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).

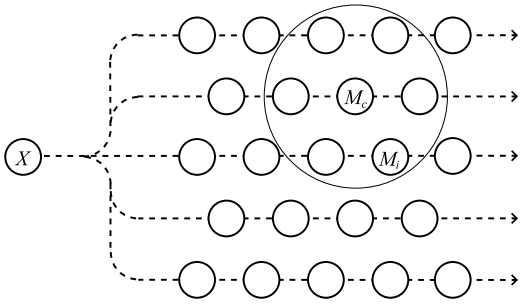

Fig. 1. Illustration of a self-organizing map. An input data item X is broadcast to a set of models M_i, of which M_c matches best with X. All models that lie in the neighborhood (larger circle) of M_c in the grid match better with X than with the rest.

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

rule, on the other hand, was originally set up only for theoretical reasons and to facilitate a comparison with the other self-organizing models. Moreover, it is not possible to use the stepwise learning with general metrics, but we will see that batch learning also solves this problem.

More detailed descriptions of the SOM algorithms will be given below.

Several commercial software packages as well as plenty of freeware on the SOM are available. This author strongly encourages the use of well-justified public-domain software packages. For instance, there exist two freeware packages developed by us, namely, the *SOM_PAK* (Kohonen, Hynninen, Kangas, & Laaksonen, 1996; SOM_PAK Team, 1990) and the *SOM Toolbox* (SOM Toolbox Team, 1999; Vesanto, Alhoniemi, Himberg, Kiviluoto, & Parviainen, 1999; Vesanto, Himberg, Alhoniemi, & Parhankangas, 1999), both downloadable from the Internet. Both packages contain auxiliary analytical procedures, and especially the SOM Toolbox, which makes use of the MATLAB functions, is provided with versatile graphics means.

Unlike in most biologically inspired map models, the topographic order in the SOM can always be materialized *globally* over the whole map.

The spatial order in the display facilitates a convenient and quick visual inspection of the similarity relationships of the input data as well as their clustering tendency, and comes in handy in the verification and validation of data samples. Moreover, with proper calibration of the models, the clustering and classification of the data become explicit.

The rest of this article concentrates on the SOM principles and applications. The SOM has been used extensively as a visualization tool in exploratory data analysis. It has had plenty of practical applications ranging from industrial process control and finance analyses to the management of very large document collections. New, promising applications exist in bioinformatics. The largest applications so far have been in the management and retrieval of textual documents, of which this paper contains two examples.

Many versions of the SOM algorithms have been suggested over the years. They are too numerous to be reviewed here; cf. the extensive bibliographies mentioned as Refs. (Kaski, Kangas, & Kohonen, 1998; Oja, Kaski, & Kohonen, 2003; Pöllä, Honkela, & Kohonen, 2009). See also the Discussion, Section 7.

3.2. Calibration of the SOM

If the input items fall in a finite number of *classes*, the different models can be made to correspond to these classes and to be provided with corresponding symbolic labels. This kind of *calibration* of the models can be made in two ways: 1. If the number of input items is sufficiently large, one can first study the distribution of matches that all of the input data items make with the various models. A particular model is labeled according to that class that occurs in the *majority of input samples that match with this model*. In the case of a tie, one may carry out, e.g., a majority voting over a larger neighborhood of the model. 2. If there is only a smaller number of input data items available so that the above majority voting makes no sense (e.g., there are too many ties, or there are no hits at some of the models), one can apply the so-called *k-nearest-neighbors (kNN)* method. For each model, those k input data items that are closest to it (in the metric applied in the construction of the SOM) are searched, and a majority voting over them is carried out to determine the most probable classification of the node. In the case of a tie, the value of k is increased until the tie is resolved. Usually k is selected to be on the order of half a dozen to a hundred, depending on the number of input data items and the size of the SOM array.

When a new, unknown input item is compared with all of the models, it will be identified with the best-matching model. The classification of the input item is then understood as that of the best-matching model.

3.3. On "matching by similarity"

There exist many versions of the SOM, which apply different definitions of "similarity". This property deserves first a short discussion. "Similarity" and "distance" are usually opposite concepts.

The *cognitive* meaning of similarity is a very vague one. For instance, one may talk of the similarity of two persons or two historical eras, although such a comparison is usually based on a subjective opinion.

If the same comparison should be implemented automatically, it can only be based on some very restricted analytical, say statistical attributes. The situation is much clearer, if we deal with concrete objects in science or technology, since we can then base the definition of dissimilarity on basic mathematical concepts of, say, *distance measures* between attribute vectors. The statistical figures are usually also expressed as real vectors, consisting of numerical results or other statistical indicators. Various kinds of spectra and other transformations can also be regarded as multidimensional vectors of their components.

The first problem in trying to compare such vectors is usually *different scaling* of their elements. For metric comparison, a simple remedy is to *normalize* the scales so that either the variances of the variables in the different dimensions, or their maxima and minima, respectively, become the same. After that, some standard distance measure, such as the Euclidean, or more generally, the Minkowski distance, etc. can be tried, the choice depending on the nature of the data. It has turned out that the Euclidean distance, with normalization, is already applicable to most practical studies, since the SOM is able to display even complex interdependencies of the variables in its display.

A natural measure of the similarity of vectorial items is in general some *inner product*. In the SOM research, the *dot product* is commonly used. This measure also complies better with the biological neural models than the Euclidean distance. However, the model vectors $\mathbf{m}_i$, for their comparison with the input $\mathbf{x}$, must be kept normalized to constant length all the time. If the vector dimensionality is high, and also the input vectors are normalized to constant length, the difference between SOMs based on the Euclidean distances and the dot products is insignificant. (For the construction of Euclidean and dot-product SOMs, cf. Sections 4.1 and 4.5, respectively.) On the other hand, if there are plenty of zero elements in the vectors, the computation of dot products is correspondingly faster. This property can be utilized effectively especially in the fast computation of document maps discussed at the end of this article.

Before proceeding further, it will be necessary to emphasize a basic fact. An *image*, often given as a set of pixels or other structural elements, will usually not be applicable as such as an input vector. The natural variations in the images, such as translations, rotations, variations of size, etc., as well as variations due to different lighting conditions are usually so wide that a direct comparison of the objects on the basis of their appearances does not make any sense. Instead, the classification of natural items shall be based on the extraction and classification of their characteristic *features* which must be as *invariant* as possible. Features of this type may consist of color spectrograms, expansions of the images in Fourier transforms, wavelets, principal components, or eigenvectors of some image operators, etc. If one can describe the input objects by a restricted set of *invariant features*, the dimensionality of the input representations, and the computing load are reduced drastically.

A special kind of dissimilarity or distance measure is applied in an SOM that is called the *Adaptive-Subspace SOM (ASSOM)*, cf. Kohonen (1995, 1996, 2001) and Kohonen, Kaski, and Lappalainen (1997). In it, certain elementary systems are associated with the nodes, and these systems develop into specific filters that respond invariantly to some class (e.g., translation-invariant, rotation-invariant, or scale-invariant) of *local features*. Their parameters

are determined by adaptation to transforms of the same input patterns (whatever they are) when these patterns occur in various elementary transformations of observations. Mathematically each of these systems represents a *subspace* of the signal space, and is described by a few *basis vectors*. The distance of an input vector from the subspace is defined as its *orthogonal projection on the orthogonal complement of this subspace*, cf. Kohonen (2001, p. 6). So, if the length of the distance from the subspace is zero, the input vector is expressible as a linear combination of the basis vectors; but in general, the length of that distance constitutes the error. In the adaptive learning, the basis vectors are *rotated* to decrease the error, i.e., the distance from the due subspace.

The selection of a characteristic set of features and their automatic extraction from the primary observations must often be based on heuristic rules. In biology, various feature detectors have been developed in a very long course of evolution.

For more complex comparisons one may also look for other kinds of features to be used as the vector elements. For instance, in *text analysis*, complete *documents* can be distinguished from each other based on their word statistics, i.e., *word histograms*, whereupon very careful attention must be paid to the relative occurrence of the words in different texts; cf. Salton and McGill (1983). So, the elements of the histogram, corresponding to the various words, must be *weighted* by multiplicative factors derived from the relative occurrences of the words. In addition to using the statistical *entropy* of a word, the words in histograms can be weighted (and thus, rare and very common words can be ignored) based on their *inverse document frequency (IDF)*. The "document frequency" means in how many documents in a text corpus a particular word occurs, and IDF is the inverse of this figure. With proper weighting, the word histograms, which constitute the feature vectors, can be restricted to, say, some hundreds of dimensions.

The *strings of symbols* constitute another common type of variables. Except in text, string variables occur, e.g., in bioinformatics and organic chemistry: in genetic codes, sequences of atoms in macromolecules etc.: cf., e.g., Kohonen and Somervuo (2002) and Oja, Somervuo, Kaski, and Kohonen (2003). Normally the strings are of very different length. Some kind of *edit distance*, i.e., the number of elementary editing operations needed to transform one string into the other is a very effective definition of the distance between string variables. These operations must normally be weighted based on the statistics of the various errors. For very long strings, such as the protein sequences, some heuristic shortcut computations such as those applied in the wide-spread FASTA method (Pearson, 1999; Pearson & Lipman, 1988) may be necessary. Such distance measures have often been precomputed in the databases.

There are other, more abstract kinds of similarity measures. One of them is the *contextual similarity of words*. Consider a word in a text, within the context of its neighboring words. If each word in the vocabulary is represented by a random code, the mutual correlations between the representations of the words remain very small. However, the measure of similarity of two *local contexts*, e.g., triplets of three successive words in the text, then ensues from the occurrence of the same random codes in identical positions in the triplets. Analyses of the semantic values of words can be based on contextual-similarity studies, and very deep linguistic conclusions can be drawn from such analyses, as demonstrated in Kohonen and Xing (2011).

