The use of similarity relations in neural systems

Lluís A. Belanche Muñoz

January 21, 2002

OVERVIEW

- General class of neuron models:
  - Exploits the notion of **similarity relations** between (input, weight)
  - Permits expression of **prior knowledge**
  - Handles **data heterogeneity**

- Comprehensive framework:
  - Inclusion of known models
  - Derivation of new ones

- Design of feed-forward (supervised) nets, Kohonen (unsupervised) nets, connection to Support Vector Machines

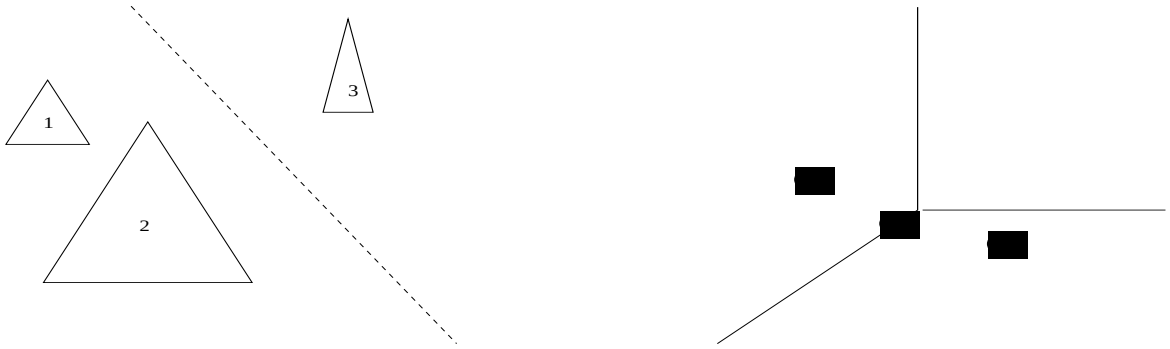
Principle 1 Similar patterns should yield similar outputs.

Principle 2 Similar patterns should have similar representations.

1. A similarity judgement depends on what input patterns mean to us.

2. *Physical similarity* may not be suitable to capture *functional* or *psychological* aspects.

Physical similarity: variables as points in $\mathbb{R}^n$.



Learning task: show the network the similarity relation.

Data heterogeneity in artificial neural networks

- Prior knowledge: strongly related to **input representation**.

- Real-world data come from many different **sources**.

- Patterns are described by mixtures of variables:
  - Ordered or lacking an order.
  - Continuous or discrete.
  - Crisp or fuzzy.
  - May have missing values.

- Require completely different treatments.

- This **heterogeneous** information has to be cast in the form of *real* values.

- This pre-processing is *not* part of the original task.

- May have deep consequences in the problem structure:
  - Change in input distribution.
  - Increase in dimension.
  - Growth in the number of weights.
  - Increment in training time.

- The network has to:
  - Discover the relations or structure induced by the coding scheme.
  - Find ways to accommodate the *underlying similarity* relationship to its fixed computation.

- Besides:
  - Solving any practical problem, having the network learn what is known in advance is an added and wasteful effort.
  - The difficulty of the learning task is exacerbated because data and problem are adapted to the neuron model.

Precise formulation of neuron models as heterogeneous **similarity computing devices**.

Definition. (Similarity, dissimilarity) *Proximity indexes fulfill properties (Chandon & Pinson'81):*

1. Non-negativity. $s_{ij} \geq 0 \qquad \delta_{ij} \geq 0$

2. Symmetry. $s_{ij} = s_{ji} \qquad \delta_{ij} = \delta_{ji}$

3. Boundedness. $s_{ij} \leq s_{max} \qquad \delta_{ij} \geq \delta_{min}$

4. Minimality *(Reflexivity in the strong sense)*.

$$s_{ij} = s_{max} \Leftrightarrow \vec{x}_i = \vec{x}_j$$

$$\delta_{ij} = \delta_{min} \Leftrightarrow \vec{x}_i = \vec{x}_j$$

5. Semantics. $s_{ij} > s_{ik}$

6. Transitivity s_{jk}, s_{ik}, s_{ij} *(?)*

- 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 as a means to express *psychological* aspects of similarity.

- 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.

