The Self-Organizing Maps: Background, Theories, Extensions and Applications

Hujun Yin

School of Electrical and Electronic Engineering, The University of Manchester,
M60 1QD, UK, hujun.yin@manchester.ac.uk

1 Introduction

For many years, artificial neural networks (ANNs) have been studied and used to model information processing systems based on or inspired by biological neural structures. They not only can provide solutions with improved performance when compared with traditional problem-solving methods, but also give a deeper understanding of human cognitive abilities. Among various existing neural network architectures and learning algorithms, Kohonen's self-organizing map (SOM) [46] is one of the most popular neural network models. Developed for an associative memory model, it is an unsupervised learning algorithm with a simple structure and computational form, and is motivated by the retina-cortex mapping. Self-organization in general is a fundamental pattern recognition process, in which intrinsic inter- and intra-pattern relationships among the stimuli and responses are learnt without the presence of a potentially biased or subjective external influence. The SOM can provide topologically preserved mapping from input to output spaces. Although the computational form of the SOM is very simple, numerous researchers have already examined the algorithm and many of its problems, nevertheless research in this area goes deeper and deeper – there are still many aspects to be exploited.

In this Chapter, we review the background, theories and statistical properties of this important learning model and present recent advances from various pattern recognition aspects through a number of case studies and applications. The SOM is optimal for vector quantization. Its topographical ordering provides the mapping with enhanced fault- and noise-tolerant abilities. It is also applicable to many other applications, such as dimensionality reduction, data visualization, clustering and classification. Various extensions of the SOM have been devised since its introduction to extend the mapping as effective solutions for a wide range of applications. Its connections with other learning paradigms and application aspects are also exploited. The Chapter is intended

H. Yin: *The Self-Organizing Maps: Background, Theories, Extensions and Applications*, Studies in Computational Intelligence (SCI) **115**, 715–762 (2008)
www.springerlink.com

to serve as an updated, extended tutorial, a review, as well as a reference for advanced topics in the subject.

2 Background

Kohonen's self-organizing map (SOM) is an abstract mathematical model of topographic mapping from the (visual) sensors to the cerebral cortex. Modeling and analyzing the mapping are important to understanding how the brain perceives, encodes, recognizes and processes the patterns it receives and thus, if somewhat indirectly, are beneficial to machine-based pattern recognition. This Section looks into the relevant biological models, from two fundamental phenomena involved – lateral inhibition and Hebbian learning – to Willshaw and von der Malsburg's self-organization retinotopic model, and then subsequently to Kohonen's simplified and abstracted SOM model. Basic operations and the SOM algorithm, as well as methods for choosing model parameters, are also given.

2.1 Biological Background: Lateral Inhibition and Hebbian Learning

Human visual perception and the brain make up the most complex cognition system and the most complex of all biological organs. Visual inputs contribute to over 90% of the total information (from all sensors) entering the brain. Nature and our living environment are constantly shaping our perception and cognition systems. Physiologically, human and indeed other animal visual (and other perception) systems have been evolved to so many different types of eyes and mammalian visual pathways for different purposes and conditions. For example, many insects have compound eyes, which have good temporal resolution and are more directionally sensitive and at the same time make them smaller and group them into a single structure – giving insects a better ability to detect spatial patterns and movement in order to escape from predators. Compound eyes have good time resolving power. Human eyes need 0.05 second to identify objects, while compound eyes need only 0.01 second. That is, they are good at recognizing (fast) moving objects. Eyes of large animals including humans have evolved to single-chambered 'camera lens' eyes, which have excellent angle resolution and are capable of seeing distant objects. Camera eyes have great space resolving power: high spatial resolution for good details of objects and patterns, and long depth resolution for both very near and very far objects. They also have brilliant sensitivities for light intensity – over 20 billion times (that is, 206 dB) range (the brightest to the darkest) – compared with most digital cameras, which are below 16-bit resolution (30 dB).

What information do eyes extract from the retina or sensory cells? Visual information is processed in both the retina and brain, but it is widely believed

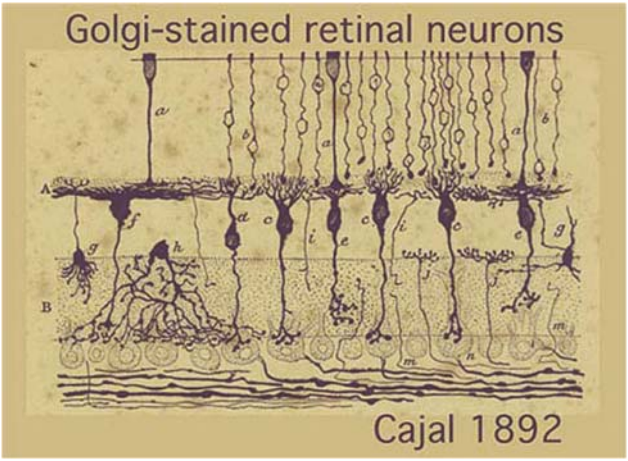


Fig. 1. A cross-sectional diagram of the retina, drawn by Santiago Ramón y Cajal (1853–1934)

and verified that most processing is done in the retina, such as extracting lines, angles, curves, contrasts, colours, and motion. The retina then encodes the information and sends through optic nerves and optic chiasma, where some left and right nerves are crossed, to the brain cortex in the left and/or right hemispheres. The retina is a complex neural network. Figure 1 shows a drawing of the cross section of the retina. The human retina has over 100 million photosensitive cells (combining rods and cones), processing in parallel the raw images, codes and renders to just over one million optic nerves, to be transmitted in turn to the brain cortex.