A very special measure of similarity is *functional similarity*, which may mean, e.g., the following. Consider, e.g., a set of signal filters described by a finite number of parameters; otherwise the structure of the filters is identical. For the same input signal, the various filters then usually produce different responses. Consider, e.g., filters used as *predictors*. An elementary prediction error, relating to a given input, is the absolute value of the difference of the prediction and the true future signal value. For given statistics of input signals, the *distance between two filters* (i.e. that of the sets of their parameters) may be defined as the root-mean-square of their due prediction errors, as suggested by Lampinen and Oja (1989). Note that two sets of filter parameters may look quite different, although the differences of the prediction errors of the respective filters may be small. One may find an indefinite number of different kinds of functional similarities in practical applications.

An important task is to compare *dynamic phenomena*. This becomes possible, if the models are made to represent *dynamic states*. A very interesting discussion of dynamic SOMs has been presented by Hammer, Micheli, Sperduti, and Strickert (2004).

3.4. Levels of abstraction in modeling

Biological modeling. The earliest SOM models, tending to replicate the detailed neural-network structures, were intended for the description and explanation of the creation of brain maps. A very modern approach to the formation of feature-sensitive cells in the visual cortex has been made by Miikkulainen, Bednar, Choe, and Sirosh (2005). One might also mention a recent version of self-organizing projections (Kohonen, 2005, 2006) in which a global order can be achieved. With it, a model of diffusion of the plasticity-controlling molecules is involved.

Mathematical abstraction. In abstract mathematical models, the details of network connections, synaptic plasticity, and chemical control are ignored, and the dynamics of the activities is represented by matrix-vector functions and computations. Using these abstractions of neural networks, one has been able to scale up many problems so that the SOM has become a practical data-analysis tool.

3.5. Network architectures

The two-dimensional organization of the models is usually already effective for the approximation of similarity relations of high-dimensional data items, like in some earlier methods called the *multidimensional scaling (MDS)* (Kruskal & Wish, 1978; Sammon, 1969). One should notice, however, that in the MDS methods, *every data item* must be displayed geometrically, whereas the SOM uses only a relatively small set of models that it displays on a regular grid. In this way, plenty of computations will be saved.

For special purposes, grids with dimensionality higher than two may be used, too. A simple rank ordering can also be made to take place along a one-dimensional grid which, however, will not be able to display any more general topological relationships between the data items.

Regular arrays. Most applications of the SOM are based on regular arrays of nodes. Sometimes one uses rectangular arrays of nodes for simplicity. However, the hexagonal arrays are visually much more illustrative and accurate, and are recommended. Whatever regular architectures are used, it is advisable to select the lengths of the horizontal and vertical dimensions of the array to correspond to the lengths of the two largest principal components (i.e., those with the highest eigenvalues of the input correlation matrix), because then the SOM complies better with the low-order signal statistics. The oblong regular arrays have the advantage over the square ones of guaranteeing faster and safer convergence in learning.

Cyclic arrays. The usual SOM models, after learning, exhibit familiar *border effects*: the spacings of the neighboring models are not as regular near the borders as in the middle of the SOM. For this reason, some researchers have suggested that the network should be made end-around cyclic, either toroidal or spherical. The latter choice has been made by the BLOSSOM Team (2005). A cyclic

network is suitable, if the data itself has a simple cyclic structure. The cyclic structure may also be advantageous, if the SOM is used, e.g., in process control to look up for optimal corrections from the SOM. In cyclic maps, namely, there are no discontinuities due to the network borders. Even in cyclic networks, however, one can discern similar irregularities in the spacing of the models, if some models are located near strong extremities of the input distribution. So, the irregularities seem to be due to the *extremities of the data distribution* and not so much to the network borders.

Growing networks. A few researches have suggested that the *structure* of the network ought to be made to comply better with the input data. So, the network structure ought to be made *variable*, e.g., it should be allowed to grow in the directions of the input data distribution; cf., e.g., Fritzke (1994). If this is made, one may be able to see hierarchical structures of the data emerging explicitly. Nonetheless there is some arbitrariness in the definition of the conditions that have been suggested to define the directions of growing. Arbitrary branching criteria define arbitrary structures in the network. The visual display is nonetheless better in regular arrays, and the clustering tendencies can be illustrated uniquely by the *U-matrix* techniques, as devised by Ultsch (1993).

3.6. A frequently asked question

One of the most frequently asked questions concerning the structure of the SOM is how many nodes one needs in the array. Maybe it is necessary to state first that the main virtue of the SOM is in the *visualization* of the data space, whereupon the *clustering structures* ought to become visible. There is no sense in using the SOM for very small data sets, because there exist much better data analysis methods for them.

So, assume that we have enough data items in order to visualize the data space with sufficient accuracy. Then it should be realized that the SOM is also a *quantizing method*, and has a limited *resolution* to show the cluster structures. Sometimes the data set may contain only a few clusters, whereupon a coarse resolution is sufficient. However, if one suspects that there are interesting fine structures in the data, then a larger array is needed for sufficient resolution.

Typical SOM arrays range from a few dozen to a few hundred nodes, and the relation of the horizontal and vertical dimensions of the array ought to comply at least roughly with the relation of the two largest principal components of the input data, respectively. It is not possible to guess or estimate the exact size of the array beforehand. It must be determined by the trial-and-error method, after seeing the quality of the first guess.

However, it is also necessary to realize that the SOM is often used as a kind of *histogram* on which one displays the number of input data items that is mapped into each of the nodes. This histogram can be visualized by shades of gray or by pseudocolors. The statistical accuracy of such a histogram depends on how many input items are mapped on the average per node. A very coarse rule-of-thumb may be that about 50 items per node on the average should be sufficient, otherwise the resolution is limited by the sparsity of data. So a compromise must be made between resolution and statistical accuracy. These aspects should be taken into account especially in statistical studies, where only a limited number of samples are available.

On the other hand, the SOM may be at its best in the visualization of industrial processes, where unlimited amounts of measurements can be recorded. In the latter case one may try to use as big an array as one is able to compute, and with the modern computers, even personal ones, it is possible to deal with thousands of nodes in the SOM array.

4. Two main SOM algorithms

4.1. The original, stepwise recursive SOM algorithm

The original formulation of the SOM algorithm resembles a gradient-descent procedure. It must be emphasized, however, that this version of the algorithm was introduced heuristically, when trying to materialize the general learning principle given in Section 3.1. This basic form has not yet been shown to be derivable from any energy function. An approximative and purely formal, but not very strict derivation ensues from the *stochastic approximation* method (Robbins & Monro, 1951); it was applied in Kohonen (2001, pp. 146–147). Notwithstanding, successful derivations of the convergence of slightly modified SOM algorithms, based on correspondingly modified objective functions, have been presented by Heskes and Kappen (1993), as well as Kohonen (1991) and Luttrell (1992).

The convergence of the original SOM algorithm to the globally ordered state has been proven mathematically in some simple low-dimensional cases: see Cottrell and Fort (1987). On the other hand, Ritter, Martinetz, and Schulten (1992) have presented a proof that a *local order* can be reached for more general dimensionalities of the vectors, if the distribution of the input vectors is discrete-valued.

Below, only the original form of the SOM is shown, because it is computationally the simplest and lightest, and in practice, almost without exception, it will produce useful results for high-dimensional input vectors, too. The correctness of the ordering result, and the quality of the produced maps can be analyzed mathematically. The theoretical foundations of the SOM have been discussed thoroughly by Cottrell, Fort, and Pagés (1997).

Consider again Fig. 1. Let the input data items this time constitute a sequence $\{\mathbf{x}(t)\}$ of real n-dimensional Euclidean vectors $\mathbf{x}$, where t, an integer, signifies a step in the sequence. Let $\{\mathbf{m}_i(t)\}$ be another sequence of n-dimensional real vectors that represent the successively computed approximations of model $\mathbf{m}_i$. Here i is the spatial index of the grid node with which $\mathbf{m}_i$ is associated. The original SOM algorithm assumes that the following process converges and produces the wanted ordered values for the models:

$$\mathbf{m}_i(t+1) = \mathbf{m}_i(t) + h_{ci}(t)[\mathbf{x}(t) - \mathbf{m}_i(t)], \tag{3}$$

where $h_{ci}(t)$ is called the *neighborhood function*. This function resembles the kernel that is applied in usual smoothing processes. The subscript c is the index of a particular node (winner) in the grid, namely, the one with the model $\mathbf{m}_c(t)$ that has the smallest Euclidean distance from $\mathbf{x}(t)$:

$$c = \operatorname*{argmin}_i \{\|\mathbf{x}(t) - \mathbf{m}_i(t)\|\}. \tag{4}$$

Eqs. (3) and (4) can be illustrated as defining a recursive step where first the input data item $\mathbf{x}(t)$ defines or selects the best-matching model (winner) in the grid according to Eq. (4). Then, according to Eq. (3), the model at this node as well as at its *spatial neighbors* in the grid are modified. The modifications always take place in such a direction that the modified models will match better with the input.

The rates of the modifications at different nodes depend on the mathematical form of the function $h_{ci}(t)$. A much applied choice for the neighborhood function $h_{ci}(t)$ is

$$h_{ci}(t) = \alpha(t)\ \exp[-\text{sqdist}(c, i)/2\sigma^2(t)], \tag{5}$$

where $\alpha(t)$ is a monotonically (e.g., hyperbolically, exponentially, or piecewise linearly) decreasing scalar function of t, sqdist(c, i) is the square of the geometric distance between the nodes c and i in the grid, and $\sigma(t)$ is another monotonically decreasing function of t, respectively. The true mathematical form of $\sigma(t)$ is not crucial, as long as its value is fairly large in the beginning of the process,

say, on the order of half of the diameter of the grid, whereafter it is gradually reduced to a fraction of it in about 1000 steps. The topological order is developed during this period. On the other hand, after this initial phase of *rough ordering*, the *final convergence* to nearly optimal values of the models takes, say, an order of magnitude more steps. For a sufficient statistical accuracy, every model must be updated sufficiently often. *However, we must give a warning: the final value of σ shall not go to zero, because otherwise the process loses its ordering power. It should always remain, say, above half of the grid spacing.* In very large SOM grids, the final value of σ may be on the order of five per cent of the shorter side of the grid.

There are also other possible choices for the mathematical form of $h_{ci}(t)$. One of them is very simple, in which we have $h_{ci} = 1$ up to a certain radius from the winner, and zero otherwise.

