Heterogeneous Kohonen networks

Sergio Negri[1] and Lluís A. Belanche[2]

[1] Politecnico di Torino
Corso Duca degli Abruzzi, 16
Torino, Italy
sernegri@tin.it
[2] Dept. de Llenguatges i Sistemes Informàtics
Universitat Politècnica de Catalunya
c/Jordi Girona Salgado, 1-3
08034 Barcelona, Spain
belanche@lsi.upc.es

Abstract. A large number of practical problems involves elements that are described as a mixture of qualitative and quantitative information, and whose description is probably incomplete. The self-organizing map is an effective tool for visualization of high-dimensional continuous data. In this work, we extend the network and training algorithm to cope with heterogeneous information, as well as missing values. The classification performance on a collection of benchmarking data sets is compared in different configurations. Various visualization methods are suggested to aid users interpret post-training results.

1 Introduction

Kohonen networks (also known as self-organizing maps) [4, 7] were born to emulate the human brain characteristic of topological and geometrical organization of information. The training algorithm aims at finding analogies between *similar* incoming data in a non-supervised process.

The algorithm places the weight vectors such that geometrically close vectors (in weight space) are also topologically close in the grid represented by the network. In other words, for similar incoming vectors (in input space), the neurons responding more vigorously should also be similar (in terms of their weight vectors) and located in nearby positions in the network grid.

The interest in using these networks is not limited to the discovery of regularities or to tracking the input data density. Once the training process has ended, the result can be *visualized.* Furthermore, it the class labels are available, they can be superimposed to the discovered regularities. Much information can be extracted from such graphical plots, although it has to be identified with care.

1.1 Data heterogeneity

Real-world data come from many different sources, described by mixtures of numeric and qualitative variables. These variables include continuous or discrete numerical processes, symbolic information, etc. In particular, qualitative variables might have a different nature. Some are *ordinal* in the usual statistical sense (i.e., with a discrete domain composed by k categories, but totally ordered w.r.t a given relation) or *nominal* (discrete but without an ordering relation). The data also come with their own peculiarities (vagueness, uncertainty, incompleteness), and thus may require completely different treatments.

In the neural network paradigm, this *heterogeneity* is traditionally handled, if at all, by *preparing* the data using a number of coding methods, so that all variables are treated as real quantities. However, this pre-processing is not part of the original task and may have deep consequences in the structure of the problem. These consequences range from a change in input distribution to an increase in dimension, which results in a growth in the number of weights the network is forced to learn, an added difficulty in their interpretation, an increase in training time, and so on. The choice of representation (if any) should be as faithful as possible, in the sense that the relations between the represented entities should correspond to meaningful relations on the original data items.

1.2 Euclidean geometry

The common assumption of artificial neural models about the Euclidean nature of the input space leads naturally to the use of a scalar product or a distance metric as the standard neuron models (usually followed by a non-linear activation function). In particular, this space is taken to be $\mathbb{R}^n$, with the customary definition of scalar product. This assumption not only means that the features of the problem at hand can be expressed in terms of vectors of real quantities, but also that scalar product and Euclidean distance are adequate ways of measuring the similarity between elements in the space.

In consequence, in order to determine which neuron has a weight vector that is more similar to the input, there are two possibilities: choose the neuron which maximizes the scalar product or choose that which minimizes the distance. These methods may or may not be meaningful for a particular problem, and in principle are only appropriate for real-valued vectors. What to do in cases where input patterns contain heterogeneous information? In this case, a *similarity* index can be used, as a more flexible way to measure likeness.

1.3 Aims and structure

The aim of this work is to extend the main ideas of Kohonen networks in such a way that they can work in generic heterogeneous spaces, making the neurons compute a similarity measure among the elements of the space. This is to be done in stages, which we call *network configurations*, in order to appreciate the effect of each decision. The resulting networks are studied regarding two aspects: classification accuracy and ability to express meaningful information in a visual way. From the point of view of classification accuracy it is shown how the consideration of heterogeneous and/or incomplete information without the need of a coding scheme results in significantly better classifiers. On the other hand, it is illustrated how a solution can be visually analyzed.

