Available online at www.sciencedirect.com
SCIENCE DIRECT
Neural Networks 19 (2006) 855–863
Neural Networks
www.elsevier.com/locate/neunet

2006 Special Issue

Fast algorithm and implementation of dissimilarity self-organizing maps

Brieuc Conan-Guez[a], Fabrice Rossi[b,*], Aïcha El Golli[b]

[a] *LITA EA3097, Université de Metz, Ile du Saulcy, F-57045 Metz, France*
[b] *Projet AxIS, INRIA, Domaine de Voluceau, Rocquencourt, B.P. 105, 78153 Le, Chesnay Cedex, France*

Abstract

In many real-world applications, data cannot be accurately represented by vectors. In those situations, one possible solution is to rely on dissimilarity measures that enable a sensible comparison between observations.

Kohonen's self-organizing map (SOM) has been adapted to data described only through their dissimilarity matrix. This algorithm provides both nonlinear projection and clustering of nonvector data. Unfortunately, the algorithm suffers from a high cost that makes it quite difficult to use with voluminous data sets. In this paper, we propose a new algorithm that provides an important reduction in the theoretical cost of the dissimilarity SOM without changing its outcome (the results are exactly the same as those obtained with the original algorithm). Moreover, we introduce implementation methods that result in very short running times.

Improvements deduced from the theoretical cost model are validated on simulated and real-world data (a word list clustering problem). We also demonstrate that the proposed implementation methods reduce the running time of the fast algorithm by a factor up to three over a standard implementation.

Keywords: Fast implementation; Self-organizing map; Clustering; Nonlinear projection; Unsupervised learning; Dissimilarity data; Proximity Data; Pairwise Data

1. Introduction

The vast majority of currently available data analysis methods are based on a vector model in which observations are described with a fixed number of real values, i.e. by vectors from a fixed and finite-dimensional vector space. Unfortunately, many real-world data depart strongly from this model. It is quite common, for instance, to have variable-sized data. They are natural, for example, in online handwriting recognition (Bahlmann & Burkhardt, 2004) where the representation of a character drawn by the user can vary in length because of the drawing conditions. Other data, such as texts for instance, are strongly non-numerical and have a complex internal structure: they are very difficult to represent accurately in a vector space. While a lot of work has been done to adapt classical data analysis methods to structured data such as tree and graph (see, e.g., Hammer, Micheli, Strickert, and Sperduti (2004) for neural-based unsupervised processing of structured data and also Hammer and Jain (2004), Hammer and Villmann (2005)), as well as to data with varying size, there is still a strong need for efficient and flexible data analysis methods that can be applied to any type of data.

A way to design such methods is to rely on one-to-one comparison between observations. In general, it is possible to define a similarity or a dissimilarity measure between arbitrary data, as long as comparing them is meaningful. In general, data analysis algorithms based solely on (dis)similarities between observations are more complex than their vector counterparts, but they are universal and can therefore be applied to any kind of data. Moreover, they allow one to rely on specific (dis)similarities constructed by experts rather than on a vector representation of the data that, in general, induces unwanted distortion in the observations.

Many algorithms have been adapted to use solely dissimilarities between data. In the clustering field, the k-means algorithm (MacQueen, 1967) has been adapted to dissimilarity data under the name of Partitioning Around Medoids (PAM, Kaufman & Rousseeuw, 1987). More recently, approaches based on deterministic annealing have been used to propose another class of extensions of the k-means

* Corresponding author. Tel.: +33 1 39 63 54 45; fax: +33 1 39 63 58 92.
E-mail addresses: Brieuc.Conan-Guez@univ-metz.fr (B. Conan-Guez), Fabrice.Rossi@inria.fr (F. Rossi), Aicha.ElGolli@inria.fr (A. El Golli).

0893-6080/$ - see front matter © 2006 Elsevier Ltd. All rights reserved.
doi:10.1016/j.neunet.2006.05.002

principle (see Buhmann and Hofmann (1994) and Hofmann and Buhmann (1995, 1997). Following the path taken for the k-means, several adaptations of Kohonen's self-organizing map (SOM, Kohonen, 1995) to dissimilarity data have been proposed. Ambroise and Govaert (1996) proposed a probabilistic formulation of the SOM that can be used directly for dissimilarity data. Deterministic annealing schemes have also been used for the SOM (Graepel, Burger, & Obermayer, 1998; Graepel & Obermayer, 1999; Seo & Obermayer, 2004). In the present paper, we focus on an adaptation proposed in Kohonen and Somervuo (1998, 2002), where it was applied successfully to a protein sequence clustering and visualization problem, as well as to string clustering problems. This generalization is called the Dissimilarity SOM (DSOM, also known as the median SOM), and can be considered as a SOM formulation of the PAM method. Variants of the DSOM were applied to temperature time series (El Golli, Conan-Guez, & Rossi, 2004a), spectrometric data (El Golli, Conan-Guez, & Rossi, 2004b) and web usage data (Rossi, El Golli, & Lechevallier, 2005).