The neighborhood function has the most central role in self organization. However, attempts to implement it by pure neural components have not been successful. On the contrary, in biological modeling, it seems that it is best materialized by the diffusion of some chemical control agents from places where the cell activity is high; cf., e.g., Kohonen (2006).

The reader is advised to use some of the ready-made complete software packages, since there are many parameters and plenty of other aspects to be taken into account in the definition of a good learning process. Also the following batch computation procedure is to be preferred in practice, especially since it contains fewer parameters than the stepwise recursive algorithm, and converges much faster.

4.2. Stable state of the learning process

Assuming that the convergence to some stable state of the SOM is true, we require that the expectation values of $\mathbf{m}_i(t+1)$ and $\mathbf{m}_i(t)$ for $t \to \infty$ must be equal, while h_{ci} is nonzero, where $c = c(\mathbf{x}(t))$ is the index of the winner node for input $\mathbf{x}(t)$. In other words we must have

$$\forall i, \quad E_t\{h_{ci}(\mathbf{x}(t) - \mathbf{m}_i(t))\} = 0. \tag{6}$$

Here E_t is the mathematical expectation value operator over t. In the assumed asymptotic state, for $t \to \infty$, the $\mathbf{m}_i(t)$ are independent of t and are denoted by $\mathbf{m}_i^*$. If the expectation values $E_t(.)$ are written, for $t \to \infty$, as $(1/t)\sum_t(.)$, we can write

$$\mathbf{m}_i^* = \frac{\sum_t h_{ci}\mathbf{x}(t)}{\sum_t h_{ci}}. \tag{7}$$

This, however, is still an *implicit* expression, since c depends on $\mathbf{x}(t)$ and the $\mathbf{m}_i$. Nonetheless, Eq. (7) shall be used for the motivation of the iterative solution for the $\mathbf{m}_i$, known as the *batch computation of the SOM ("Batch Map")*.

Then it is useful to notice that for the different nodes i, the same addends occur a great number of times. Therefore it is advisable, especially with very large SOMs, to first compute the mean $\mathbf{x}_{m,j}$ of all of the the $\mathbf{x}(t)$ that are closest to model $\mathbf{m}_j$ in the data space, and then weight it by the number n_j of the samples in this subset and by h_{ji}. Then we obtain

$$\mathbf{m}_i^* = \frac{\sum_j n_j h_{ji}\mathbf{x}_{m,j}}{\sum_j n_j h_{ji}}, \tag{8}$$

where the notation $\mathbf{x}_{m,j}$ is used to denote the mean of the inputs that are closest to the model $\mathbf{m}_j$, and n_j is the number of those inputs.

Eq. (7) or Eq. (8) will be used later in the derivation of the iterative batch-computation algorithm.

4.3. Initialization of the models

A special question concerns the selection of the *initial values* for the $\mathbf{m}_i$. It has been demonstrated by Kohonen (2001) that they can be selected even as *random vectors*, whereas much faster ordering and convergence follow if the initial values are selected as a *regular, two-dimensional sequence of vectors taken along a hyperplane spanned by the two largest principal components of* $\mathbf{x}$ (i.e., principal components associated with the two highest eigenvalues); cf. Kohonen (2001). This method is called *linear initialization*.

The initialization of the models as random vectors was originally used only to demonstrate the capability of the SOM to become ordered, starting from an arbitrary initial state. In practical applications one expects to achieve the final ordering as quickly as possible, so the selection of a good initial state may speed up the convergence of the algorithms by orders of magnitude.

In the next subsection we shall discuss the batch computation of the SOM, which also applies to general distance measures of the input vectors. If the input vectors were Euclidean, the principal-component method is recommended for the initialization. For general distance measures the random initialization is always possible; the optimization of initialization for general metrics, however, is very tricky; cf. e.g., Kohonen and Somervuo (2002), and cannot be discussed at length here.

For the initialization of SOMs that are based on functional similarity, one may have to resort to random initialization of the model parameters.

4.4. The batch computation of the SOM

In the continuation we shall concentrate on the batch computation of the SOM, because it is faster and safer than the stepwise recursive algorithm, and can also be generalized for nonvectorial data.

Consider Fig. 2, where a two-dimensional hexagonal array of nodes, depicted by the circles, is shown. With each node i, a model $\mathbf{m}_i$ is associated. Also a list, containing copies of certain input vectors $\mathbf{x}(t)$, is associated with each node.

Like in the stepwise recursive algorithm, the initial values of the $\mathbf{m}_i$, in principle, may be selected as random vectors, preferably from the domain of the input vectors. However, a better strategy, in the case of the Euclidean metric, is to take for the $\mathbf{m}_i$ a regular two-dimensional sequence of values picked up from the two-dimensional hyperplane, spanned by the two largest principal components of $\mathbf{x}$.

Then consider the set of input data vectors $\{\mathbf{x}(t)\}$, where t is an integer-valued index of a vector. Compare each $\mathbf{x}(t)$ with all of the models and make a copy of $\mathbf{x}(t)$ into the sublist associated with the node, the model vector of which matches best with $\mathbf{x}(t)$ in the Euclidean metric.

In this illustrative example we assume a neighborhood function that has the value 1 in a *neighborhood set* N_i of nodes, consisting of the nodes up to a certain distance from node i, and is equal to zero otherwise.

Since according to Eqs. (7) and (8) the equilibrium value of every model must now be the mean of the $\mathbf{x}(t)$ falling into its neighborhood set N_i, we want to approach this equilibrium state by the following strategy. We compute the *mean* of the $\mathbf{x}(t)$ over N_i, that is, of all the $\mathbf{x}(t)$ that have been copied into the union of all of the sublists in N_i. A similar mean is computed for every node i, i.e. over the neighborhoods around all of the nodes. Updating of the $\mathbf{m}_i$ then means that the old values of the $\mathbf{m}_i$ are replaced by the respective means, *in one concurrent computing operation over all nodes of the grid*. This concludes one updating cycle.

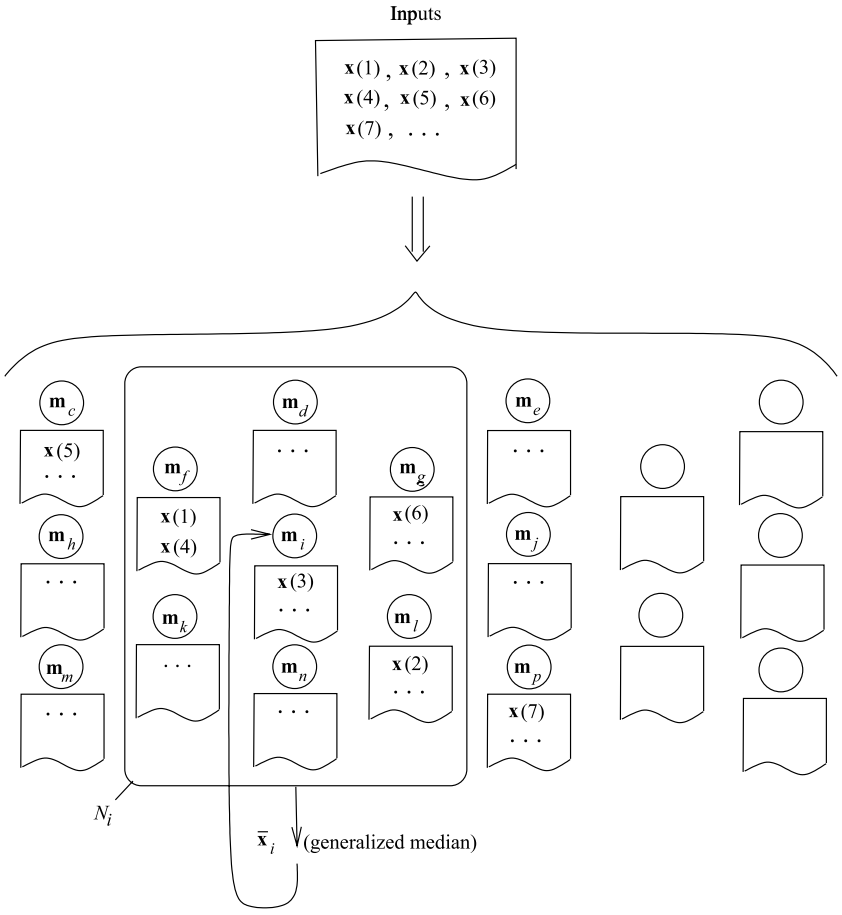

Fig. 2. Illustration of one cycle in the batch process. The input data items $\mathbf{x}(t)$ are first distributed into sublists associated with their best-matching models, and then the new values of the models are determined as *means*, or more generally, as *generalized medians* (as written in the figure) over the neighborhood sets of sublists.

In Fig. 2, however, there appears a new term "generalized median", written below the set line. In the above discussion it should be replaced by the word "mean", but because Fig. 2 is also used for to explain the self-organization in the case of a general distance measure, the term "generalized median" has been used in it and has to be explained next.

The batch-computing method, as several times mentioned, can be generalized for nonvectorial data, e.g., strings of symbols, if a *generalized median* over the samples $\mathbf{x}(t)$ can be defined. *This kind of median of a set of data items is defined as the item that has the smallest sum of distances from the other items in the set.* This concept was originally introduced by the author; cf. Kohonen (1985), but it can also be found from Kohonen (2001). Where we earlier used the mean over the union of the sublists in the neighborhood set N_i, we shall now use the generalized median in the same role. Such SOMs have been constructed for protein sequences by Kohonen and Somervuo (2002) and for virus-type DNA in Oja, Somervuo et al. (2003); Oja, Sperber, Blomberg, and Kaski (2004, 2005).

This updating cycle is reiterated, always first clearing all lists and thereafter distributing new copies of the input vectors under those nodes, the (updated) models of which match best with the new input vectors. The updated model values, sooner or later, become steady and are no longer changed in continued iterations, whereupon the training can be stopped.