The paper is organized as follows. In section (2) we introduce the data characteristics considered in this work. Sections (3) and (4) briefly review the Kohonen algorithm and the basics of a similarity measure. Section (5) describes the proposed extension of the algorithm. The last two sections present practical matters, about classification ability in benchmarking data sets –section (6)– and visualization of the results –section (7).

2 Data heterogeneity

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

Nominal (categorical) : non-numerical variable on which no order relation has been defined. It thus can be seen as having a *set* of values (finite or not).
Ordinal : variable (numerical or not) for which a linear order relation has been defined on a finite number of values, where each value has a precise sense.
Continuous : numerical and crisp variable for which a linear order relation has been defined on a continuum of values.
Linguistic : variable whose values are expressing uncertainty in the form of *vagueness* (e.g. *cool, fast, young*).

The values of the last type can be obtained —where appropriate— by converting an existing set of values (ordinal or continuous) into fuzzy quantities. In all cases, we assume some values may be missing in a particular data set.

3 The Kohonen network and algorithm

The classical Kohonen network [4,7] assumes a set of laterally interacting adaptive neurons, usually arranged as a two-dimensional sheet (a rectangular grid of neurons). Each neuron r is represented by an n-dimensional prototype vector $\boldsymbol{w}_r$, where n is the dimension of the input space. On each training step t, a data sample $\boldsymbol{x}(t)$ is presented to the network and the unit $\boldsymbol{w}_s$ *most similar* to $\boldsymbol{x}(t)$ is identified (the Best Matching Unit or BMU). The adaptation step shifts $\boldsymbol{w}_s$ (and those $\boldsymbol{w}_r$ corresponding to neighbouring units r to s) towards $\boldsymbol{x}(t)$:

$$\boldsymbol{w}_r(t+1) = \boldsymbol{w}_r(t)+\epsilon(t)\, h_{rs}(t)\, (\boldsymbol{x}(t)-\boldsymbol{w}_r(t)), \qquad \text{for all units } r \text{ in the grid.} \quad (1)$$

In this formula, $h_{rs}(t)$ establishes the scope and amount of the changes, centered on the BMU s, at time t. In other words, it represents the (varying) neighbourhood. The factor $\epsilon(t)$ acts as a learning ratio, controlling the size of the adaptive steps towards the input vector at time t. Both are decreasing functions of time. In this work, we use the following commonly found formulas:

$$h_{rs}(t) = exp\left\{-\frac{d(r,s)^2}{\sigma(t)^2}\right\} \text{ with } \sigma(t) = \sigma_{in}\left(\frac{\sigma_{fin}}{\sigma_{in}}\right)^{\frac{t}{t_{max}}} \tag{2}$$

$$\epsilon(t) = \begin{cases} \epsilon_0 + \frac{t}{t_1}(\epsilon_{t_1} - \epsilon_0) & \text{if } t \leq t_1 \\ \epsilon_{t_1} + \frac{t-t_1}{t_{max}-t_1}\epsilon_{t_1} & \text{if } t \geq t_1 \end{cases}, \qquad \text{for all } t \leq t_{max}. \tag{3}$$

In formula (2), $d(r,s)$ is the (topological) distance between units in the network structure. In case this structure is a rectangular grid, the standard Euclidean distance $d(r,s) = ||r - s||$ can be used. Note also that formula (1) ensures that the BMU would be the same in the hypothetical case $\boldsymbol{x}(t{+}1) = \boldsymbol{x}(t)$. The initial vectors $\boldsymbol{w}_r(0)$ are set to small random values.

As stated in (1.2), in order to determine the BMU, there are two basic possibilities: pick the neuron with the greatest scalar product to the input vector, or pick that with the smallest (Euclidean) distance. The first choice usually involves vector normalization (both input and weight). The Euclidean distance is a more general criterion (which reduces to scalar product for normalized vectors) and is usually adopted to compute similarity (closeness, in this case) in input space. In this work, all real-valued variables are standardized (to zero-mean, unit standard deviation) so that all of them have a uniform influence in the computation of distance.