Aggregation of similarities

Consider a collection of m quantities, grouped by convenience as a vector $\vec{s} = \{s_1, s_2, \ldots, s_m\}$, where $s_i \in [0, s_{max}] \cup \{\mathcal{X}\}$

A similarity aggregation operator is a function

$$\ominus : ([0, s_{max}] \cup \{\mathcal{X}\})^m \to [0, s_{max}] \cup \{\mathcal{X}\}$$

$\mathcal{X}$ denotes a missing component

Let $F_{\mathcal{X}}$ be a *filter* operator:

$$F_{\mathcal{X}} : ([0, s_{max}] \cup \{\mathcal{X}\})^m \to ([0, s_{max}])^n, \ (n \leq m)$$

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

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

Properties

i) Minimality

ii) Symmetry

iii) Monotonicity

iv) Idempotency $s_i^n[\Theta] = s_i, \forall n \in \mathbb{N}^+$.

- Includes the boundary values $s_{max}^n[\Theta] = s_{max}$ and $0^n[\Theta] = 0$.
- Implies $s_i^{n+1}[\Theta] = s_i^n[\Theta], \forall n \in \mathbb{N}^+$.

v) Cancellation law

$$\Theta(\{s_1, s_2\}) = \Theta(\{s_1, s_3\}),\ s_1 > 0 \Longrightarrow s_2 = s_3$$

vi) Continuity

- Smoothness to small changes in s_i.
- All values of $\Theta(\vec{s})$ can be generated.

vii) Compensativeness

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

A good (bad) s_i can be compensated by a bad (good) s_j.

viii) Treatment of missing components.

- $\Theta(\vec{s}) = \Theta(F_{\mathcal{X}}(\vec{s}))$
- Let $\vec{s}_{\mathcal{X}} = (\mathcal{X}, \ldots, \mathcal{X})$. Then $\Theta(\vec{s}_{\mathcal{X}}) = \mathcal{X}$.

Proposition. *A measure s obtained as $s = \Theta(\vec{s})$, provided $\vec{s} = \{s_1, \ldots, s_n\}$ are also similarities s_i in X_i, makes s a similarity measure in $X = X_1 \times X_2 \times \ldots \times X_n$.*

Proposition. *Valid families of Θ:*

1. *Additive measures:*

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

   - *$f : [0, s_{max}] \to [0, s_{max}]$ is increasing and continuous.*
   - *$f(0) = 0$ and $f(s_{max}) = s_{max}$.*

2. *Multiplicative measures:*

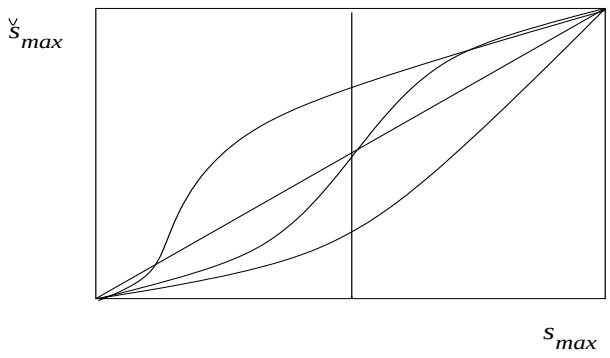
$$\Theta(\vec{s}) = f^{-1}\left(\prod_{i=1}^{n} f(s_i)^{v_i}\right)$$

Definition. (Similarity keeping) *A similarity keeping function (denoted $\bar{s}$) is a strictly increasing, bounded and continuous function $\bar{s}$: $[0, s_{max}] \to [0, \bar{s}_{max}]$ fulfilling:*

i) $\bar{s}(0) = 0$

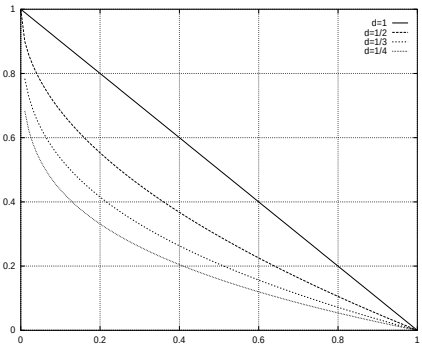
ii) $\bar{s}(s_{max}) = \bar{s}_{max} > 0$

iii) $\exists \epsilon > 0 \mid \quad \forall x \leq \epsilon \, \bar{s}(x) \leq x$