The Perceptron models some cells in the retina, especially the bipolar and ganglion cells. These cells take inputs from the outputs of cells in the previous layer. To put many units together and connect them into layers, one may hope the resulting network – the Multi-Layer Perceptron – will have some functionality similar to the retina (despite neglecting some horizontal interconnections). Indeed, such a structure has been demonstrated as being capable of certain cognitive and information processing tasks.

Cells in neural networks (either in the retina or brain) also connect and interact horizontally. The experiment on limulus, or the horseshoe crab, by Haldan K. Hartline (1967 Nobel Prize Laureate) and his colleagues in the 1960s, has confirmed such processing on the limulus retina (the surface of the compound eye is shown in Fig. 2(a)). They revealed the so-called 'lateral inhibition' activity among the retina cells. In other words, there exist both short-range excitatory interaction between nearby cells, as well as long-range inhibitory interaction between distant neighbours. This consequently explains the so-called 'Mach band' phenomenon on the edges or sharp changes of light intensity [87] – see Fig. 2(b).

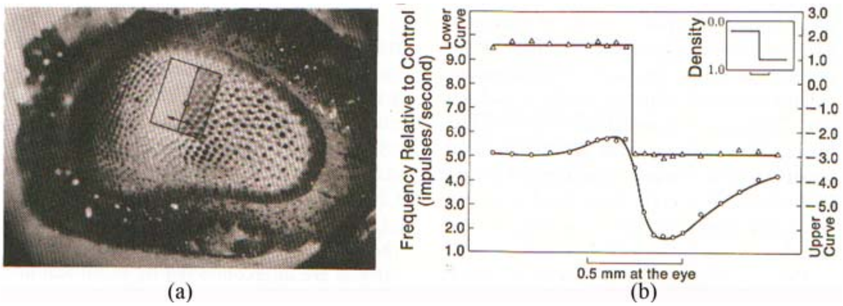


Fig. 2. (**a**) Surface of the Limulus eye and simuli: small spot light and rectangular lighter/darker pattern; (**b**) recordings of spike rate in the ommatidium axon (the upper curve is the response to the small spot light at high and low intensities corresponding to those of the test pattern in the insert; the lower curve is the response to the rectangular lighter/darker test pattern (from [87]; also see [99])

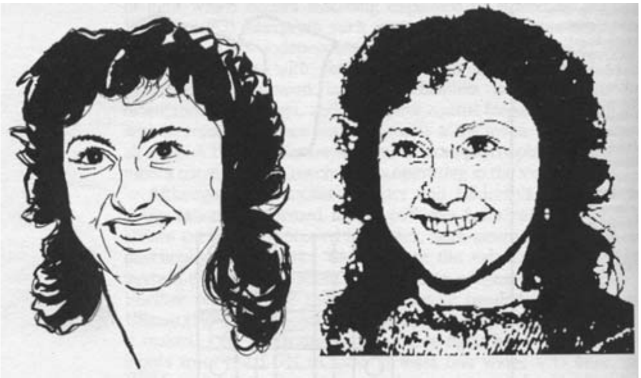


Fig. 3. Artistic drawing of a woman (*left*), and applying Pearson and Ronbinson's edge detector – an improved Marr and Hildreth mask – on the photo of the same woman (*right*) ([86], reprinted by permission of Cambridge University Press; also cited in [9])

Lateral inhibition tells us that neurons in the retina do not just feed the information to upper levels, but also perform an important visual processing task: edge detection and enhancement. Figure 3 demonstrates a psychological experiment that also confirms such fundamental processing in visual perception.

Neural networks present completely different approaches to computing and machine intelligence from traditional symbolic AI. The goal is to emulate the way that natural systems, especially brains, perform on various cognitive or recognition tasks. When a network of simple processing units interconnect with each other, there are potentially a massive number of synaptic weights available to be configured and modified such that the network will suit a particular task. This configuration and modification process is carried out by a learning procedure, that is, learning or training algorithm. The way these simple units connect together is called the neural architecture. There are two basic types: feed-forward, in which layers of neurons are concatenated, and recurrent, in which neurons have feedback from themselves and others. Examples of these two architectures are the Multi-Layer Perceptron (MLP), and the Hopfield network, respectively.

The Oxford English Dictionary defines learning as "the process which leads to the modification of behavior".[1] Learning in general intelligent systems is often referred to as a process of the systems' adaptation to their experience or environment – a key feature of intelligence. According to Hebb, learning occurs when "some growth process or metabolic change takes place" [30]. Learning in the context of neural networks can be defined as [29]:

> "Learning is a process by which the free parameters of neural networks are adapted through a process of simulation by the environment in which the network is embedded. The type of learning is determined by the manner in which the parameter changes take place."

Neural networks also differ from traditional pattern recognition approaches, which usually require solving some well-defined functions or models, such as feature extraction, transformation, and discriminant analysis by a series of processing steps. Neural networks can simply learn from examples. Presented repeatedly with known examples of raw patterns and with an appropriate learning or training algorithm, they are able to extract the most intrinsic nature of the patterns and perform recognition tasks. They will also have the ability to carry out similar recognition tasks, not only on trained examples but also on unseen patterns. Learning methods and algorithms undoubtedly play an important role in building successful neural networks.