4 Similarity measures

4.1 Definition

Let us represent input patterns belonging to a space $X \neq \emptyset$ (about which the only assumption is the existence of an equality relation) as vectors $\boldsymbol{x}_i$ 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. A *similarity measure* is a unique number expressing how "like" two given objects are, given only the features in A [2]. Let us denote by s_{ij} the similarity between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, that is, $s : X \times X \to \mathbb{R}^+ \cup \{0\}$ and $s_{ij} = s(\boldsymbol{x}_i, \boldsymbol{x}_j)$.

Definition 11 *A similarity measure in X fulfills the following properties:*

1. Non-negativity. $s_{ij} \geq 0 \quad \forall \boldsymbol{x}_i, \boldsymbol{x}_j \in X$
2. Symmetry. $s_{ij} = s_{ji} \quad \forall \boldsymbol{x}_i, \boldsymbol{x}_j \in X$
3. Boundedness. *There is a maximum attained similarity (that of an object with itself):* $\exists s_{max} \in \mathbb{R}^+ : \; s_{ij} \leq s_{max} \quad \forall \boldsymbol{x}_i, \boldsymbol{x}_j \in X.$
4. Minimality $s_{ij} = s_{max} \Leftrightarrow \boldsymbol{x}_i = \boldsymbol{x}_j \quad \forall \boldsymbol{x}_i, \boldsymbol{x}_j \in X$
5. Semantics. *The meaning of $s_{ij} > s_{ik}$ is that object i is more similar to object j than is to object k.*

4.2 Heterogeneous measures

In this section we define similarity measures (with $s_{max} = 1$) for the types of variables mentioned in (§2). Let $\boldsymbol{x}, \boldsymbol{y}$ be two heterogeneous vectors.

- For **nominal** variables:

$$s(x_k, y_k) = \begin{cases} 1 & \text{if } x_k = y_k \\ 0 & \text{if } x_k \neq y_k \end{cases} \tag{4}$$

- For **ordinal** variables:

$$s(x_k, y_k) = \frac{1}{1 + \frac{|x_k - y_k|}{rank(k)}} \tag{5}$$

  where $rank(k)$ is the number of values the variable k can take.
- For **continuous** variables:

$$s(x_k, y_k) = \frac{1}{1 + |x_k - y_k|} \tag{6}$$

  where x_k, y_k are standardized continuous variables.
- For **linguistic** variables (where their values are fuzzy intervals of the LR-type [9]) the problem is a bit more complex. We started using trapezoids, but we found that better results were in general achieved with the following bell-shaped function:

$$\mu_{\tilde{A}}(x) = \frac{1}{1 + (a(x - m))^2} \tag{7}$$

  where m is the mean and a controls the fuzziness. Let $\mathbb{F}(X)$ be the (crisp) set of all such fuzzy intervals in X. Given $\tilde{A}, \tilde{B} \in \mathbb{F}(\Delta)$, with Δ a real interval, and respective support sets $\Delta_{\tilde{A}}, \Delta_{\tilde{B}} \subset \Delta$, we define [1]:

$$I(\tilde{A}, \tilde{B}) = \int_{\Delta_{\tilde{A}} \cup \Delta_{\tilde{B}}} \mu_{\tilde{A} \cap \tilde{B}}(u) du; \qquad U(\tilde{A}, \tilde{B}) = \int_{\Delta_{\tilde{A}} \cup \Delta_{\tilde{B}}} \mu_{\tilde{A} \cup \tilde{B}}(u) du \tag{8}$$

  and then

$$s(x_k, y_k) = \frac{I(x_k, y_k)}{U(x_k, y_k)} \tag{9}$$

4.3 Gower's similarity index

A basic but very useful similarity-based neuron can be devised using a Gower-like similarity index, well-known in the literature on multivariate data analysis [3]. For any two objects $\boldsymbol{x}_i$, $\boldsymbol{x}_j$ of cardinality n, this index is given by the expression:

$$s_G(\boldsymbol{x}_i, \boldsymbol{x}_j) = \frac{\sum_{k=1}^{n} s_k(x_{ik}, x_{jk}) \delta_{ijk}}{\sum_{k=1}^{n} \delta_{ijk}} \tag{10}$$

where s_k is the partial similarity index according to variable k, and δ_{ijk} is a binary function expressing whether the objects are *comparable* or not according to variable k. Let $\mathcal{X}$ represent a missing value, then:

$$\delta_{ijk} = \begin{cases} 1 & \text{if } x_{ik} \neq \mathcal{X} \wedge x_{jk} \neq \mathcal{X} \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

5 Heterogeneous algorithm for the Kohonen network

The weight propagation in equation (1) can only be applied to real values. For other variable types, the following propagation formulas are used. Let $w_{r,i}(t)$ represent the i-th component of weight vector $\boldsymbol{w}_r(t)$.

- For **ordinal** variables:

$$w_{r,i}(t+1) = \lfloor w_{r,i}(t) + \epsilon(t)\, h_{rs}(t)\, (x_i(t) - w_{r,i}(t)) + 0.5 \rfloor \tag{12}$$

- For **linguistic** variables, formula (1) is applied to both m and a in (7).
- For **nominal** variables, a deeper explanation is needed. In all of the previous cases, there is a linear order (continuous or discrete). Hence the notion of "getting closer" to the input vector makes sense and the basic formula (1) can be applied. In absence of an order, there is no shift, but a *change* in value (which can be regarded as a more general concept). In addition, though the intuition behind the product $\epsilon(t)\, h_{rs}(t)$ must be kept, its practical role has to be different. We take it as the *probability* of such a change.
  In essence, a random number $\xi \in [0,1]$ with uniform probability is generated. Then, the updating rule is given by:

$$w_{r,i}(t+1) = \begin{cases} x_i(t) & \text{if } \epsilon(t)\, h_{rs}(t) > \xi \\ w_{r,i}(t) & \text{otherwise} \end{cases} \tag{13}$$

  This scheme is intuitively pleasing but has a serious drawback: it ignores the past. We hence propose an updating rule in which the probability of changing the weight vector component increases proportionally to the number of times this change was already attempted, relative to the number of attempts of the current value of the weight and the rest of possible values, and influenced by the distance $d(r,s)$, where s is the BMU for $\boldsymbol{x}(t)$ (see the Appendix for details).

6 An experimental comparison

A number of experiments are carried out to illustrate the validity of the approach, using several benchmarking problems. These are selected as representatives because of the diversity in the kind of problem and richness in data heterogeneity.

6.1 Problem description

A total of ten learning tasks are worked out, taken from [6] and [5], and altogether representative of the kinds of variables typically found in real problems, while displaying different degrees of missing information (from 0% to 26%). Their main characteristics are displayed in Table 1.

Table 1. Basic features of the data sets. Missing refers to the *percentage* of missing values. R (real), N (nominal), I (ordinal) and L (linguistic).

Name	Cases	Heterogeneity	Classes	Class distribution	Missing
Credit Card	690	6R-9N-0I-0L	2	55.5%/44.5%	0.52%
Heart Disease	303	5R-7N-1I-0L	2	54.1%/45.9%	0.00%
Horse Colic	364	7R-4N-4I-5L	2	61.5%/24.2%/14.3%	26.1%
Solar Flares	244	0R-6N-4I-0L	2	70.9%/29.1%	0.00%
Gene Sequences	3175	0R-60N-0I-0L	3	25%/25%/50%	0.00%
Mushroom	8124	0R-22N-0I-0L	2	51.8%/49.2%	1.26%
Cylinder bands	540	20R-15N-0I-0L	2	42.2%/57.8%	5.29%
Meta data	528	14R-2N-5I-0L	3	32.4%/37.1%/30.5%	0.00%
Servo data	167	2R-0N-2I-0L	2	47.3%/52.7%	0.00%
Annealing data	898	6R-10N-3I-0L	6	1.75%/10.0%/75.5% 0.25%/7.75%/4.75%	12.2%

