_i}\right)^q\right)^{\frac{1}{q}} = |\lambda|\rho(\vec{z})$$

Proof of Proposition (3.5). By application of Propositions (3.4) and (A.1).

Proof of Proposition (3.6). First, note that an n-linear aggregation Θ fulfills the conditions of a general aggregation operator in (3.2). Second, since such an operator performs a linear combination of its arguments, and these are distances $d_i \in D(X^i)$, by Lemma (A.3), its n-linear aggregation Θ is a metric distance in $X = X^1 \times \ldots \times X^n$.

Proof of Proposition (3.7). It is clearly a n-linear operator (Definitions (3.4) and (3.5)).

Proof of Proposition (3.8). Is a particular case of Definition (3.3), for $\vec{v} = \vec{1}, n' = n$.

Proof of Proposition (3.9). By Proposition (3.7) for $\vec{v} = (\frac{1}{n}, \ldots, \frac{1}{n})$.

Proof of Proposition (3.10). We prove it by fulfiment of the conditions in Definition (3.6).

1. Non-negativity. By definition of Θ_s.
2. Symmetry. By the symmetry of Θ_s.
3. Boundedness. By definition of Θ_s.
4. Minimality. This reads:

   $\Theta_s(\{s_1, \ldots, s_n\}) \Leftrightarrow \forall i : 1 \leq i \leq n : s_i = s_{max}$

   (a) ($\Leftarrow$) By the idempotency of Θ_s.

   (b) ($\Rightarrow$) By contradiction. Suppose $\exists i : 1 \leq i \leq n : s_i < s_{max}$ such that $\Theta_s(\vec{s}) = s_{max}$. In this case, an $\vec{s}'$ such that $\vec{s}' = \{s_{max}, \ldots, s_{max}\}$ would yield, by the monotonicity property of Θ_s, $\Theta_s(\vec{s}') > \Theta_s(\vec{s}) = s_{max}$!
5. Semantics is expressed in conditions (v) to (vii).

Proof of Proposition (3.12). By fulfiment of the conditions in Definition (3.6).

1. Non-negativity. By definition of $\breve{s}$.
2. Symmetry. By the symmetry of s.
3. Boundedness. By definition of $\breve{s}$.
4. Minimality. This reads:

   $\breve{s}(s(x,y)) = \breve{s}_{max} \Leftrightarrow x = y$
   $\equiv (\breve{s}(z) = \breve{s}_{max}$ only for $z = s_{max})$
   $s(x,y) = s_{max} \Leftrightarrow x = y$
   $\equiv$ (s is a similarity)
   $true$ $\square$
5. The semantics of s is kept in $\breve{s} \circ s$ since $\breve{s}$ is a continuous and strictly increasing function.

Proof of Proposition (3.13). By fulfiment of the conditions in Definition (3.6).

1. Non-negativity. By definition of $\hat{s}$.
2. Symmetry. By symmetry of d.

   $s(x,y)$
   $\equiv$
   $\hat{s}(d(x,y))$
   $\equiv$ (d is a distance)
   $\hat{s}(d(y,x))$
   $\equiv$
   $s(y,x)$ $\square$
3. Boundedness. By definition of $\hat{s}$.
4. Minimality. This reads:

   $s(x,y) = s_{max} \Leftrightarrow x = y$
   $\equiv$
   $\hat{s}(d(x,y)) = s_{max} \Leftrightarrow x = y$
   $\equiv (\hat{s}(0) = s_{max}$, $\hat{s}$ is strictly decreasing)
   $d(x,y) = 0 \Leftrightarrow x = y$
   $\equiv$ (d is a distance)
   $true$ $\square$
5. The semantics of s is kept in $\breve{s} \circ s$ since $\breve{s}$ is a continuous and strictly increasing function.

Proof of Proposition (3.14). By consecutive application of Propositions (3.13) and (3.12).