Although many learning methods have been proposed, there are two fundamental kinds of learning paradigms: supervised learning and unsupervised learning. The former is commonly used in most feed-forward neural networks, in which the input-output (or input-target) functions or relationships are built from a set of examples, while the latter resembles a self-organization process in the cortex and seeks inter-relationships and associations among the input.

The most representative supervised learning rule is error-correction learning. When presented with an input-output pair, learning takes place when an error exists between a desired response or target output and the actual output of the network. This learning rule applies an adjustment, proportional to this error, to the weights of the neuron concerned. Derivation of such a rule can be often traced backed to minimizing the mean-square error function – more details can be found in [29]. A derivative of supervised learning is so-called

[1] Simpson JA, Weiner ESC (eds.) (1988) *Oxford English Dictionary (2nd ed).* Clarendon Press, Oxford, UK.

reinforcement learning, based on trail-and-error (and reward). The following definition has been given by [101]:

> "If an action taken by a learning system is followed by a satisfactory state of affairs, then the tendency of the system to produce that particular action is strengthened or reinforced. Otherwise, the tendency of the system to produce that action is weakened".

In contrast to supervised learning, there is no direct teacher to provide how much output error a particular action has produced. Instead, the output has been quantified into either 'positive' or 'negative' corresponding to closer to or further from the goal. Reinforcement learning has popular backing from psychology.

Self-organization often involves both competition and correlative learning. When presented with a stimulus, neurons compete among themselves for possession or ownership of this input. The winners then strengthen their weights or their relationships with this input. Hebbian learning is the most common rule for unsupervised or self-organized learning. As stated in [30]:

> "When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic changes take place in one or both cells such that As efficiency as one of the cells firing B, is increased."

Mathematically, the Hebbian learning rule can be directly interpreted as,

$$\frac{\partial w_{ij}(t)}{\partial t} = \alpha x_i(t) y_i(t) \tag{1}$$

where α is a positive learning rate ($0 < \alpha < 1$), and x and y are the input and output of the neural system, respectively (or can also be regarded as the outputs of the two neurons). That is, the change of the synaptic weight is proportional to the correlation between an input and its associated output. If the input and output are coherent, the weight connecting them is strengthened (xy is positive), otherwise, weakened (xy is either negative or zero).

Hebbian learning requires some modification before it can be used in practice, otherwise the weight will easily become saturated or unlimited (positive or negative). One solution is to add a 'forgetting term' to prevent weights from increasing/decreasing monotonically as in the SOM (see the next Section). Alternatively, we can normalize the weights. For instance, Oja proposed a weight normalization scheme on all weights. This naturally introduces a forgetting term to the Hebbian rule [81],

$$\begin{aligned} w_i(t+1) &= \frac{w_i(t) + \alpha x_i(t) y(t)}{\{\sum_{j=1}^{n} [w_j(t) + \alpha x_j(t) y(t)]^2\}^{1/2}} \\ &\approx w_i(t) + \alpha(t)[x_i(t) - y(t) w_i(t)] + O(\alpha^2) \end{aligned} \tag{2}$$

where $\mathrm{O}(\alpha^2)$ represents second- and high-order terms in α, and can be ignored when a small learning rate is used.

The resulting Oja learning algorithm is a so-called principal component network, which learns to extract the most variant directions among the data set. Other variants of Hebbian learning include many algorithms used for Independent Component Analysis ([36, 82]).

2.2 From Von Marsburg and Willshaw's Self-Organization Model to Kohonen's SOM

External stimuli are received by various sensors or receptive fields (for example, visual-, auditory-, motor-, or somato-sensory), coded or abstracted by the living neural networks, and projected through axons onto the cerebral cortex, often to distinct parts of the cortex. In other words, the different areas of the cortex (cortical maps) often correspond to different sensory inputs, though some brain functions require collective responses. Topographically ordered maps are widely observed in the cortex. The main structures (primary sensory areas) of the cortical maps are established before birth ([47], [115], and similar) in a predetermined topographically ordered fashion. Other more detailed areas (associative areas), however, are developed through self-organization gradually during life and in a topographically meaningful order. Therefore studying such topographically ordered projections, which had been ignored during the early period of neural information processing research [48], is undoubtedly important for understanding and constructing dimension-reduction mapping and for the effective representation of sensory information and feature extraction.

The self-organized learning behavior of brains has been studied for a long time by many people. Pioneering works include for example, Hebb's learning law (1949) [30], Marr's theory of the cerebellar cortex (1969) [72], Willshaw, Buneman and Longnet-Higgins's non-holographic associative memory (1969) [114], Gaze's studies on nerve connections (1970), von der Malsburg and Willshaw's self-organizing model of retina-cortex mapping ([111], [115]), Amari's mathematical analysis of self-organization in the cortex (1980), Kohonen's self-organizing map (1982), and Cottrell and Fort's self-organizing model of retinotopy (1986). Many still have immense influence on today's research. Von der Malsburg (1973) and Willshaw (1976) first developed, in mathematical form, the self-organizing topographical mapping, mainly from two-dimensional presynaptic sheets to two-dimensional postsynaptic sheets, based on retinatopic mapping: the ordered projection of visual retina to the visual cortex (see Fig. 4). Their basic idea was:

> "......the geometrical proximity of presynaptic cells is coded in the form of correlations in their electrical activity. These correlations can be used in the postsynaptic sheet to recognize axons of neighbouring presynaptic cells and to connect them to neighbouring postsynaptic cells, hence producing a continuous mapping......"