A process that complies even better with the stepwise recursive learning is obtained if the means are formed as *weighted averages*, where the weights are related to each other like the h_{ci}. Here c is the index of the node, the model of which is updated, and i stands for the indices of the nodes in its neighborhood.

A discussion of the convergence of the batch-computing method has been presented by Cheng (1997).

4.5. Dot-product maps

For metric vectors, a practical computation of the SOM is based on their *dot products*. For Euclidean vectors this method is particularly advantageous, if there are plenty of zero elements in the vectors, because they are skipped in the evaluation of similarities. However, the model vectors $\mathbf{m}_i$, for their comparison with the input $\mathbf{x}$, must be kept normalized to constant length all the time. Instead of Eq. (1), the index of the winner location is now defined by

$$c = \underset{i}{\operatorname{argmax}}\{\operatorname{dot}(\mathbf{x}, \mathbf{m}_i)\}. \tag{9}$$

Since the computation of the SOM is in practice carried out by the batch algorithm, the mapping of all of the input items onto the respective winner nodes (i.e., the associated lists) is made in a similar way as described before. The only modification is modification of the definition of the "generalized median". There may exist several possibilities for doing that, but in our applications we have successfully used the following rule, which is applicable at least if the number of items mapped on each node is high on the average:

> *In the "dot-product SOMs", the "generalized median" of a set of items is identified with that item that has the minimum sum of dot products with all of the other items.*

The normalization of the $\mathbf{m}_i$ to constant length shall be made after each iteration cycle.

5. Applications of the SOM

5.1. Main application areas of the SOM

Before looking into the details, one may be interested in knowing the justification of the SOM method. Briefly, by the end of the year 2005 we had documented 7768 scientific publications: cf. Kaski, Kangas et al. (1998), Oja, Kaski et al. (2003) and Pöllä et al. (2009) that analyze, develop, or apply the SOM. The following short list gives the main application areas:

1. Statistical methods at large
   (a) exploratory data analysis
   (b) statistical analysis and organization of texts
2. Industrial analyses, control, and telecommunications: Kohonen, Oja, Simula, Visa, and Kangas (1996)
3. Biomedical analyses and applications at large
4. Financial applications: Deboeck and Kohonen (1998)

In addition to these, one may mention a few specific applications, e.g., profiling of the behavior of criminals, categorization of galaxies (Naim, Ratnatunga, & Griffiths, 1997), categorization of real estates, etc.

It is not possible to give a full account of the theory and different versions of the SOM, or applications of the SOM in this article. We can only refer to the above lists of 7768 SOM publications (today, their number is over 10,000), and to more than ten textbooks, monographs, or edited books, e.g. Allinson, Yin, Allinson and Slack (2001), Kohonen (1989, 2001), Miikkulainen (1993), Obermayer and Sejnowski (2001), Oja and Kaski (1999), Ritter et al. (1992), Seiffert and Jain (2002), Tokutaka, Kishida, and Fujimura (1999), Tokutaka, Ookita, and Fujimura (2007) and Van Hulle (2000), and a great number of Ph.D. Theses.

Two special issues of this journal have been dedicated to the SOM: The 2002 Special Issue with the subtitle "New Developments in Self-Organizing Maps", *Neural Networks*, Vol. 1, Numbers 8–9, October/November 2002, and the 2006 Special Issue "Advances in Self-Organizing Maps—WSOM'05", *Neural Networks*, Vol. 1, Numbers 6–7, July/August 2006. Moreover, the journal *Neurocomputing* has published a special SOM issue in Vol. 21, Numbers 1–3, October 1998.

A series of meetings named the *WSOM (Workshop on Self-Organizing Maps)* has been in progress since 1997. They have been organized in the following venues: Otaniemi, Finland (1997 and 1999); Lincoln, UK (2001), Kitakyushu, Japan (2003); Paris, France (2005); Bielefeld, Germany (2007); St. Augustine, FL, USA (2009), Espoo, Finland (2011), and Santiago, Chile (2012).

5.2. Hints for the construction of very large SOMs

With very large maps, both winner-search and updating are time-consuming operations. Notice that with large arrays, the initial radii of the neighborhoods are usually very large, too. Even the final radius may be on the order of a dozen grid spacings. Then, if the initial values of the models are close to the final state, the convergence of the SOM can be made at least an order of magnitude faster.

A good estimate of the initial values of a large map can be obtained by starting with a smaller map, initializing it properly, e.g., by the principal-component method (linear initialization), letting it to get stabilized approximately, and then adding new, interstitial nodes to the SOM array. The initial values of the models associated with the interstitial nodes can be interpolated on the basis of those of the smaller array that have already converged approximately. After that, the larger map must be let to get converged carefully in a longer training process.

Especially in word histograms regarded as vectors, there are usually plenty of zero elements. If the dot-product similarity measure is applied, the zero elements do not contribute to its calculation. It is possible to pretabulate the indices of the nonzero elements for each input vector, and thereafter consider only those elements in computing the dot products.

The searching of the winner can in general be accelerated by orders of magnitude by storing the addresses of the old winner locations together with the corresponding training data. The pointers to the old winners can be utilized during the next training cycle, confining the searching for the new winner only to the neighborhood of the old winner by following the stored pointers, after which the pointer to the winner is updated. If a smaller map is computed first, approximate pointers to a much larger map can also be obtained by the interpolation method explained above.

Eq. (8) allows for a very efficient implementation of parallel computing, in which extra memory need not be reserved for the computed new values of the $\mathbf{m}_i$. After the new pointers to the new winners have been computed, the previous values of the model vectors are not needed any longer. They can be replaced by the updated means $\mathbf{x}_{m,j}$, and the new values of the model vectors can be computed directly from the above equation.

Moreover, the winner search can be partly parallelized by dividing the data into different processors in a shared-memory computer.

For a more detailed explanation, see Kohonen et al. (2000) or Kohonen (2001, p. 294).

With a high dimensionality of the vectors, the memory requirements can further be reduced significantly by using a low representation accuracy of the vector elements. Notice that the truncation errors are comparable to white noise, which is statistically smoothed out when computing vectorial distances of high-dimensional vectors. In our largest SOM we have used only eight-bit accuracy for the vector elements.

5.3. SOMs of document collections

It is possible to form similarity graphs of *text documents* by the SOM principle, when *models of word collections* of the documents are used as features; cf. Lin, Soergel, and Marchionini (1991), Merkl (1995), Merkl, Tjoa, and Kappel (1994), Scholtes (1991), and Zavrel (1995). The simplest features consist of weighted histograms of the words, regarded as real vectors, but usually some dimensionality reduction of the very-high dimensional histograms must be carried out.

One of the new ideas, called the WEBSOM, was to construct first a *word-category SOM* based on the *contextual similarity of words* (cf. Section 3.3). Then, the words of a free natural text were clustered onto the grid points of such a word-category SOM to form a *word category histogram*. This histogram provided the input features to an SOM of documents; cf. Kaski, Honkela, Lagus, and Kohonen (1998).

We abandoned this idea, however, for two reasons. First, the construction of a word-category SOM was not uniquely defined. Second, we found that another dimensionality-reduction principle, based on a *random projection* of the weighted word histogram, together with some computational tricks (discussed below) turned out to be orders of magnitude more effective. The generic name "WEBSOM", nonetheless, was later used also for this improved version of a document SOM.

5.3.1. Clustering of documents by classes

The text corpus used in this first example consisted of documents collected from four different fields of applications, and it was

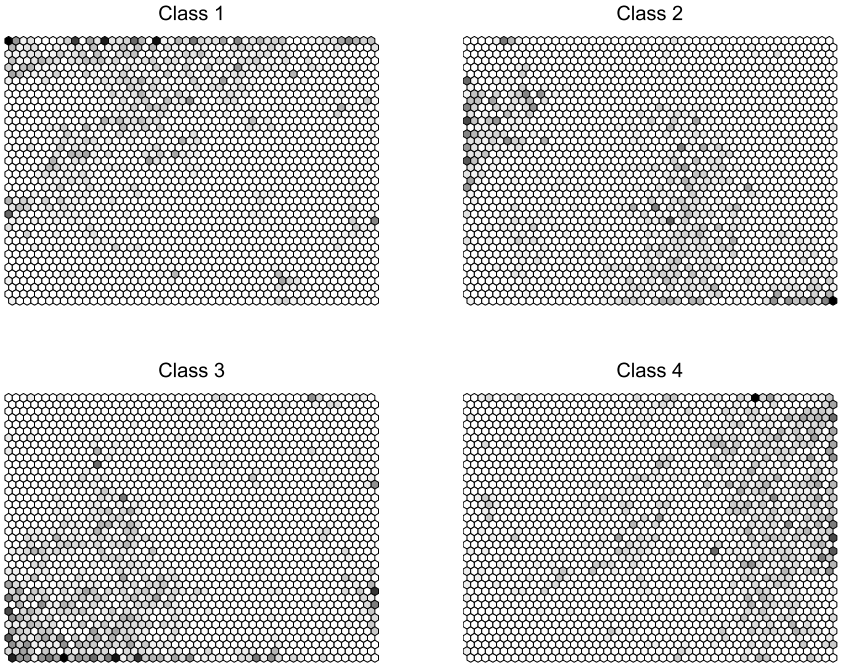

Fig. 3. Mapping of the four Reuters document classes onto the SOM. The densities of the "hits" are shown by shades of gray.

compiled by the Reuters corporation. No original documents were made available, but Lewis, Yang, Rose, and Li (2004) have preprocessed the textual data, removing the stop words, and reducing the words into their stems; in other words, our work was based on the reduced word histograms. Salojärvi from our laboratory selected a 4000-document subset from this preprocessed corpus, restricting only to such articles that were classified into one of the following categories:

1. Corporate-industrial.
2. Economics and economic indicators.
3. Government and social.
4. Securities and commodities trading and markets.

There were 1000 documents in each class. Salojärvi then picked up those 1960 words to the histograms that appeared at least 200 times in the selected texts. In order to carry out statistically independent experiments, a small subset of documents was set aside for testing. The 1960-dimensional word histograms were weighted by entropy-type factors as explained by Manning and Schütze (1999). Using the weighted word histograms of the rest of the 4000 documents as input, a 2000-node SOM was constructed.

