Neural Networks 37 (2013) 52–65

Contents lists available at SciVerse ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

Essentials of the self-organizing map

Teuvo Kohonen

Aalto University, School of Science, P.O. Box 15400, FI-00076 AALTO, Finland

ARTICLE INFO

Keywords:
Self-organizing map
SOM
Data analysis
Brain map
Similarity
Vector quantization

ABSTRACT

The self-organizing map (SOM) is an automatic data-analysis method. It is widely applied to clustering problems and data exploration in industry, finance, natural sciences, and linguistics. The most extensive applications, exemplified in this paper, can be found in the management of massive textual databases and in bioinformatics. The SOM is related to the classical vector quantization (VQ), which is used extensively in digital signal processing and transmission. Like in VQ, the SOM represents a distribution of input data items using a finite set of models. In the SOM, however, these models are automatically associated with the nodes of a regular (usually two-dimensional) grid in an orderly fashion such that more similar models become automatically associated with nodes that are adjacent in the grid, whereas less similar models are situated farther away from each other in the grid. This organization, a kind of similarity diagram of the models, makes it possible to obtain an insight into the topographic relationships of data, especially of high-dimensional data items. If the data items belong to certain predetermined classes, the models (and the nodes) can be calibrated according to these classes. An unknown input item is then classified according to that node, the model of which is most similar with it in some metric used in the construction of the SOM. A new finding introduced in this paper is that an input item can even more accurately be represented by a linear mixture of a few best-matching models. This becomes possible by a least-squares fitting procedure where the coefficients in the linear mixture of models are constrained to nonnegative values.

1. Brain maps

It has been known for over hundred years that various cortical areas of the brain are specialized to different modalities of cognitive functions. However, it was not until, e.g., Mountcastle (1957) as well as Hubel and Wiesel (1962) found that certain single neural cells in the brain respond selectively to some specific sensory stimuli. These cells often form local assemblies, in which their topographic location corresponds to some feature value of a specific stimulus in an orderly fashion. Such systems of cells are called *brain maps*.

It was believed first that the brain maps are determined genetically, like the other bodily formations and organizations. It was not until many of these maps, at least their fine structures and feature scales were found to depend on sensory experiences and other occurrences. Studies of brain maps that are strongly modified by experiences have been reported especially by Merzenich et al. (1983).

Among some theoretical biologists in the 1970s, e.g. Grossberg (1976), Nass and Cooper (1975), and Perez, Glass, and Shlaer (1975), the question arose whether feature-sensitive cells could be formed also in artificial systems automatically, by learning (i.e., adaptation to simulated sensory stimuli). However, already Malsburg (1973), and later Amari (1980) demonstrated that their topographic order may also ensue from the input data.

The above modeling approaches deserve to be mentioned among the first successful theoretical proofs of input-driven self organization. In them, the emergence of feature-sensitive cells was implemented by the so-called *competitively learning neural networks*. In a subset of cells, adaptation of the strongest-activated cells to the afferent input signals made them become tuned to specific input features or their combinations.

The early, biologically inspired brain map models, however, were not suitable for practical data analysis. One of their inherent handicaps was that the resulting maps were *partitioned*. They were made up of small patches, between which the ordering jumped discontinuously and at random, and thus no *global order* over the whole map array was achieved. Although such partial ordering is commonplace in biology, many brain maps that represent *abstract features*, such as the tonotopic maps, the color maps, and the sonar-echo maps as reported in Suga and O'Neill (1979), Tunturi (1950, 1952), and Zeki (1980) respectively, are globally organized. Neither did these models *scale up*, i.e., they could not be used for large networks and high signal dimensionalities, in spite of highly increased computing power.

It is possible to state in retrospection that from the early neural models of self organization there was an important factor missing. It is a control factor or function, the amount of which depends

E-mail addresses: teuvo.koho@welho.com, teuvo.kohonen@aalto.fi.

0893-6080/$ – see front matter © 2012 Elsevier Ltd. All rights reserved.
doi:10.1016/j.neunet.2012.09.018

on local signal activity, but *which itself does not contribute to the signals*. Its only purpose is to *control the plasticity (modifiability by the signals) of selected subsets of neural connections in the network*. So, in the neural models, it will not be enough to control the activities of the nodes by the activities of other nodes through the links, i.e., the neural connections. One needs extra kinds of control factors that mediate information without mediating the activities. It is generally known that such an information is carried in the neural realms by, e.g., the chemical messenger molecules.

On the other hand, if the above neural and chemical functions are taken into account at least in abstract form, it is possible to scale up the self-organizing systems up to the capacity limits of the modern computers.

2. The classical vector quantization (VQ)

The implementation of optimally tuned feature-sensitive filters by competitive learning was actually demonstrated in abstract form much earlier in signal processing. I mean the *classical vector quantization (VQ)*, the basic idea of which was introduced (in scalar form) by Lloyd (1957), and (in vector form) by Forgy (1965). Actually the optimal quantization of a vector space dates back to 1850, called the *Dirichlet tessellation* in two- and three-dimensional spaces and the *Voronoi tessellation* in spaces of arbitrary dimensionality; cf. Dirichlet (1850) and Voronoi (1907). The VQ has since then become a standard technology in modern digital signal processing.