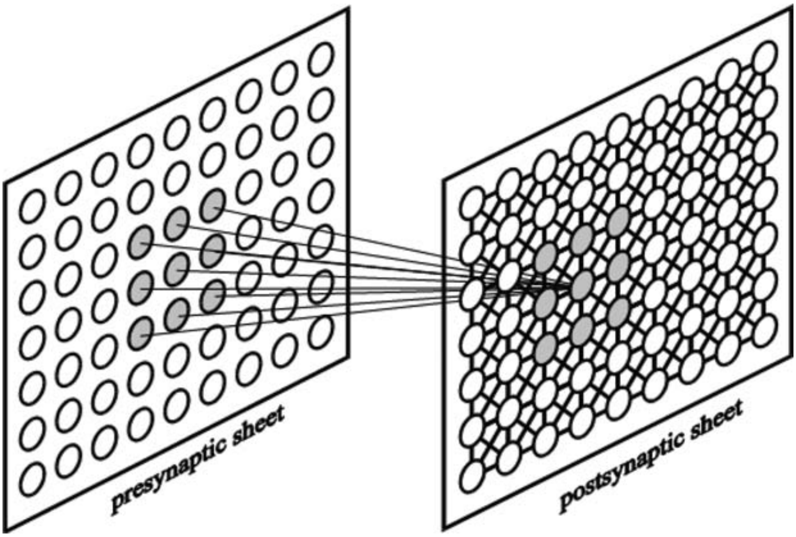


Fig. 4. von der Malsburg's self-organizing map model: local clusters in a presynaptic sheet are connected to local clusters in a postsynaptic sheet; there are lateral interconnections within the postsynaptic sheet (solid lines are used to indicate such connections)

The model uses short-range excitatory connections between cells so that activity in neighbouring cells becomes mutually reinforced, and uses long-range inhibitory interconnections to prevent activity from spreading too far. The postsynaptic activities $\{y_j(t), j = 1, 2, \ldots N_y\}$, at time t, are expressed by

$$\frac{\partial y_i(t)}{\partial t} + cy_i(t) = \sum_j w_{ij}(t)x_i(t) + \sum_k e_{ik}y_k^*(t) - \sum_{k'} b_{ik'}y_{k'}^*(t) \tag{3}$$

where c is the membrane constant, $w_{ij}(t)$ is the synaptic strength between cell i and cell j in pre- and post-synaptic sheets respectively, $\{x_i(t), i = 1, 2, \ldots N_x\}$, the state of the presynaptic cells, equal to 1 if cell i is active or 0 otherwise, e_{kj} and b_{kj} are short-range excitation and long-range inhibition constants respectively, and $y_j^*(t)$ is an active cell in postsynaptic sheet at time t. The postsynaptic cells fire if their activity is above a threshold, say,

$$y_i^*(t) = \begin{cases} y_j^*(t) - \theta, & \text{if} \quad y_j^*(t) > \theta \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

The modifiable synaptic weights between pre- and post-synaptic sheets are then facilitated in proportion to the product of activities in the appropriate pre- and postsynaptic cells (Hebbian learning):

$$\frac{\partial w_{ij}(t)}{\partial t} = \alpha x_i(t)y_j^*(t), \qquad \text{subject to} \quad \frac{1}{N_x}\sum_i w_{ij} = \text{constant} \tag{5}$$

where α is a small constant representing the learning rate. To prevent the synaptic strengths becoming unstable, the total strength associated with each postsynaptic cell is limited by normalization to a constant value after each iteration.

Kohonen (1982) abstracted the above self-organizing learning principle and function and proposed a much simplified learning mechanism which cleverly incorporates the Hebbian learning rule and lateral interconnection rules and can emulate the self-organizing learning effect. Although the resulting SOM algorithm was more or less proposed in a heuristic manner ([54]), it is a simplified and generalized model of the above self-organization process. As commented in [91]:

> "Kohonen's model of self-organizing maps represented an important abstraction of the earlier model of von der Malsburg and Willshaw; the model combines biological plausibility with proven applicability in a broad range of difficult data processing and optimization problems..."

In Kohonen's model, the postsynaptic activities are similar to Eqn. (3). To find the solutions of this equation and ensure they are non-negative properties, a sigmoid type of nonlinear function is applied to each postsynaptic activity:

$$y_j(t+1) = \varphi\left[\mathbf{w}_j^T \mathbf{x}(t) + \sum_i h_{ij} y_i(t)\right] \tag{6}$$

where h_{kj} is similar to e_{kj} and b_{kj}, and the input is described as a vector as the map can be extended to any dimensional input. A typical structure is shown in Fig. 5.