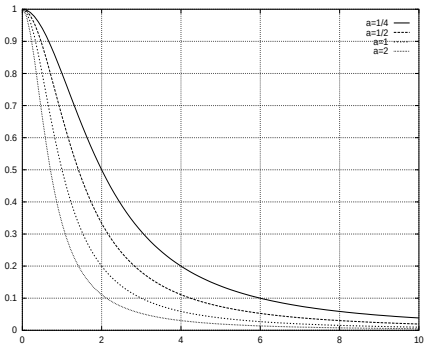


Proposition. *Let s be a similarity function in X defined on $[0, s_{max}]$ and $\bar{s}$ a similarity keeping function. Then $\bar{s} \circ s$ is a similarity function in X.*

Definition. (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 $\lim_{z \to +\infty} \hat{s}(z) = 0$.*



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



$$\hat{s}_1(z) = \frac{1}{1+(az)^\alpha}$$

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

Definition of heterogeneous measures

Nominal : Discrete, finite, no order.

e.g., car makers

Ordinal : Idem, ordered.

e.g., rank in a race, number of children.

Continuous : Numerical and crisp, ordered on a continuum.

Fuzzy Number : continuous, uncertainty as *imprecision*.

e.g., reliability or error of measured values.

Fuzzy interval : ordinal, uncertainty as *vagueness*. e.g., linguistic terms “small”, “cold”.

Definition. (H-neuron) *Given $\vec{x} \in \widehat{\mathcal{H}}^n$, an heterogeneous neuron or H-neuron is a function:*

$$F_i(\vec{x}) = \{h(\vec{x}, \vec{w}_i), \vec{w}_i \in \widehat{\mathcal{H}}^n\}$$

with $h : \widehat{\mathcal{H}}^n, \widehat{\mathcal{H}}^n \to \widehat{\mathcal{H}}$, where the domains $\widehat{\mathcal{H}}^n$ are **heterogeneous spaces**.

Definition. (S-neuron) *A similarity-based neuron or S-neuron is an H-neuron for which h is a similarity, pseudo-similarity or point similarity.*

- Classical models are included:

  1. Setting $\widehat{\mathcal{H}}^n = \mathbb{R}^n$

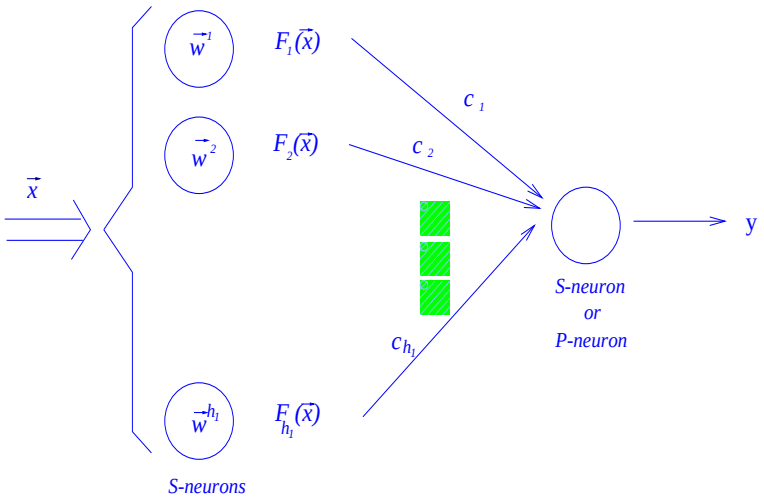
  2. They compute kinds of similarities

Flexibility of the framework:

1. Any valid set of partial similarity measures

2. Designed in accordance with the nature of their respective inputs, and

3. Grouped by means of a similarity aggregation operator.



The interest is not so much in embracing previous neuron models, but in devising **new ones**.

An example of a S-neuron (basic but useful)

- A Gower-like similarity index:

$$s_G(\vec{x}_i, \vec{x}_j) = \frac{\sum_{k=1}^{n} g_{ijk}\, \delta_{ijk}}{\sum_{k=1}^{n} \delta_{ijk}}$$

  - g_{ijk} is the partial similarity for $\vec{x}_i$, $\vec{x}_j$ for variable k.

  - δ_{ijk} expresses whether they are comparable or not.

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

- Additive aggregation operator, taking

  - $f(s_i) = s_i$ and $g_{ijk} = \mathcal{X}$ if $\delta_{ijk} = 0$.