Fig. 3 shows the four distributions of the hits on the SOM, when the input items from each of the four classes were applied separately to the SOM. It is clearly discernible that the map is ordered, i.e., the four classes of documents are segregated to a reasonable accuracy, and the classes 1, 3, and 4 are even singly connected, in spite of all of them dealing with closely related topics.

5.3.2. The SOM of the Encyclopaedia Britannica

In this example we shall emphasize the benefits of SOM methods in massive problems such as the organization of and searching from document corpora. For documents, the distributions of words and other terms (word combinations) constitute a possible set of characteristic features. It may be necessary to forestall eventual criticisms by stating that although semantic analyses of the texts are more accurate for the classification of the true textual contents, we could not afford such methods. Our objective was to deal with really large collections of literary documents, whereupon the standard contemporary computers, albeit rather advanced ones, already had to operate at the limits of their capabilities. Our choice for the features then demonstrated the power of (weighted) word histograms as features, combined with the SOM, in the exploration of large text collections.

From the electronic version of Encyclopaedia Britannica, 68,000 articles were picked up. In addition, 43,000 textual items such as summaries, updates, and other miscellaneous material were collected. Very long articles were split into several sections, resulting in a total of 115,000 documents. An average document contained 490 words. An SOM for this material was then computed by Kohonen et al. (2000).

These documents were first preprocessed to remove the HTML markups, links, and images. Inflected word forms were converted to their base forms using a standard morphological analyzer; otherwise, every inflected form would have been regarded as a different word. The words were then weighted by the *inverse document frequency (IDF)*. By ignoring words that had a very low IDF value (such as stopwords), the size of the finally accepted vocabulary was 39,058 words. This would then have been the dimensionality of the input vectors to the SOM, but it would still have meant very heavy computations.

In text analysis, the so-called *latent semantic indexing (LSI)* (Deerwester, Dumais, Furnas, & Landauer, 1990) is used for the reduction of the dimensionalities of the word histograms. Kaski (1998) of our group found out that a certain *random projection*, as introduced in Ritter and Kohonen (1989), of the word histograms is essentially as effective, but is computationally very much lighter.

The dimensionality of the feature space was reduced by forming a random projection of each of the 39,058-dimensional feature vectors onto a 1000-dimensional space. To that end, every weighted-histogram vector was multiplied by the same 39,058-by-1000 matrix, the rows of which were permanently set to some normalized random-vector values. It has been found that the relative mutual distances between the projected low-dimensional vectors are approximately the same as those of the original high-dimensional vectors.

The size of the SOM grid in this application was 12,096 nodes. Several speedups in the SOM computation were applied: e.g., by first computing an SOM with a much smaller grid, then introducing interstitial nodes to form a larger grid, interpolating values for their

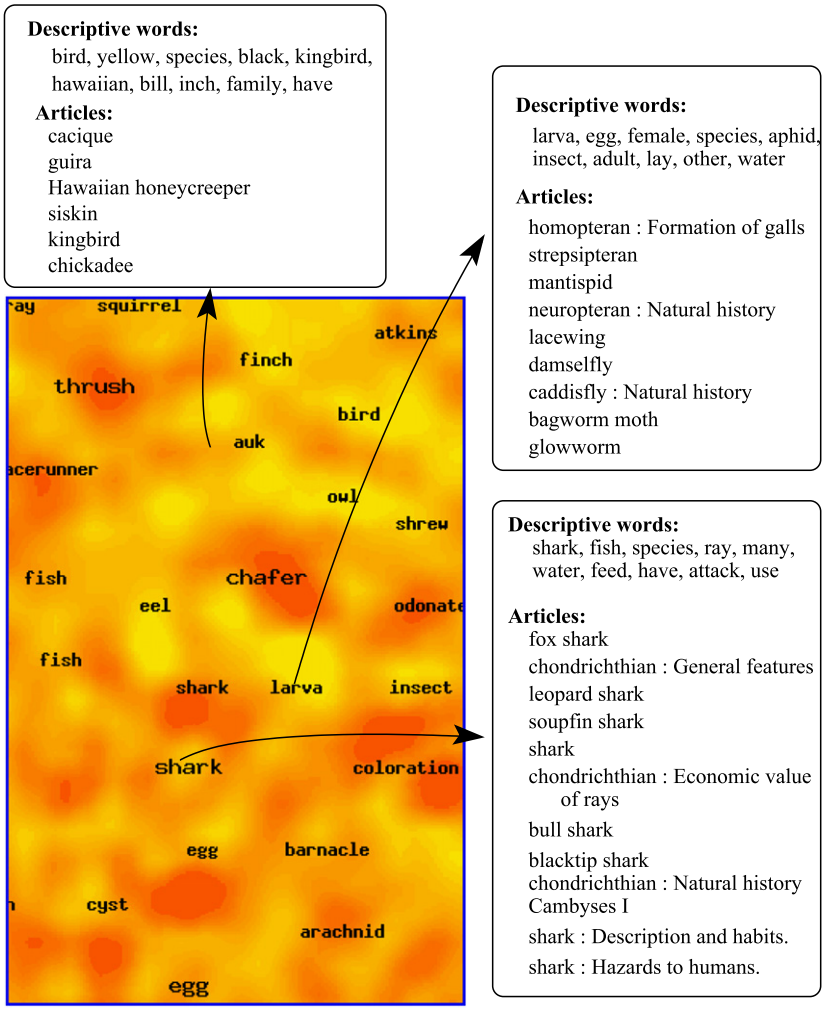

Fig. 4. A close-up view of the map of Encyclopaedia Britannica articles. When the user clicks a node in the map region with the label "shark", he or she obtains the listing of articles on sharks. The node "larva" contains articles on insects and larvae, and a node in the middle of the area of birds contains articles on birds. A bit more remote areas (not shown here) contain articles on whales and dolphins, and areas beyond them describe whaling, harpoons, and Eskimos, which proves that the topics change smoothly and continuously on the map. By clicking the searched titles, the complete articles can be read out.

model vectors, and finally carrying out a shorter fine tuning with the larger grid.

When an SOM has been constructed and the models have been fixed, each document will be mapped into one and only one node, namely, the one whose model vector matches best with the feature vector of the given document. One can store the original documents in a separate buffer area in the computer memory system, and associate with each SOM model only an address pointer to the respective document. Usually several documents will be mapped into each node. There exist now several modes of use of this system.

One simple mode of use of the SOM is *browsing*. There is usually a provision with the graphics such that when a particular node in the display is "clicked", all of the address pointers associated with that node are activated, and the titles of the corresponding documents and eventually some additional information such as descriptive words are written out. After that, the wanted document can be read out in full.

Fig. 4 shows a close-up view of a part of the large SOM. The keywords written into the vicinity of some locations on the map have been determined by an automatic annotation procedure explained in Lagus and Kaski (1999).

The map is usually provided with a form field into which the user can type a query. We shall exemplify the use of the form field in connection with the next example.

5.3.3. The SOM of nearly 7 million patent abstracts

Next we describe how a content-addressable searching operation from an extremely large corpus of texts can be implemented. As mentioned earlier, to that end the interface of the map must be provided with a form field into which the user can type a query, e.g., a set of keywords or a descriptive sentence. The weighted vocabulary of this query shall match best with the weighted vocabulary of some of the documents.

The size of the text corpus in this application (Kohonen et al., 2000) was about 20 times that of Encyclopaedia Britannica. It consisted of 6,840,568 patent abstracts written in English. They were available on about two hundred CD ROMs and were obtained from US, European, and Japan patent offices as two databases: the "first page" database (1970–1997), and the "Patent Abstracts of Japan" (1976–1997). The average length of the abstracts was 132 words. The size of the vocabulary which was finally accepted after omission of stopwords, numerals and very rare words was 43,222 words, and they were weighted by the Shannon entropy.

The selected size of the SOM was 1,002,240 nodes and the dimensionality of each model (after having formed the random projections of the weighted word histograms) was 500. In order to fit a map of this size into the central computer of our laboratory, we had to reserve only one byte for each vector element. With 500 vector elements, the accuracy was still sufficient for statistical comparisons. This is the largest SOM that to our knowledge has

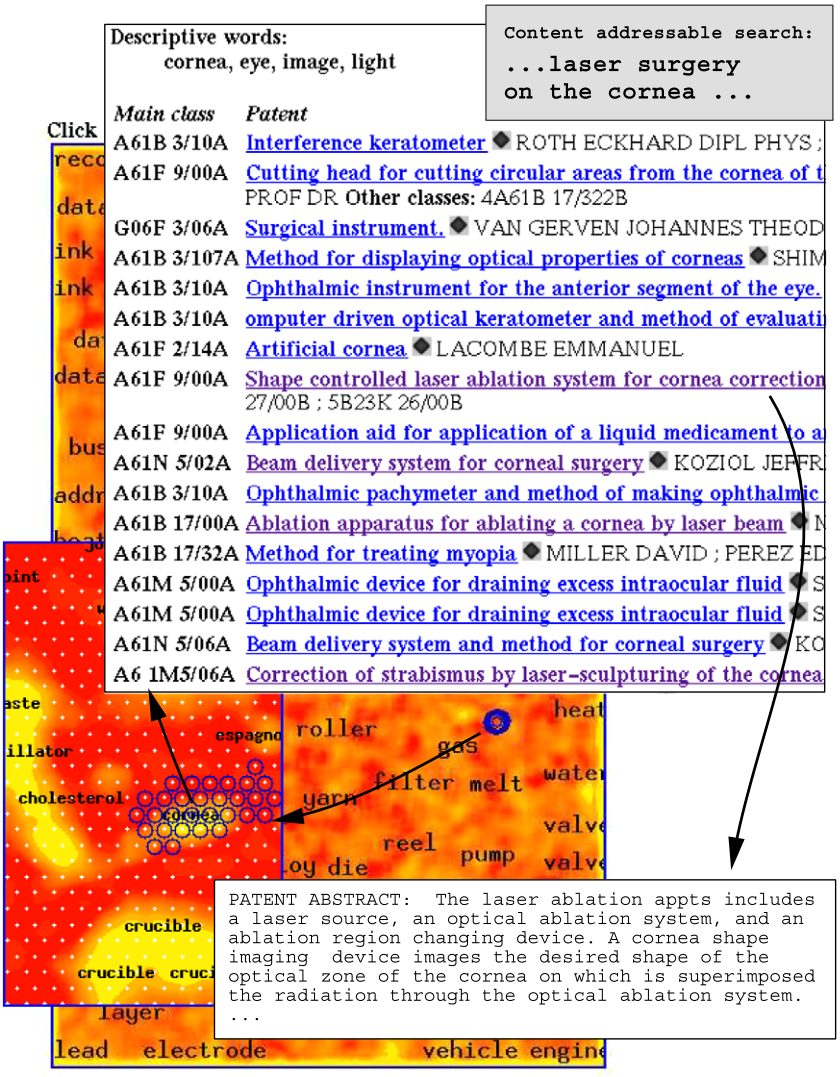

Fig. 5. The document map of nearly 7 million patent abstracts is depicted in the background, and the form field for the query is shown in the upper right-hand corner. The query was "laser surgery on the cornea", and twenty best-matching nodes have been marked with small circles. In the main map, these circles are fused together. An enlarged portion of the map, in which the twenty nodes are visible separately, can be obtained by digital zooming, as shown in the lower left-hand corner by the small circles. By clicking a particular node, the list of the titles associated with this node is obtained, and by clicking a selected title, the corresponding abstract can be read out.