A spatially-bounded cluster or bubble will then be formed among the postsynaptic activities and will stabilize at a maximum (without loss of generality which is assumed to be unity) when within the bubble, or a minimum (that is, zero) otherwise,

$$y_j(t+1) = \begin{cases} 1, & \text{if neuron } j \text{ is inside the bubble} \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

The bubble is centred on a postsynaptic cell whose synaptic connection with the presynaptic cells is mostly matched with the input or presynaptic state, that is the first term in the function in Eqn. (6) is the highest. The range or size, denoted by $\eta(t)$, of the bubble depends on the ratio of the lateral excitation and inhibition. To modify the Hebbian learning rule, in other words Eqn. (5), instead of using normalization, a forgetting term, $\beta y_j(t) w_{ij}(t)$, is added. Let $\alpha = \beta$, and apply Eqn. (7), the synaptic learning rule can then be formulated as,

$$\begin{aligned} \frac{\partial w_{ij}(t)}{\partial t} &= \alpha y_j(t) x_i(t) - \beta y_j(t) w_{ij}(t) = \alpha[x_i(t) - w_{ij}(t)] y_j(t) \\ &= \begin{cases} \alpha[x_i(t) - w_{ij}(t)], & \text{if } j \in \eta(t) \\ 0 & \text{if } j \notin \eta(t) \end{cases} \end{aligned} \tag{8}$$

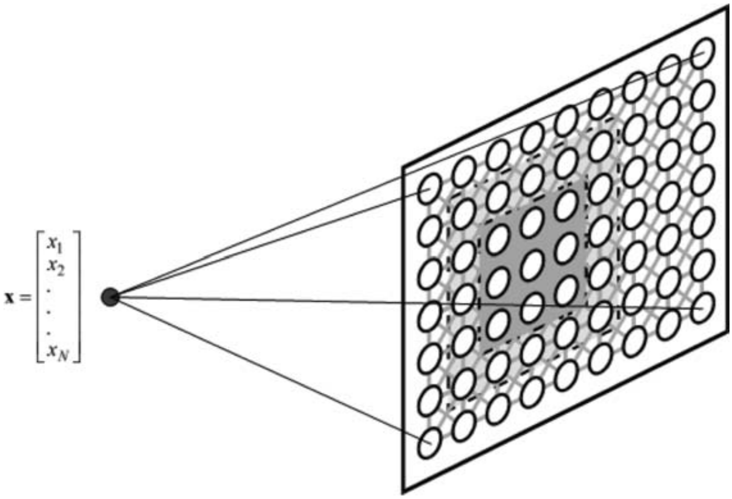

$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ \cdot \\ x_N \end{bmatrix}$

Fig. 5. Kohonen's self-organizing map model. The input is connected to every cell in the postsynaptic sheet (the map). The learning makes the map localized, in other words different local fields will respond to different ranges of inputs. The lateral excitation and inhibition connections are emulated by a mathematical modification, namely local sharing, to the learning mechanism (so there are no actual connections between cells – grey lines are used to indicate these virtual connections)

At each time step the best matching postsynaptic cell is chosen according to the first term of the function in Eqn. (6), which is the inner product, or correlation, of the presynaptic input and synaptic weight vectors. When normalization is applied to the postsynaptic vectors, as it usually is, this matching criterion is similar to the Euclidean distance measure between the weight and input vectors. Therefore the model provides a very simple computational form. The lateral interconnection between neighbouring neurons and the 'Mexican-hat' excitatory or inhibitory rules are simulated (mathematically) by a simple local neighbourhood excitation centred on the winner. Thus the neuron's lateral interconnections (both excitatory and inhibitory) have been replaced by neighbourhood function adjustment. The neighbourhood function's width can emulate the control of the exciting and inhibiting scalars. The constrained (with a decaying or forgetting term) Hebbian learning rule has been simplified and becomes a competitive learning model.

Most of Kohonen's work has been in associative memories ([43]–[48], and so on). In his studies, he has found that the spatially ordered representation of sensory information in the brain is highly related to the memory mechanism, and that the inter-representation and information storage can be implemented simultaneously by an adaptive, massively parallel, and self-organizing network [48]. This simulated cortex map, on the one hand can perform a self-organized search for important features among the inputs, and on the other hand can arrange these features in a topographically meaningful order. This is why the map is also sometimes termed the 'self-organizing feature map', or SOFM. In this Chapter, however, it will be referred to by its commonly used term, the 'self-organizing map' (SOM), which comes from Kohonen's original definition and purpose, namely associative memory.

2.3 The SOM Algorithm

The SOM uses a set of neurons, often arranged in a 2-D rectangular or hexagonal grid, to form a discrete topological mapping of an input space, $\mathbf{X} \in \mathcal{R}^n$. At the start of the learning, all the weights $\{\mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_M\}$ are initialized to small random numbers. $\mathbf{w}_i$ is the weight vector associated to neuron i and is a vector of the same dimension – n – of the input, M is the total number of neurons, and let $\mathbf{r}_i$ be the location vector of neuron i on the grid. Then the algorithm repeats the steps shown in Algorithm 1, where $\eta(\nu, k, t)$ is the neighbourhood function, and Ω is the set of neuron indexes. Although one can use the original stepped or top-hat type of neighbourhood function (one when the neuron is within the neighbourhood; zero otherwise), a Gaussian form is often used in practice – more specifically $\eta(\nu, k, t) = exp[-\frac{\|\mathbf{r}_\nu - \mathbf{r}_k\|^2}{2\sigma(t)^2}]$, with σ representing the effective range of the neighbourhood, and is often decreasing with time.