6.2 Different network configurations

The extended algorithm has been tested in seven different configurations. In all cases, formula (10) is used, as follows:

1. Kohonen network, treating all attributes as real-valued, with the usual 1-out-of-k coding for nominal ones (with k the number of values) and an extra attribute signaling a missing value for those attributes that can be missing.
2. As 1., but coding nominals as 0,1,2...
3. As 2., without the coding for missing values –thus treating them directly by using formula (10).
4. As 3, using formula (12) for ordinal attributes, (considering all linguistic attributes as ordinals) and (13) for nominals.
5. and 6., as 4. but using two different ways of enhancing (13) for nominals (see the Appendix).
7. As 5., considering linguistic attributes as indicated.

In all cases, the interest is not in building a classifier *per se*, but in assessing the relative differences between network configurations. Hence, all the information is used to train the network (without the class labels). After this process has ended, the network is *colored* using the weighted global method – see section (7)– and its classification accuracy computed. The results are shown graphically in Fig. (1).

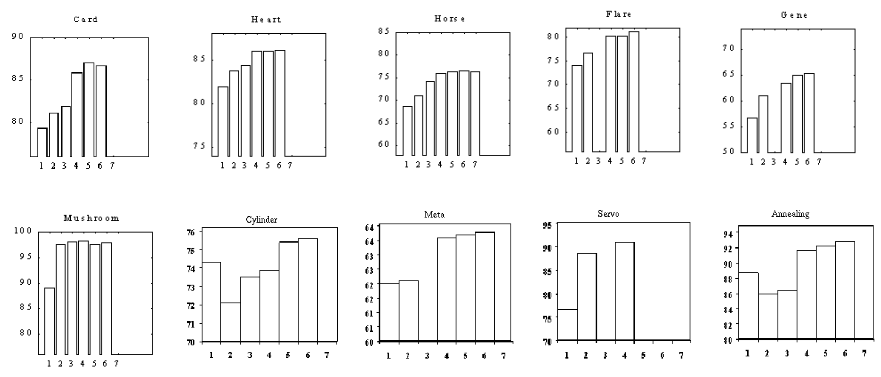

Fig. 1. Graphical impression of the results. The horizontal axis stands for the network configuration. The vertical axis shows classification accuracy. Each plotted value is the average of forty runs.

7 Visualization methods

In order to help users understand the post-training result of the network we use two different visualization methods.

7.1 Weighted global BMU

At the end of the training process we assign a class to each neuron of the grid (that is, we *color* the grid). For each neuron r, we compute its *receptive field* $F(r)$, composed of all those input vectors for which its BMU is r. A basic criterion is then to assign to r the majority class in $F(r)$. This criterion does not take into account the relevance of each vector in $F(r)$. A simple way to do this is to weight each vector in $F(r)$ by its similarity to $\boldsymbol{w}_r$. Afterwards, the class with greater weighted similarity to $\boldsymbol{w}_r$ is chosen to label r. If a neuron is never a BMU, it is not assigned to any class, and is painted "neutral" –*white* in Fig. 2 (left).

7.2 U-matrix

As shown in [8] the U-matrix is useful to visualize similarity among contiguous map units. The *minimum* similarity between each grid unit and its four adjacent neighbours, is computed, and displayed using grey shade –Fig. 2 (center). We have also found very useful displaying the *contour plot* of the same U-matrix –Fig. 2 (right)– and compare it to the weighted global BMU.