A major drawback of the DSOM is that its running time can be very high, especially when compared to the standard vector SOM. It is well known that the SOM algorithm behaves linearly with the number of input data (see, e.g., Kohonen (1995)). In contrast, the DSOM behaves quadratically with this number (see Section 2.2). In this paper, we propose several modifications of the basic algorithm that allow a much faster implementation. The quadratic nature of the algorithm cannot be avoided, essentially because dissimilarity data are intrinsically described by a quadratic number of one-to-one dissimilarities. Nevertheless, the standard DSOM algorithm cost is proportional to $N^2M + NM^2$, where N is the number of observations and M the number of clusters that the algorithm has to produce, whereas our modifications lead to a cost proportional to $N^2 + NM^2$. Moreover, a specific implementation strategy reduces the actual computation burden even more. An important property of all our modifications is that the obtained algorithm produces **exactly** the same results as the standard DSOM algorithm.

The paper is organized as follows. In Section 2, we recall the SOM adaptation to dissimilarity data and obtain the theoretical cost of the DSOM. In Section 3, we describe our proposed new algorithm as well as the implementation techniques that decrease its running time in practice. Finally, we evaluate the algorithm in Section 4. This evaluation validates the theoretical cost model and shows that the implementation methods reduce the running time. The evaluation is conducted on simulated data and on real-world data (a word list clustering problem).

2. Self-organizing maps for dissimilarity data

2.1. A batch SOM for dissimilarity data

In this section, we recall the DSOM principle, as proposed in Kohonen and Somervuo (1998, 2002). Let us consider N input data from an arbitrary input space $\mathcal{X}$, $(\mathbf{x}_i)_{1\leq i\leq N}$. The set of those N data is denoted $\mathcal{D}$. The only available information on the data set is the dissimilarities between its elements: the dissimilarity between $\mathbf{x}_i$ and $\mathbf{x}_k$ is denoted $d(\mathbf{x}_i, \mathbf{x}_k)$. We assume standard dissimilarity behaviour for d, that is, d is symmetric, positive and $d(\mathbf{x}_i, \mathbf{x}_i) = 0$.

As the standard SOM, the DSOM maps input data from an input space to a low-dimensional organized set of M models (or neurons) which are numbered from 1 to M, and arranged via a prior structure (a grid in general). Model j is associated with an element of $\mathcal{D}$, its prototype, denoted $\mathbf{m}_j$ (therefore, for each j there is an i that depends on j, such that $\mathbf{m}_j = \mathbf{x}_i$): this is the first difference with the standard SOM in which prototypes can take arbitrary values in the input space.

The prior structure on the models is represented by an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ whose vertices are model numbers (i.e. $\mathcal{V} = \{1, \ldots, M\}$). We denote $g(j, k)$ as the length of the shortest path in $\mathcal{G}$ from j to k. Given a kernel-like function K that is a decreasing function from $\mathbb{R}^+$ to $\mathbb{R}^+$, with $K(0) = 1$ and $\lim_{s\to\infty} K(s) = 0$, the neighbourhood relationship between models j and k, $h(j, k)$, is defined by $h(j, k) = K(g(j, k))$. As for the standard SOM, the kernel is modified during training: at the beginning, the neighbourhood is very broad to allow organization to take place. The kernel sharpens during training and models become more and more adapted to a small subset of the data.

Given this information, the Dissimilarity SOM algorithm can be defined (see Algorithm 1).

While some initialization techniques proposed for the standard SOM can be extended to the case of dissimilarity data (see Kohonen and Somervuo (1998)), in this article we use a simple random choice: $\mathcal{M}^0$ is a random subset of the data set.

After initialization, the algorithm runs for L epochs. One epoch consists of an *affectation phase*, in which each input is associated with a model, followed by a *representation phase* in which the prototype of each model is updated. The DSOM is therefore modelled after the batch version of the SOM. As mentioned above, the neighbourhood relationship depends on l: at epoch l, we use h^l (see Eq. (2)).

At the end of the algorithm, an additional affectation phase can be performed to calculate $c^{l+1}(i)$ for all i and to define M clusters $\mathcal{C}_1^{l+1}, \ldots, \mathcal{C}_M^{l+1}$ with $\mathcal{C}_j^{l+1} = \{1 \leq i \leq N | c^{l+1}(i) = j\}$.

It should be noted that in practice, as pointed out in Kohonen and Somervuo (1998), the simple affectation phase of Eq. (1) induces some difficulties: for certain types of dissimilarity measure, the optimization problem of Eq. (1) has many solutions. A tie-breaking rule should be used. In this paper, we use the affectation method proposed in Kohonen and Somervuo (1998). In short, it consists of using a growing neighbourhood around each neuron to build an affinity of a given observation to the neuron. Details can be found in Kohonen and Somervuo (1998). Other tie-breaking methods have been proposed, for instance in El Golli et al. (2004a, 2004b). They give similar results and have the same worst-case complexity. However, the method of Kohonen and Somervuo (1998) has a smaller best-case complexity.

Algorithm 1 provides a general template. In the rest of this paper, we provide mostly partial algorithms (called schemes) that fill the missing parts of Algorithm 1.