Algorithm 1 Self-Organizing Map algorithm

```
**repeat**
    1. At each time $t$, present an input $\mathbf{x}(t)$, and select the winner,
        $$\nu(t) = \arg\min_{k \in \Omega} \| \mathbf{x}(t) - \mathbf{w}_k(t) \| \quad (9)$$
    2. Update the weights of the winner and its neighbours,
        $$\Delta\mathbf{w}_k(t) = \alpha(t)\eta(\nu, k, t)[\mathbf{x}(t) - \mathbf{w}_\nu(t)] \quad (10)$$
**until** the map converges
```

The coefficients $\{\alpha(t), t \geq 0\}$, termed the 'adaptation gain', or 'learning rate', are scalar-valued, decrease monotonically, and satisfy [47]:

$$(i)\ 0 < \alpha(t) < 1; \quad (ii)\ \lim_{t\to\infty} \sum \alpha(t) \to \infty; \quad (iii)\ \lim_{t\to\infty} \sum \alpha^2(t) < \infty; \quad (11)$$

They are the same as to those used in stochastic approximation ([92,94]). The third condition in Eqn. (11) has been relaxed by Ritter and Schulten to a less restrictive one, namely, $\lim_{t\to\infty} \alpha(t) \to 0$ [90].

If the inner product similarity measure is adopted as the best matching rule,

$$\nu(t) = \arg\min_{k\in\Omega}[\mathbf{w}_k^T \mathbf{x}(t)] \tag{12}$$

then the corresponding weight updating will become [53]:

$$\mathbf{w}_k(t+1) = \begin{cases} \frac{\mathbf{w}_k(t)+\alpha(t)\mathbf{x}(t)}{\|\mathbf{w}_k(t)+\alpha(t)\mathbf{x}(t)\|} & k \in \eta_\nu \\ \mathbf{w}_k(t) & k \notin \eta_\nu \end{cases} \tag{13}$$

Such a form is often used in text/document mining applications (for example, [23]).

The SOM algorithm vector-quantizes or clusters the input space and produces a map which preserves topology. It can also be and has been used for classification. In this case, the map is trained on examples of known categories. The nodes are then classified or labelled so that the map can be used to classify unseen samples. Various methods for labelling nodes can be found in [59]. The classification performance can be further improved by the Learning Vector Quantization (LVQ) algorithm [53].

3 Theories

3.1 Convergence and Cost Functions

Although the SOM algorithm has a simple computational form, a formal analysis of it and the associated learning processes and mathematical properties is not easily realized. Some important issues still remain unanswered. Self-organization, or more specifically the ordering process, has been studied in some depth; however a universal conclusion has been difficult, if not impossible, to obtain. This Section reviews and clarifies the statistical and convergence properties of the SOM and associated cost functions, the issue that still causes confusions to many even today. Various topology preservation measures will be analyzed and explained.

The SOM was proposed to model the sensory-to-cortex mapping thus the unsupervised associative memory mechanism. Such a mechanism is also related to vector quantization (VQ) [63] in coding terms. The SOM has been shown to be an asymptotically optimal VQ [117, 126]. More importantly, with its neighbourhood learning, the SOM is both an error tolerant VQ and Bayesian VQ [66,68].

Convergence and ordering has only been formally proven in the trivial one-dimensional case. A full proof of both convergence and ordering in multidimensional systems is still outstanding, although there have been several attempts (for instance, [19,20,62,64,90,126]). [19] and [20] especially showed that there was no cost function that the SOM will follow exactly. Such an issue is also linked to the claimed lack of an exact cost function that the algorithm follows. Recent work by various researchers has already shed light on this intriguing issue surrounding the SOM. Yin and Allison extended the Central Limit Theorem and used it to show that when the neighbourhood reduces to just the winner, the weight vectors (code references) are asymptotically Gaussian distributed and will converge in a mean squares sense to the means of the Voronoi cells – in other words, an optimal VQ (with the SOM's nearest distance winning rule) [126],

$$\mathbf{w}_k \rightarrow \frac{1}{P(X_k)} \int_{V_k} \mathbf{x} p(\mathbf{x}) d\mathbf{x} \tag{14}$$

where V_k is the Voronoi cell (the data region) for which the weight vector w_k is responsible, and $p(\mathbf{x})$ is the probability density function of the data. In general cases with the effect of the neighbourhood function, the weight vector is a kernel smoothed mean [119],

$$\mathbf{w}_k \rightarrow \frac{\sum_{t=1}^{T} \eta(\nu, k, t)\mathbf{x}(t)}{\sum_{t=1}^{T} \eta(\nu, k, t)} \tag{15}$$

Yin and Allison have also proved that the initial state has a diminishing effect on the final weights when the learning parameters follow the convergence conditions [126]. Such an effect has been recently verified by [15] using Monte-Carlo bootstrap cross validation; the ordering was not considered. In practice, as only limited data samples are used and training is performed in finite time, good initialization can help guide to a faster or even better convergence. For example, initializing the map to a principal linear sub-manifold can reduce the ordering time, if the ordering process is not a key requirement.

Luttrell first related hierarchical noise tolerant coding theory to the SOM. When the transmission channel noise is considered, a two-stage optimization has to be done, not only to minimize the representation distortion (as in VQ) but also to minimize the distortion caused by the channel noise. He revealed that the SOM can be interpreted as such a coding algorithm. The neighbourhood function acts as the model for the channel noise distribution and should not go to zero as in the original SOM. Such a noise tolerant VQ has the following objective function ([66, 67]),

$$D_2 = \int d\mathbf{x} p(\mathbf{x}) \int d\mathbf{n} \pi \parallel \mathbf{x} - \mathbf{w}_k \parallel^2 \tag{16}$$

where $\mathbf{n}$ is the noise variable and $\pi(\mathbf{n})$ is the noise distribution. [18] and [78] have also linked the SOM and this noise tolerant VQ with minimal wiring of cortex-like maps.