ever been constructed, and even though we used many different speedup methods that shortened the computation time by several decades, it took six weeks to construct the map. The searching operations, on the other hand, could be accomplished in a few seconds, so the map could be used in real time. Fig. 5 exemplifies a case of content-addressable search.

5.4. Human endogenous retroviruses

Only about two per cent of human DNA controls the synthesis of proteins. The function of the rest is largely unknown. About eight per cent consists of segments called the human endogenous retroviruses (HERVs). The name means that the retrovirus sequences are endogenous, i.e. within the genome, in humans. These HERVs are remnants of ancient infections by retroviruses, such as the HIV. The retrovirus has integrated into the genome of the ancestor species and thus become part of it. It is suggested that these retrovirus sequences acquire new functions in the human genomes in regulating the activity of some genes. They may also produce proteins under some conditions.

The HERVs stem from several kinds of retroviruses. The current HERV taxonomy is incomplete: some sequences cannot be assigned to any class and the classification is ambiguous for others.

The mutual relationships of the HERVs and their similarities to some other related DNA elements were studied by Oja, Somervuo et al. (2003); Oja et al. (2004, 2005), using the SOM techniques. The map of a collection of 3661 HERV sequences was computed by the batch computing algorithm, and the distances between whole DNA sequences were evaluated by the FASTA method, cf. Pearson (1999) and Pearson and Lipman (1988).

The clustering of the DNA segments is shown in Fig. 6. Quite distinct clusters of various types are visible. They are separated by shades of gray around the hexagons that represent the SOM nodes. The cluster borders have been shaded by the *U*-matrix techniques (Ultsch, 1993).

The clustering visualized the same kinds of relationships as what are known from more traditional biological classifications and phylogenetic taxonomies. In addition, the SOM detected a completely new group of *epsilonretroviral sequences* and was able to shed light into the relationships of three pre-existing HERV families. The SOM detected a group of ERV9, HERVW, and HUERSP3 sequences, which suggested that the ERV9 and the HERVW sequences may have a common origin.

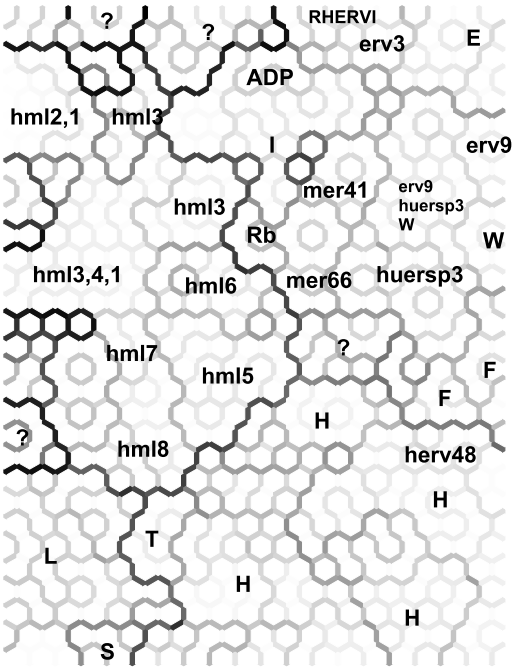

Fig. 6. The SOM of human endogenous retroviruses. The labels in the figure are manually assigned names for different areas of the map. The labels describe the class of the sequences in each area (class names like HERVADP, HERVH, HERVRb etc. have been abbreviated by dropping the HERV from the beginning). The question marks are used to mark areas where most of the sequences are unclassified. The gray scale coloring describes the distances between map units; black denotes large distance and white small. The darkest borders divide the three major groups and lighter borders the different groups inside the major groups, respectively.

6. Approximation of an input data item by a linear mixture of models

An analysis hitherto generally unknown is introduced in this chapter; cf. also Kohonen (2007) and Kohonen (2008). The purpose is to extend the use of the SOM by showing that instead of a single winner model, one can approximate the input data item more accurately by means of a set of *several models* that *together* define the input data item more accurately. It shall be emphasized that we do not mean k winners that are rank-ordered according to their matching. Instead, the input data item is approximated by an *optimized linear mixture of the models, using a nonlinear constraint*.

Consider the n-dimensional SOM models $\mathbf{m}_i, i = 1, 2, \ldots, p$, where p is the number of nodes in the SOM. Their general linear mixture is written as

$$k_1\mathbf{m}_1 + k_2\mathbf{m}_2 + \cdots + k_p\mathbf{m}_p = \mathbf{M}\,\mathbf{k}, \tag{10}$$

where the k_i are scalar-valued weighting coefficients, $\mathbf{k}$ is the p-dimensional column vector formed by them, and $\mathbf{M}$ is the matrix with $\mathbf{m}_i$ as its columns. Now $\mathbf{M}\,\mathbf{k}$ shall be the *estimate* of some input vector $\mathbf{x}$. The vectorial fitting error is then

$$\mathbf{e} = \mathbf{M}\,\mathbf{k} - \mathbf{x}. \tag{11}$$

Our aim is to minimize the norm of $\mathbf{e}$ in the sense of least squares. *However, the special nonlinear constraint must then be taken into account in this optimization.*

Much attention has recently been paid to least-squares problems where the fitting coefficients are constrained to *nonnegative values*. Such a constraint is natural, when the *negatives* of the items have no meaning, for instance, when the input item consists of statistical indicators that can have only nonnegative values, or is a weighted word histogram of a document. In these cases at least, the constraint contains additional information that is expected to make the fits more meaningful.

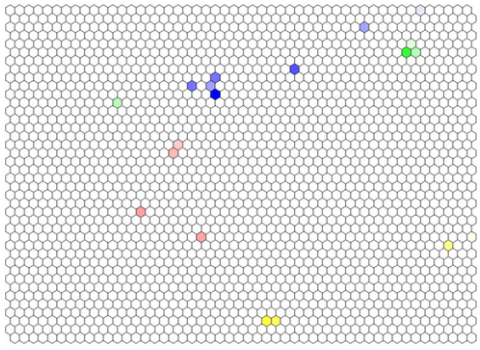

Fig. 7. A linear mixture of SOM models fitted to a new, unknown document. The weighting coefficients k_i in the mixture are shown by using a coloring with a relative shade of gray of the due models.

6.1. The lsqnonneg function

The present fitting problem belongs to the broader category of *quadratic programming* or *quadratic optimization*, for which numerous methods have been developed in recent years. A much-applied one-pass algorithm is based on the *Kuhn–Tucker theorem*, as explained in Lawson and Hanson (1974), but it is too involved to be reviewed here in full. Let it suffice to mention that it has been implemented in MATLAB as the function named the *lsqnonneg*. Below, the variables $\mathbf{k}$, $\mathbf{M}$, and $\mathbf{x}$ must be understood as being defined in the MATLAB format. Then we obtain the weight vector $\mathbf{k}$ as

$$\mathbf{k} = \text{lsqnonneg}(\mathbf{M}, \mathbf{x}). \tag{12}$$

The *lsqnonneg* function can be computed, and the result will be meaningful, for an *arbitrary rank* of the matrix $\mathbf{M}$. Nonetheless it has to be admitted that there exists a rare theoretical case where the optimal solution is not *unique*. This case occurs, if some of the $\mathbf{m}_i$ *in the final optimal mixture* are *linearly dependent*. In practice, if the input data items to the SOM are stochastic, the probability for the optimal solution not being unique is negligible. At any rate, the *locations* of the nonzero weights are unique even in this case!

6.2. Description of a document by a linear mixture of SOM models

The following analysis applies to most of the SOM applications. Here it is exemplified by textual databases.

In text analysis, one possible task is to find out whether a text comes from different sources, whereupon its word histogram is expected to be a linear mixture of other known histograms.

The example that demonstrates the fitting of a linear mixture of models to a given document is based on the lsqnonneg function. The text corpus was derived from the Reuters data as described earlier.

Fig. 7 shows a typical example, where a linear mixture of SOM models was fitted to a new, unknown document. The values of the weighting coefficients k_i in the mixture are shown by dots with relative shades of gray in the due positions of the SOM models. It is to be emphasized that this fitting procedure also defines the optimal *number* of the nonzero coefficients. In the experiments with large document collections, this number was usually very small, less than a per cent of the number of models.

When the models fall in classes that are known a priori, the weight of a model in the linear mixture also indicates the *weight of the class label associated with that model*. Accordingly, by summing up the weights of the various types of class labels one then obtains the *class-affiliation* of the input with the various classes.

7. Discussion

The self-organizing map (SOM) principle has been used extensively as an analytical and visualization tool in exploratory data analysis. It has had plenty of practical applications ranging from industrial process control and finance analyses to the management of very large document collections. New, very promising applications exist in bioinformatics. The largest applications so far have been in the management and retrieval of textual documents, of which this paper contains two large-scale examples.