In vector quantization, the space of vector-valued input data, such as feature vectors, is partitioned into a finite number of contiguous regions, and each region is represented optimally by a single *model vector*, originally called the *codebook vector* in the VQ. (The latter term comes from digital signal transmission, where the VQ is used for the encoding and decoding of transmitted information.)

> *In an optimal partitioning, the codebook vectors are constructed such that the mean distance (in some metric) of an input data item from the best-matching codebook vector, called the winner, is minimized, i.e., the mean quantization error is minimized.*

For simplicity, the VQ is illustrated using the Euclidean distances only. Let the input data items constitute n-dimensional Euclidean vectors, denoted by $\mathbf{x}$. Let the codebook vectors be denoted by $\mathbf{m}_i$, indexed by subscript i. Let the subscript c be the index of a particular codebook vector $\mathbf{m}_c$, called the *winner*, namely, the one that has the smallest Euclidean distance from $\mathbf{x}$:

$$c = \underset{i}{\operatorname{argmin}}\{\|\mathbf{x} - \mathbf{m}_i\|\}. \tag{1}$$

If $p(\mathbf{x})$ is the probability density of $\mathbf{x}$, the *mean quantization error* E is defined as

$$E = \int_V \|\mathbf{x} - \mathbf{m}_c\|^2 p(\mathbf{x}) dV, \tag{2}$$

where dV is a volume differential of the data space V. The *objective function* E, being an *energy function*, can be minimized by a gradient-descent procedure. However, the problem is highly nonlinear; nonetheless, e.g., this author has shown that it converges to a local minimum; cf. Kohonen (1991).

If the set of the input data items is finite, a *batch computation method* is also feasible. It is called the *Linde–Buzo–Gray (LBG)* algorithm, cf. Linde, Buzo, and Gray (1980), but it was devised already by Forgy (1965). There exists a wealth of literature on the above VQ, which is also called "*k-means clustering*". For classical references, cf., e.g., Gersho (1979), Gray (1984), and Makhoul, Roucos, and Gis (1985).

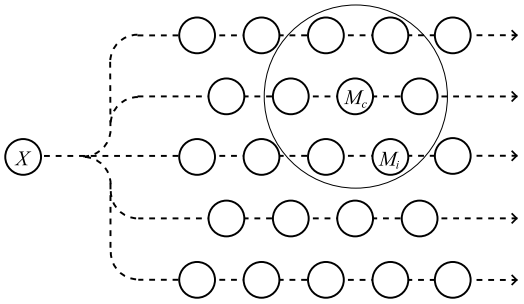

Fig. 1. Illustration of a self-organizing map. An input data item X is broadcast to a set of models M_i, of which M_c matches best with X. All models that lie in the neighborhood (larger circle) of M_c in the grid match better with X than with the rest.

3. The self-organizing map (SOM): general

3.1. Motivation of the SOM

Around 1981–82 this author introduced a new *nonlinearly projecting mapping*, called the *self-organizing map (SOM)*, which otherwise resembles the VQ, but in which, additionally, the *models* (corresponding to the codebook vectors in the VQ) become *spatially, globally ordered* (Kohonen, 1982a, 1982b, 1990, 2001).

The SOM models are associated with the *nodes* of a regular, usually two-dimensional *grid* (Fig. 1). The SOM algorithm constructs the models such that:

> *More similar models will be associated with nodes that are closer in the grid, whereas less similar models will be situated gradually farther away in the grid.*

It may be easier to understand the rather involved learning principles and mathematics of the SOM, if the central idea is first expressed in the following simple illustrative form:

> *Every input data item shall select the model that matches best with the input item, and this model, as well as a subset of its spatial neighbors in the grid, shall be modified for better matching.*

Like in the VQ, the modification is concentrated on a selected node that contains the winner model. On the other hand, since a whole spatial neighborhood in the grid around the winner is modified at a time, the degree of local ordering of the models in this neighborhood, due to a smoothing action, will be increased. The successive, different inputs cause corrections in different subsets of models. The local ordering actions will gradually be propagated over the grid. However, the real mathematical process is a bit more complicated than that.

The actual computations for producing the ordered set of the SOM models can be implemented by either of the following main types of algorithms: 1. The models in the original SOM algorithm are computed by a *recursive, stepwise approximation process* in which the input data items are applied to the algorithm one at a time, in a periodic or random sequence, for as many steps as it will be necessary to reach a reasonably stable state. 2. In the *batch-type process*, on the other hand, all of the input data items are applied to the algorithm as one batch, and all of the models are updated in a single concurrent operation. This batch process usually needs to be reiterated a few to a few dozen times, after which the models will usually be stabilized exactly. Even the time to reach an approximately stabilized state is an order of magnitude shorter than in the stepwise computation.

It should be emphasized that only the batch-learning version of the SOM is recommendable for practical applications, because it does not involve any learning-rate parameter, and its convergence is an order of magnitude faster and safer. The stepwise learning