When the code book (the map) is finite, the noise can be considered as discrete, then the cost function can be re-expressed as,

$$D_2 = \sum_i \int_{V_i} \sum_k \pi(i,k) \parallel \mathbf{x} - \mathbf{w}_k \parallel^2 p(\mathbf{x}) d\mathbf{x} \tag{17}$$

where V_i is the Voronoi region of cell i. When the channel noise distribution is replaced by a neighbourhood function (analogous to inter-symbol dispersion), it becomes the cost function of the SOM algorithm. The neighbourhood function can be interpreted as a channel noise model. Such a cost function has been discussed in the SOM community (for example, [32], [50], [58], [117]). The cost function is therefore,

$$E(\mathbf{w}_1, \cdots, \mathbf{w}_N) = \sum_i \int_{V_i} \sum_k \eta(i,k) \parallel \mathbf{x} - \mathbf{w}_k \parallel^2 p(\mathbf{x}) d\mathbf{x} \tag{18}$$

This leads naturally to the SOM update algorithm using the sample or stochastic gradient descent method [92] – that is, for each Voronoi region, the sample cost function is,

$$\hat{E}_i(\mathbf{w}_1, \cdots, \mathbf{w}_N) = \int_{V_i} \sum_k \eta(i,k) \parallel \mathbf{x} - \mathbf{w}_k \parallel^2 p(\mathbf{x}) d\mathbf{x} \tag{19}$$

The optimization for all weights $\{\mathbf{w}_1, \mathbf{w}_2, \cdots, \mathbf{w}_N\}$ can be sought using the sample gradients. The sample gradient for $\mathbf{w}_j$ is,

$$\frac{\partial \hat{E}_i(\mathbf{w}_1, \cdots, \mathbf{w}_N)}{\partial \mathbf{w}_j} = \frac{\partial \sum_k \eta(i,k) \parallel \mathbf{x} - \mathbf{w}_k \parallel^2}{\partial \mathbf{w}_j} = 2\eta(i,k) \parallel \mathbf{x} - \mathbf{w}_j \parallel \tag{20}$$

which leads to the SOM update rule – Eqn. (10). Note that although the neighbourhood function $\eta_{i,k}$ is only implicitly related to $\mathbf{w}_j$, it does not contribute to the weight optimization, nor does the weight optimization lead to its adaptation (neighbourhood adaptation is often controlled by a pre-specified scheme, unrelated to the weight adaptation); thus the neighbourhood can be omitted from the partial differentiation. This point has caused problems in interpreting the SOM cost function in the past.

It has however been argued that this energy function is violated at boundaries of Voronoi cells where input has exactly the same smallest distance to two neighbouring neurons. Thus this energy function holds mainly for the discrete case where the probability of such boundary input points is close to zero or the local (sample) cost function $\hat{E}_i$ should be used in deciding the winner [32]. When a spatial-invariant neighbourhood function is used (as is often the case), assigning the boundary input to either cell will lead to the same local sample cost (or error), therefore any input data on the boundary can be assigned to either Voronoi cells that have the same smallest distance to it, just as in the ordinary manner (on a first-come-first-served fashion, for example). Only when the neurons lie on the map borders does such violation occur, due to unbalanced neighbourhood neurons. The result is slightly more contraction towards to the centre (inside) of the map for the border neurons, compared to the common SOM algorithm, as shown in [50]. Using either the simple distance or local distortion measure as the winning rule will result in border neurons being contracted towards the centre of the map, especially when the map is not fully converged or when the effective range of the neighbourhood function is large. With the local distortion rule, this boundary effect is heavier as greater local error is incurred at the border neurons due to their few neighbouring neurons compared with any inside neurons.

To follow the cost function exactly, the winning rule should be modified to follow the local sample cost function $\hat{E}_i$ (or the local distortion measure) instead of the simplest nearest distance, that is,

$$\nu = \arg\min_i \sum_k \eta(i,k) \parallel \mathbf{x} - \mathbf{w}_k \parallel^2 \tag{21}$$

When the neighbourhood function is symmetric (as is often the case), and when the data density function is smooth, this local distortion winning rule is the same as the simplest nearest distance rule for most non-boundary nodes, especially if the number of nodes is large. On the map borders, however, differences exist due to the imbalance of nodes present in the neighbourhoods. Such differences become negligible to the majority of the neurons, especially when a large map is used, and when the neighbourhood function shrinks to its minimum scale.