  - Plus non-linear component:

$$s(\vec{x}, \vec{w}_i) = \bar{s}(s_G(\vec{x}, \vec{w}_i)) \qquad \vec{x}, \vec{w}_i \in \widehat{\mathcal{H}}^n$$

Heterogeneous Kohonen networks

- Emulate the human brain
- Topological & geometrical organization of data
- Goal: find analogies between *similar* data in a non-supervised process

Learning places geometrically close weight vectors also topologically close in the sheet represented by the network

$\Longrightarrow$ (in similarity terms)

For similar incoming data, the neurons (weight vectors) responding more vigorously should also be similar, and located in similar positions in the sheet.

Interest

- Discovery of *regularities* / tracking of input data distribution
- Result can be *visualized*
- Class labels can *color* the winning units
  - Cluster/class overlap analysis
  - Use it as a classifier

The Kohonen network algorithm

- Classical network: 2-d rectangular sheet
- Each neuron r has a n-dimensional weight vector $\vec{w}_r$
- Training step t:
  1. present input data sample $\vec{x}(t)$
  2. identify the BMU of $\vec{x}(t)$: the $\vec{w}_s(t)$ *most similar* to $\vec{x}(t)$
  3. shift $\vec{w}_s(t)$ and all neurons r in its neighbourhood towards $\vec{x}(t)$

Weight update

$$\vec{w}_r(t+1) = \vec{w}_r(t) + \epsilon(t)\, h_{rs}(t)\, (\vec{x}(t) - \vec{w}_r(t))$$

(for all units r in the grid)

$h_{rs}(t)$: scope and degree of change, centered at BMU s, at time t

$\epsilon(t)$ learning ratio: size of adaptive steps

(both decreasing functions of time)

$$h_{rs}(t) = exp\left\{-\frac{d(r,s)^2}{\sigma(t)^2}\right\}$$

BMU determination:

1. Greater input/weight vector dot product (requires normalization)

2. Smaller Euclidean distance (more general, requires standardization, usually adopted)

3. Two assumptions:

   - Problem features can be expressed as real-valued quantities

   - Dot product/distance are adequate means of computing similarity

BMU determination (similarity):

- For **nominal** variables:

$$s(x_k, y_k) = \begin{cases} 1 & \text{if } x_k = y_k \\ 0 & \text{if } x_k \neq y_k \end{cases}$$

- For **ordinal** variables:

$$s(x_k, y_k) = \frac{1}{1 + \frac{|x_k - y_k|}{rank(k)}}$$

$rank(k)$ = number of values of variable k

- For **continuous** variables:

$$s(x_k, y_k) = \frac{1}{1 + |x_k - y_k|}$$

where x_k, y_k are standardized variables.

- For **linguistic** variables (fuzzy intervals of the LR-type), define a *bell-shaped* membership function:

$$\mu_{\tilde{A}}(x) = \frac{1}{1 + (a(x - m))^2}$$

m is the mean and a controls fuzziness.

Let Δ a real interval, respective support sets $\Delta_{\tilde{A}}, \Delta_{\tilde{B}} \subset \Delta$, define:

$$I(\tilde{A}, \tilde{B}) = \int_{\Delta_{\tilde{A}} \cup \Delta_{\tilde{B}}} \mu_{\tilde{A} \cap \tilde{B}}(u) du;$$

$$U(\tilde{A}, \tilde{B}) = \int_{\Delta_{\tilde{A}} \cup \Delta_{\tilde{B}}} \mu_{\tilde{A} \cup \tilde{B}}(u) du$$

and then

$$s(x_k, y_k) = \frac{I(x_k, y_k)}{U(x_k, y_k)}$$

Weight update (similarity)

Let $w_{r,i}(t)$ represent the i-th component of weight vector $\vec{w}_r(t)$.

- For **continuous** variables: traditional formula

- For **ordinal** variables:

$$w_{r,i}(t+1) = \lfloor w_{r,i}(t) + \epsilon(t)\, h_{rs}(t) \\ |x_i(t) - w_{r,i}(t)| + 0.5 \rfloor$$

- For **linguistic** variables, formula for continuous variables applies to both m and a.

For **nominal** variables:

- Linear order (continuous or discrete) $\Rightarrow$ notion of "getting closer" makes sense.

- In absence of order, there is no shift, but a *change* in value (a more general concept).

- New intuition behind the product $\epsilon(t)\, h_{rs}(t)$: *probability* of such a change.

  Idea: a uniform random number $\xi \in [0,1]$ is generated. Then $w_{r,i}(t+1)$ is updated to $x_i(t)$ if $p_{r,s}(t) > \xi$, with $p_{r,s}(t) = \epsilon(t)\, h_{rs}(t)$, otherwise it is left unchanged.