Several commercial software packages as well as plenty of freeware on the SOM are available. This author strongly encourages the use of two public-domain software packages developed by the SOM_PAK Team (1990) and the SOM Toolbox Team (1999). Both packages contain auxiliary analytical procedures, and especially the SOM Toolbox, which makes use of the MATLAB functions, is provided with good and versatile graphics as well as thoroughly proven statistical analysis programs of the results.

This paper has applied the basic version of the SOM, on which the majority of applications is based. Nonetheless there may exist at least theoretical interest in different versions of the SOM, where some of the following modifications have been introduced.

Above the SOM grid was always taken as two dimensional, regular, and hexagonal. This form of the array is advantageous if the purpose is to *visualize* the overall structure of the whole database in one image. One of the different versions of the grid, developed, e.g., by the BLOSSOM Team (2005), is *cyclic*. Such "topologies" may have some meaning if the data themselves have a cyclic structure, or if the purpose is to avoid border effects of the noncyclic array of nodes. This may be the case if the SOM is used for process control, for the continuous and smooth description of all possible process states.

Another, often suggested version of the SOM is to replace the regular grid by a *structured graph* of nodes, where the structure and the number of nodes are determined dynamically; cf., e.g., Fritzke (1994).

Then, of course, there arises a question whether one could define a SOM-like system based on quite different mathematical principles. One of the interesting suggestions is the *generative topographic mapping (GTM)* introduced in Bishop, Svensen, and Williams (1998). It is based on direct computation of the topological relations of the nodes in the grid. A different, theoretically deep approach has been made by Van Hulle (2000), using information-theoretic measures in the construction of the SOM topology.

Perhaps one of the main virtues of the basic SOM algorithm is that one can compute really large mappings in reasonable time, using only personal computers.

Finally we must remind that the traditional methodology for the representation of similarity relations between data items is to *cluster* them according to some similarity or distance measure. The classical clustering algorithms as described by Anderberg (1973), Hartigan (1975), Jain and Dubes (1988), and Tryon and Bailey (1973), however, are usually rather heavy computationally, since every data item must be compared with all of the other ones, maybe reiteratively. For masses of data this is obviously time-consuming. The remedy provided by the SOM is to represent the set of all data items by a much smaller set of *models*, each of which stands for a subset of similar or almost similar data items. In the classification of an indefinite number of input data items, their comparison with the models can be at least an order of magnitude or more lighter operation.

Notwithstanding, our original motivation of the SOM research was an attempt to explain especially the *abstract feature maps* found in the *biological central nervous systems*. The biological aspects and implications of the SOM algorithm, however, have been almost totally ignored in the paper in presentation. The primary goal in the recent SOM research has been to develop algorithms and computational procedures for practical data mining applications.

Acknowledgments

The author is indebted to all of his collaborators who over the years have implemented the SOM program packages and applications. Dr. Merja Oja has kindly provided the picture and associated material about the more recent HERV studies.

References

Allinson, N., Yin, H., Allinson, L., & Slack, J. (Eds.) (2001). *Advances in self-organizing maps*. London, UK: Springer.
Amari, S. (1980). Topographic organization of nerve fields. *Bulletin of Mathematical Biology*, *42*, 339–364.
Anderberg, M. (1973). *Cluster analysis for applications*. New York, NY: Academic.
Bishop, C. M., Svensen, M., & Williams, C. K. I. (1998). GTM: the generative topographic mapping. *Neural Computation*, *10*, 215–234.
Cheng, Y. (1997). Convergence and ordering of Kohonen's Batch map. *Neural Computation*, *9*, 1667–1676.
Cottrell, M., & Fort, J. C. (1987). Étude d'un processus d'auto-organization. *Annales de l'Institut Henri Poincaré*, *23*, 1–20.
Cottrell, M., Fort, J.C., & Pagés, G. (1997). Theoretical aspects of the SOM algorithm. In *Proceedings of the WSOM 97, workshop on self-organizing maps*. Helsinki University of Technology. Neural Networks Research Centre. Espoo, Finland (pp. 246–267).
Deboeck, G., & Kohonen, T. (1998). *Visual explorations in finance with self-organizing maps*. London, UK: Springer, pp. 246–267.
Deerwester, S., Dumais, S., Furnas, G., & Landauer, K. (1990). Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, *41*, 391–407.
Dirichlet, G. L. (1850). Über die Reduktion der positiven quadratischen Formen mit drei unbestimmten ganzen Zahlen. *Journal für die Reine und Angewandte Mathematik*, *40*, 209–227.
Forgy, E. W. (1965). Cluster analysis of multivariate data: efficiency vs. interpretability of classifications. *Biometrics*, *21*, 768. abstract.
Fritzke, B. (1994). Growing cell structures—a self-organizing network for unsupervised and supervised learning. *Neural Networks*, *7*, 1441–1460.
Gersho, A. (1979). On the structure of vector quantizers. *IEEE Transactions on Information Theory*, *IT-25*, 373–380.
Gray, R. M. (1984). Vector quantization. *IEEE ASSP Magazine*, *1*, 4–29.
Grossberg, S. (1976). On the development of feature detectors in the visual cortex with applications to learning and reaction–diffusion systems. *Biological Cybernetics*, *21*, 145–159.
Hammer, B., Micheli, A., Sperduti, A., & Strickert, M. (2004). Recursive self-organizing network models. *Neural Networks*, *17*, 1061–1085.
Hartigan, J. (1975). *Clustering algorithms*. New York, NY: Wiley.
Heskes, T. M., & Kappen, B. (1993). Error potential for self-organization. In *Proceedings of ICNN'93, international conference on neural networks, vol. III* (pp. 1219–1223). Piscataway, NJ: IEEE Service Center.
Hubel, D. H., & Wiesel, T. H. (1962). Receptive fields, binocular and functional architecture in the cat's visual cortex. *Journal of Physiology*, *160*, 106–154.
Jain, A. K., & Dubes, R. C. (1988). *Algorithms for clustering of data*. Englewood Cliffs, NJ: Prentice-Hall.
Kaski, S. (1998). Dimensionality reduction by random mapping. In *Proceedings of IJCNN'98, international joint conference on neural networks* (pp. 413–418). Piscataway, NJ: IEEE Press.
Kaski, S., Honkela, T., Lagus, K., & Kohonen, T. (1998). WEBSOM—self-organizing maps of document collections. *Neurocomputing*, *21*(1), 101–117.
Kaski, S., Kangas, J., & Kohonen, T. (1998). Bibliography of self-organizing map (SOM) papers: 1981–1997. *Neural Computing Surveys*, *1*, 1–176. Available in electronic form at http://www.cis.hut.fi/research/som-bibl/vol1_4.pdf.
Kohonen, T. (1982a). Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, *43*, 59–69.
Kohonen, T. (1982b). Clustering, taxonomy, and topological maps of patterns. In *Proceedings of the sixth international conference on pattern recognition* (pp. 114–128). Washington, DC: IEEE Computer Soc. Press.
Kohonen, T. (1985). Median strings. *Pattern Recognition Letters*, *3*, 309–313.
Kohonen, T. (1989). *Self-organization and associative memory* (3rd ed.). Berlin–Heidelberg–Germany: Springer.
Kohonen, T. (1990). The self-organizing map. *Proceedings of the IEEE*, *78*, 1464–1480.
Kohonen, T. (1991). Self-organizing maps: optimization approaches. In T. Kohonen, K. Mäkisara, O. Simula, & J. Kangas (Eds.), *Artificial neural networks, II* (pp. 981–990). Amsterdam, Netherlands: North-Holland.
Kohonen, T. (1995). Emergence of invariant-feature detectors in self organization. In M. Palaniswami, Y. Attikiouzel, R. J. Marks II, D. Fogel, & T. Fukuda (Eds.), *Computational intelligence, a dynamic system perspective* (pp. 17–31). New York, NY: IEEE Press.
Kohonen, T. (1996). Emergence of invariant-feature detectors in the adaptive-subspace self organizing ma. *Biological Cybernetics*, *75*(4), 281–291.
Kohonen, T. (2001). *Self-organizing maps* (3rd ed.). Berlin–Heidelberg, Germany: Springer.
Kohonen, T. (2005). Pointwise organizing projections. In *Proceedings of the WSOM05, 5th workshop on self-organizing maps* Panthéon-Sorbonne University. Paris, France (pp. 1–8). Also available at http://samos.univ-paris1.fr/wsom/wsom05.html.
Kohonen, T. (2006). Self-organizing neural projections. *Neural Networks*, *19*, 723–733.