8 Conclusions

An extension of Kohonen's self-organizing network and the corresponding training algorithm has been introduced that works in heterogeneous spaces (for

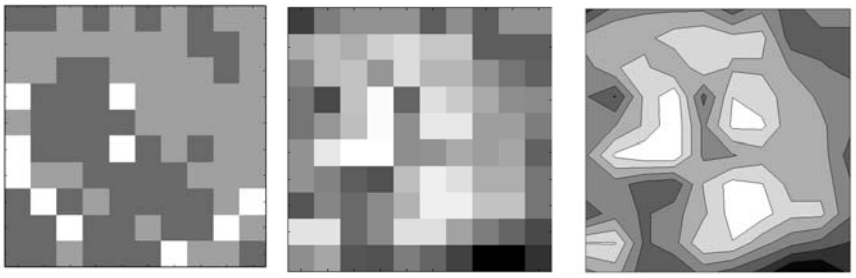

Fig. 2. Visualization of the results in a two-class example. Left: *Global BMU*. Center: *U-matrix*. Right: *U-matrix contour plot.*

both input and weight vectors). The inner workings of a neuron are grounded on a similarity measure, which allows for more flexibility and a built-in treatment of heterogeneous or incomplete data. It is also shown how to visualize the result of the training process. Some results that have been obtained for illustrative purposes in several benchmarking data sets indicate a superior performance of the extended algorithm.

Acknowledgements:

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

References

1. Belanche, Ll. Similarity-based Heterogeneous Neuron Models. In Procs. of ECAI'2000: European Conf. on Artificial Intelligence. IOS Press, 2000.
2. Chandon, J., Pinson, S. *Analyse Typologique. Théorie et Applications.* Masson, 1981.
3. Gower, J.C. A General Coefficient of Similarity and some of its Properties. *Biometrics*, 27: 857-871, 1971.
4. Kohonen, T. *Self-Organization and Associative Memory.* Springer-Verlag, 1988.
5. Murphy, P.M., Aha, D. UCI Repository of machine learning databases. UCI Dept. of Information and Computer Science, 1991.
6. Prechelt, L. Proben1: A set of Neural Network Benchmark Problems and Benchmarking Rules. Fac. für Informatik, Univ. Karlsruhe. Tech. Report 21/94, 1994.
7. Ritter, H., Kohonen, T. Self-Organizing Semantic Maps. *Biological Cybernetics*, 61: 241-254, 1989.
8. Vesanto, J. SOM-based data visualization methods. *Intel. data analysis* 3(2), 1999.
9. Zimmermann, H.J. *Fuzzy set theory and its applications.* Kluwer Acad. Publ., 1992.

A Appendix

During weight propagation, for nominal variables, a random number $\xi \in [0,1]$ with uniform probability is generated. Then, $w_{r,i}(t+1)$ is updated to $x_i(t)$ if $p(t) > \xi$, with $p(t) = \epsilon(t)\, h_{rs}(t)$, otherwise it is left unchanged.

Let n_α be the number of times $w_{r,i}(t)$ has been proposed to be the value of $w_{r,i}$ up to time t, n_β the number of times $x_i(t)$ has been proposed, and let n_γ generally denote the number of times *any other* possible value of nominal variable i has been proposed to be $w_{r,i}$, up to time t. If there are no more possible values (that is, if $rank(i) = 2$), then n_γ is undefined. Define then:

$$f(t) = \begin{cases} \frac{n_\beta^2}{n_\alpha} \sqrt{\frac{rank(i)-2}{\sum\limits_{\gamma=1,\gamma\neq\alpha,\gamma\neq\beta}^{rank(i)} n_\gamma^2}} & \text{if } rank(i) > 2 \\ \frac{n_\beta^2}{n_\alpha} & \text{if } rank(i) = 2 \end{cases} \tag{14}$$

In these conditions, the new probability of change is defined as:

$$p'(t) = (\epsilon(t)\, h_{rs}(t))^{\frac{1}{ln(e-1+f(t))}} \tag{15}$$

Whereas a simpler possibility is:

$$p''(t) = (\epsilon(t)\, h_{rs}(t))^{\frac{1}{f(t)}} \tag{16}$$

Network configuration 4. uses $p(t)$, configuration 5. uses $p'(t)$ and configuration 6. uses $p''(t)$.