Proof of Proposition (3.15). Since a normalized distance is a distance, the same proof as for Proposition (3.13) applies. The extra condition $\hat{s}(d_{max}) = 0$ is required to ensure that the formed similarity s covers $[0, s_{max}]$ (this is already guaranteed in Proposition (3.13), although not needed for the proof).

Proof of Proposition (3.16). It suffices to define:

$$\hat{s}'(z) = \frac{\hat{s}(z) - \hat{s}(d_{max})}{1 - \hat{s}(d_{max})}$$

Proof of Proposition (4.1). By fulfiment of the conditions in Definition (3.6).

1. Non-negativity. By definition of s.
2. Symmetry. By symmetry of s.
3. Boundedness. By definition of s.
4. Minimality. This reads:
   $s(x,y) = s_{max} \Leftrightarrow x = y$
   $\equiv (s_{max} = 1)$
   $s(x,y) = 1 \Leftrightarrow x = y$
   $\equiv$ (by def. of s)
   $true$ $\square$
5. The semantics of s is binary, in that either two objects are similar or not. This is an equivalence relation.

Proof of Propositions (4.2, 4.3). They are proven in the text, pages 28ss.

Proof of Proposition (4.4). By fulfiment of the conditions in Definition (3.6).

1. Non-negativity. By definition of s.
2. Symmetry. By symmetry of s.
3. Boundedness. By definition of s.
4. Minimality. This reads:
   $s(x,y) = s_{max} \Leftrightarrow x = y$
   $\equiv (s_{max} = 1)$
   $s(x,y) = 1 \Leftrightarrow x = y$
   $\equiv$ (by def. of s)
   $\frac{|x \cap y|}{|x \cup y|} = 1 \Leftrightarrow x = y$
   $\equiv$
   $|x \cap y| = |x \cup y| \Leftrightarrow x = y$
   $\equiv$ (set properties)
   $true$ $\square$
5. This measure has a clear semantics: number of shared elements w.r.t. to the number of different elements aported between the two sets. Interestingly, the empty set means the variable has no value and can thus be used to express missing information, with the semantics:

   $$s(\emptyset, x) = 0, \qquad \forall x \in \mathcal{S},$$

   in accordance to the definition. Alternatively, $s(\emptyset, x) = \mathcal{X}$ can be defined.

C Other Topics

Chebyshev inequality. Let X be a random variable with expected value μ and standard deviation σ. Then, $\forall t > 0$,

$$\Pr(|X - \mu| > t\sigma) \leq \frac{1}{t^2}$$

Proof of assertion in page (38). Let $D = \{x\}$ a real data set and $D' = \{x'\}$ the new (normalized) data set, with $x' = f(x)$ and $f(x) = ax + b$, a, b real constants obtained from D, $a > 0$. Assuming $D = [m, M]$, the similarity between any two $x, y \in D$ is given by (51):

$$s(x,y) = 1 - \frac{|x-y|}{M-m}$$

On the other hand, we have:

$1 - \frac{|x'-y'|}{M'-m'}$
$\equiv$
$1 - \frac{|f(x)-f(y)|}{f(M)-f(m)}$
$\equiv$
$1 - \frac{|ax+b-ay-b|}{aM+b-am-b}$
$\equiv$
$1 - \frac{|ax-ay|}{aM-am}$
$\equiv (a > 0)$
$1 - \frac{a|x-y|}{a(M-m)}$
$\equiv$
$1 - \frac{|x-y|}{M-m}$

Therefore $s(x, y) = s(x', y')$. This simple linear normalization scheme includes most of the commonly found methods:

1. To $D' = [0, 1]$, by setting $a = \frac{1}{M-m}, b = \frac{m}{M-m}$.
2. To zero mean, unit standard deviation, by setting $a = \frac{1}{\sigma_x}, b = -\frac{\mu_x}{\sigma_x}$.

An analogous proof can be derived for ordinal types. For nominal, the measure (48) is clearly not affected by any normalization. As for the fuzzy ones, their values are obtained from the continuous ones in such a way that scale is unimportant.