Kohonen, T. (2007). Description of input patterns by linear mixtures of SOM models. In *WSOM 2007 CD-ROM proceedings*, Bielefeld, Germany. Bielefeld University. Also available at http://biecoll.ub-bielefeld.de.
Kohonen, T. (2008). Data management by self-organizing maps. In J. M. Zurada, G. G. Yen, & J. Wang (Eds.), *Computational intelligence: research frontiers* (pp. 309–332). Berlin, Heidelberg, New York: Springer.
Kohonen, T., Hynninen, J., Kangas, J., & Laaksonen, J. (1996). *The self-organizing map program package, report A31*. Helsinki University of Technology, Laboratory of computer and information science. Espoo, Finland.
Kohonen, T., Kaski, S., Lagus, K., Salojärvi, J., Honkela, J., Paatero, V., & Saarela, A. (2000). Self organization of a massive document collection. *IEEE Transactions on Neural Networks*, *11*, 574–585.
Kohonen, T., Kaski, S., & Lappalainen, H. (1997). Self-organized formation of various invariant-feature filters in the adaptive-subspace SOM. *Neural Computation*, *9*, 1321–1344.
Kohonen, T., Oja, E., Simula, O., Visa, A., & Kangas, J. (1996). Engineering applications of the self-organizing map. *Proceedings of the IEEE*, *84*, 1358–1384.
Kohonen, T., & Somervuo, P. (2002). How to make large self-organizing maps for nonvectorial data. *Neural Networks*, *15*, 945–952.
Kohonen, T., & Xing, H. (2011). Contextually self-organized maps of Chinese words. In J. Laaksonen, & T. Honkela (Eds.), *Advances in self-organizing maps* (pp. 16–29). Berlin, Heidelberg: Springer.
Kruskal, J. B., & Wish, M. (1978). *Sage university paper series on quantitative applications in the social sciences No. 07-011, Multidimensional scaling*. Newbury Park, CA: Sage Publications.
Lagus, K., & Kaski, S. (1999). Keyword selection method for characterizing text document maps. In *Proceedings of ICANN'1999, ninth international conference on artificial neural networks. vol. 1* (pp. 371–376). London, UK: IEE.
Lampinen, J., & Oja, E. (1989). Self-organizing maps for spatial and temporal AR models. In M. Pietikäinen, & J. Röning (Eds.), *Proceedings of the 6th SCIA, scandal conference on image analysis* (pp. 120–127). Helsinki, Finland: Suomen Hahmontunnistustutkimuksen Seura ry.
Lawson, C. L., & Hanson, R. J. (1974). *Solving least-squares problems*. Englewood Cliffs, NJ: Prentice-Hall.
Lewis, D. D., Yang, Y., Rose, T. G., & Li, T. (2004). RCV1: a new benchmark collection for text categorization research. *Journal of Machine Learning Research*, *5*, 361–397.
Linde, Y., Buzo, A., & Gray, R. M. (1980). An algorithm for vector quantization. *IEEE Transactions on Communications*, *COM-28*, 84–95.
Lin, X., Soergel, D., & Marchionini, G. (1991). A self-organizing semantic map for information retrieval. In *Proceedings of the 14th annual international ACM/SIGIR conference on R&D in information retrieval* (pp. 262–269). New York, NY: ACM.
Lloyd, S. P. (1982). Least square quantization in PCM. *IEEE Transactions on Information Theory*, *28*(2), 129–137. Bell Telephone Laboratories Paper. 1957, republished in.
Luttrell, S. P. (1992). *Memorandum 4669*. Malvern, UK: Defence Research Agency.
Makhoul, J., Roucos, S., & Gis, H. (1985). Vector quantization in speech coding. *Proceedings of the IEEE*, *PROC-73*, 1551–1588.
Malsburg, Ch. v. d. (1973). Self-organization of orientation sensitive cells in the striate cortex. *Kybernetik*, *14*, 85–100.
Manning, C. D., & Schütze, H. (1999). *Foundations of statistical natural language processing*. Cambridge, MA: MIT Press.
Merkl, D. (1995). Content-based document classification with highly compressed input data. In Fogelman-Soulié, F., Gallinari, P. In *Proceedings of ICANN'95, international conference on artificial neural networks, vol. II*. Nanterre, France, EC2 (pp. 239–244).
Merkl, D., Tjoa, M., & Kappel, G. (1994). A self-organizing map that learns the semantic similarity of reusable software components. In A. C. Tsoi, & T. Downs (Eds.), *Proceedings of ACNN'94, 5th Australian conference on neural networks* (pp. 13–16). Australia: Brisbane.
Merzenich, M. M., Kaas, J. H., Wall, J. T., Nelson, R. J., Sur, M., & Felleman, D. J. (1983). Topographic reorganization of somatosensory cortical areas 3b and 1 in adult monkeys following restricted differentiation. *Neuroscience*, *8*, 33–55.
Miikkulainen, R. (1993). *Subsymbolic natural language processing: an integrated model of scripts, lexicon, and memory*. Cambridge, MA: MIT Press.
Miikkulainen, R., Bednar, J. A., Choe, Y., & Sirosh, J. (2005). *Computational maps in the visual cortex*. New York, NY: Springer.
Mountcastle, V. B. (1957). Modality and topographic properties of single neurons of cat's somatic sensory cortex. *Journal of Neurophysiology*, *20*, 408–434.
Naim, A., Ratnatunga, K. U., & Griffiths, R. E. (1997). Galaxy morphology without classification; self-organizing maps. *The Astrophysical Journal Supplement Series*, *111*, 357–367.
Nass, M. M., & Cooper, L. N. (1975). A theory for the development of feature departecting cells in visual cortex. *Biological Cybernetics*, *19*, 1–18.
Obermayer, K., & Sejnowski, T. (Eds.) (2001). *Self-organizing map formation: foundations of neural computation*. Cambridge, MA: MIT Press.
Oja, E., & Kaski, S. (1999). *Kohonen maps*. Amsterdam, Netherlands: Elsevier.
1998–2001 addendum Oja, M., Kaski, S., & Kohonen, T. (2003). Bibliography of self-organizing map (SOM) papers. *Neural Computing Surveys*, *3*, 1–156. Available in electronic form at http://www.cis.hut.fi/research/som-bibl/NCS_vol3_1.pdf.
Oja, M., Somervuo, P., Kaski, S., & Kohonen, T. (2003). Clustering of human endogenous retrovirus sequences with median self-organizing map. In *Proceedings of the WSOM'03, workshop on self-organizing maps*. Hibikino, Japan.
Oja, M., Sperber, G., Blomberg, J., & Kaski, S. (2004). Grouping and visualizing human endogenous retroviruses by bootstrapping median self-organizing maps. In *CIBCB 2004 IEEE symposium on computational intelligence in bioinformatics and computational biology*. 7–8 October. San Diego, USA (pp. 95–101).
Oja, M., Sperber, G., Blomberg, J., & Kaski, S. (2005). Self-organizing map-based discovery and visualization of human endogenous retroviral sequence groups. *International Journal of Neural Systems*, *15*(3), 163–179.
Pearson, W. (1999). The FASTA program package. ftp://ftp.virginia.edu/pub/fasta.
Pearson, W., & Lipman, D. (1988). Improved tools for biological sequence comparison. *Proceedings of the National Academy of Sciences of the United States of America*, *85*, 2444–2448.
Perez, R., Glass, R. L., & Shlaer, R. (1975). Development of specificity in the cat visual cortex. *Journal of Mathematical Biology*, *1*, 277–288.
Pöllä, M., Honkela, T., & Kohonen, T. (2009). Bibliography of SOM papers. Available at http://users.ics.tkk.fi/tho/online-papers/TKK-ICS-R23.pdf.
Ritter, H., & Kohonen, T. (1989). Self-organizing semantic maps. *Biological Cybernetics*, *61*, 241–254.
Ritter, H., Martinetz, T., & Schulten, K. (1992). *Neural computation and self-organizing maps: an introduction*. Reading, MA: Addison-Wesley.
Robbins, H., & Monro, S. (1951). A stochastic approximation method. *The Annals of Mathematical Statistics*, *22*, 400–407.
Salton, G., & McGill, M. J. (1983). *Introduction to modern information retrieval*. New York, NY: McGraw-Hill.
Sammon, J. W., Jr. (1969). A nonlinear mapping for data structure analysis. *IEEE Transactions on Computers*, *C-18*, 401–409.
Scholtes, J. C. (1991). Unsupervised context learning in natural language processing. In *Proceedings of IJCNN'91, international conference on neural networks, vol. I* (pp. 107–112). Piscataway, NJ: IEEE Service Center.
Seiffert, U., & Jain, L. C. (Eds.) (2002). *Self-organizing neural networks: recent advances and applications*. Heidelberg, Germany: Physica-Verlag.
SOM_PAK Team (1990). http://www.cis.hut.fi/research/som_lvq_pak.shtml. http://www.cis.hut.fi/research/som_pak/som_doc.ps.
SOM Toolbox Team (1999). http://www.cis.hut.fi/projects/somtoolbox/documentation/. http://www.cis.hut.fi/projects/somtoolbox/package/papers/techrep.pdf.
BLOSSOM Team (Japan) (2005). SOM Japan Co., Ltd. The BLOSSOM software package. http://www.somj.com/.
Suga, N., & O'Neill, W. E. (1979). Neural axis representing target range in the auditory cortex of the mustache bat. *Science*, *206*, 351–353.
Tokutaka, H., Kishida, S., & Fujimura, K. (1999). *Application of self-organizing maps—two-dimensional visualization of ulti-dimensional information*. Tokyo, Japan: Kaibundo (in Japanese).
Tokutaka, H., Ookita, M., & Fujimura, K. (2007). *SOM and the applications*. Tokyo, Japan: Springer Japan (in Japanese).
Tryon, R., & Bailey, D. (1973). *Cluster analysis*. New York, NY: McGraw-Hill.
Tunturi, A. R. (1950). Physiological determination of the arrangement of afferent connections to the middle ecrosylvian auditory area in the dog. *American Journal of Physiology*, *162*, 489–502.
Tunturi, A. R. (1952). The auditory cortex of the dog. *American Journal of Physiology*, *168*, 712–717.
Ultsch, A. (1993). Self-organizing neural networks for visualization and classification. In O. Opitz, B. Lausen, & R. Klar (Eds.), *Information and classification* (pp. 307–313). Berlin, Germany: Springer.
Van Hulle, M. (2000). *Faithful representations and topographic maps: from distortion-to information-based self-organization*. New York, NY: Wiley.
Vesanto, J., Alhoniemi, E., Himberg, J., Kiviluoto, K., & Parviainen, J. (1999). Self-organizing map for data mining in MATLAB: the SOM Toolbox. Simulation News Europe. 25:54, March.
Vesanto, J., Himberg, J., Alhoniemi, E., & Parhankangas, J. (1999). Self-organizing map in Matlab: the SOM Toolbox. In *Proceedings of Matlab DSP conference*. Nov. 16–17, Espoo, Finland (pp. 35–40).
Voronoi, G. (1907). Nouvelles applications des paramétres continus á la théorie des formes quadratiques. *Journal für die Reine und Angewandte Mathematik*, *133*, 97–178.
Zavrel, J. (1995). *Neural information retrieval—an experimental study of clustering and browsing of document collections with neural networks*. Master's Thesis. Amsterdam, Netherlands. University of Amsterdam.
Zeki, S. (1980). The representation of colours in the cerebral cortex. *Nature*, *284*, 412–418.