  - intuitively pleasing but ignores the past

  - $p_{r,s}(t)$ increases proportionally to the number of times this change is attempted (relative to current value and $d(r,s)$).

Experimental comparison

Name	Cases	Heterogeneity	Missing
Credit Card	690(2)	6R-9N	0.52%
Heart Disease	303(2)	5R-7N-1I	0.00%
Horse Colic	364(2)	7R-4N-4I-5L	26.1%
Solar Flares	244(2)	6N-4I	0.00%
Gene Sequences	3175(3)	60N	0.00%
Mushroom	8124(2)	22N	1.26%
Cylinder bands	540(2)	20R-15N	5.29%
Meta data	528(3)	14R-2N-5I	0.00%
Servo data	167(2)	2R-2I	0.00%
Annealing data	898(6)	6R-10N-3I	12.2%

Basic features of the data sets. Missing refers to the *percentage* of missing values. R (real), N (nominal), I (ordinal) and L (linguistic).

1. Classical Kohonen network:

   - treating all variables as real-valued
   - 1-out-of-k coding for nominal ones
   - an extra variable for signaling missingness

2. As 1., but coding nominals as 0,1,2...

3. As 2., handling missing values directly

4. As 3, handling nominal and ordinal variables

5. , 6. As 4. using two different ways of handling nominal ones (prob. of change)

7. As 5., handling linguistic variables

Connection with Support Vector Machines

- The SVM is an inductive learning algorithm in the framework of Statistical Learning Theory

- An open research area is kernel design

- A kernel is a function $K : X \times X \longrightarrow \mathbb{R}$

$$K(x, y) = [\Phi(x), \Phi(y)]$$

with $\Psi : X \longrightarrow F$ a mapping and $[,]$ a dot product in F.

- It would be desirable to design kernels that:
  - Valid ones (as required by the SVM)
  - Capture relevant aspects of the problem

Standard kernels

- Polynomial: $K(x, y) = ([x, y] + 1)^d$
- RBF: $K(x, y) = exp\,(-\|x - y\|^2/\sigma^2)$
- MLP: $K(x, y) = tanh\,(\alpha[x, y] + \beta)$

Specialized kernels

- Pixel arrays (Schölkopf'97)
- Finite strings (Cristianini & Shawe-Taylor'00)
- Text categorization (Joachims'01)

What is the condition for a SVM valid kernel?

- Mercer's condition

$$\int_{X \times X} K(x, x') f(x) f(x') dx dx' \geq 0$$

for all $f \in L_2(X)$

- Gram matrix p.s.d. (positive semi-definite)

Given a finite data set $D = \{x_1, \cdots, x_l\}$

$$k_{ij} = K(x_i, x_j)$$

Kernels as Similarities

- Suppose we define a similarity s in X
- We can make s to be the equivalent of computing the dot product of elements in X in some other space F:

$$\text{similarity in } X \equiv \text{dot product in } F$$

- Interesting since the SVM can be expressed only in terms of dot products in F
- In contrast, neural networks use dot product in input space.

Needed properties of kernels

- Symmetry

$$K(x,y) = [\Phi(x), \Phi(y)] = [\Phi(y), \Phi(x)] = K(y,x)$$

- Cauchy-Schwartz

$$K(x,y)^2 = [\Phi(x), \Phi(y)]^2 \leq \|\Phi(x)\|^2 \|\Phi(y)\|^2$$

$$= [\Phi(x), \Phi(x)][\Phi(y), \Phi(y)] = K(x,x)K(y,y)$$

Proposition Gower's Gram matrix is p.s.d.

- Weighted average

$$s_G(\vec{x}_i, \vec{x}_j) = \frac{\sum_{k=1}^{n} s_k(x_{ik}, x_{jk})\, w_k}{\sum_{k=1}^{n} w_k}$$

- Continuous, nominal and binary variables

Kernel design:

- Design (partial) similarities s_k (single or group)
- Plug them in into a (weighted) s_G
- Extension to aggregation operators

Benefits

(apart from those inherent to the SVM!)

- Clear semantics
- Flexibility in design
- Handling of data heterogeneity
- Similarity would ensure p.s.d.-ness

Further (and current) work

- Characterization of induced mapping Φ
- Transitivity
- Missing value treatment