3.2 Topological Ordering

The ordering to a large extent is still an outstanding and subtle issue, largely due to the fact that there is no clear (or agreed) definition of 'order' [25]. This is the very reason why a full self-organization convergence theorem including both statistical convergence, ordering, and the exact cost function, is still subject to debate – which has prompted many alternatives, such as [8], [26], and [100]. The ordering and ordered map are clearly defined only in the 1-D trivial case. Extension to higher dimensions proves difficult, if not impossible. [7] have proposed a measure called topology product to measure the topological ordering of the map,

$$P = \frac{1}{N^2 - N} \sum_i \sum_j \log \left(\prod_{l=1}^{j} \frac{d^D(\mathbf{w}_i, \mathbf{w}_{\eta^o(l,i)}) d^O(i, \eta^o(l,i))}{d^D(\mathbf{w}_i, \mathbf{w}_{\eta^D(l,i)}) d^O(i, \eta^D(l,i))} \right)^{\frac{1}{2k}} \tag{22}$$

where d^D and d^O represent the distance measures in the input (or data) space, and output (or map) space, respectively; $\eta(l,i)$ represents the lth neighbour of node i in either data (D) or map (O) space.

The first ratio in the product measures the ratio or match of weight distance sequences of a neighbourhood (up to j) on the map and in the data space. The second ratio is the index distance sequences of the neighbourhood on the map and in the data space. The topographic product measures the product of the two ratios of all possible neighbourhoods.

[109] proposed a topographic function to measure the 'neighbourhood-ness' of weight vectors in data space as well as on the lattice. The neighbourhood-ness of the weight vectors is defined by the adjacent Voronoi cells of the weights. The function measures the degree to which the weight vectors are ordered in the data space as to their indexes on the lattice, as well as how well the indexes are preserved when their weight vectors are neighbours.

Defining a fully ordered map can be straightforward using the distance relations [117]. For example, if all the nearest neighbour nodes (on the lattice) have their nearest neighbour nodes' weights in their nearest neighbourhood in the data space, we can call the map a 1st-order (ordered) map [117], that is,

$$d(\mathbf{w}_i, \mathbf{w}_j) \leq d(\mathbf{w}_i, \mathbf{w}_k), \quad \forall i \in \Omega; j \in \eta_i^1; k \notin \eta_i^1 \tag{23}$$

where Ω is the map, and η_i^1 denotes the 1st-order neighbourhood of node i.

Similarly if the map is a 1st-order ordered map, and all the 2nd nearest neighbouring nodes (on the lattice) have their 2nd nearest neighbouring nodes' weights in their 2nd nearest neighbourhood, we can call the map is a 2nd-order (ordered) map. For the 2nd ordered map, the distance relations to be satisfied are,

$$d(\mathbf{w}_i, \mathbf{w}_j) \leq d(\mathbf{w}_i, \mathbf{w}_k), \quad \forall i \in \Omega; j \in \eta_i^1; k \notin \eta_i^1 \& k \in \eta_i^2; l \notin \eta_i^2 \tag{24}$$

and so forth to define higher ordered maps with interneuron distance hierarchies [117].

An mth order map is optimal for tolerating channel noise spreading up to the mth neighbouring node. Such fully ordered maps however may not be always achievable, especially when the mapping is a dimensional reduction one. Then the degree (percentage) of nodes with their weights being ordered can be measured, together with the probabilities of the nodes being utilized, can be used to determine the topology preservation and that to what degree and to what order the map can tolerate the channel noise.

[25] proposed the C measure – a correlation between the similarity of stimuli in the data space and the similarity of their prototypes in the map space – to quantify the topological preservation,

$$C = \sum_i \sum_j F(i,j) G[M(i), M(j)] \tag{25}$$

where F and G are symmetric similarity measures in the input and map spaces respectively, and can be problem specific, and $M(i)$ and $M(j)$ are the mapped points or weight vectors of node i and j, respectively.

The C measure directly evaluates the correlation between distance relations between two spaces. Various other topographic mapping objectives can be unified under the C measure, such as multidimensional scaling, minimal wiring, the travelling salesperson problem (TSP), and noise tolerant VQ. It has also been shown that if a mapping that preserves ordering exists then maximizing C will find it. Thus the C measure is also the objective function of the mapping, an important property different from other topology preservation measures and definitions.

One can always use the underlying cost function Eqn. (18) to measure the goodness of the resulting map including the topology preservation, at least one can use a temporal window to take a sample of it as suggested in [50]. The (final) neighbourhood function specifies the level of topology (ordering) the mapping is likely to achieve or is required. To draw an analogy to the above C measure, the neighbourhood function can be interpreted as the G measure used in Eqn. (25) and the $\| \mathbf{x} - \mathbf{w}_k \|^2$ term represents the F measure. Indeed, the input x and weight w_j are mapped on the map as node index i and j, and their G measure is the neighbourhood function (for example, exponentials). Such an analogy also sheds light on the scaling effect of the SOM. Multidimensional scaling also aims to preserve local similarities on a mapped space (see the next Section for more details).

4 Extensions and Links with Other Learning Paradigms

The SOM has been a popular model for unsupervised learning as well as pattern organization and association. Since its introduction, various extensions have been reported to enhance its performance and scope. For instance, 'Neural Gas' was developed to map data onto arbitrary or hierarchical map structures rather than confined to rectangular grids for improved VQ performance [74, 75]. The adaptive subspace SOM (ASSOM) has been proposed to combine principal component learning and the SOM to map data with reduced feature space, in order to form translation-, rotation- and scale-invariant filters [51, 52]. The parameterized SOM (PSOM) has been proposed to extend SOM for continuous mapping using basis functions on the grid to interpolate the map [112]. The stochastic SOM [26] defines a topographic mapping from a Bayesian framework and a Markov chain encoder, and further explains the stochastic annealing effect in SOM. The Dislex [76, 77] applies hierarchical topographical maps to associate cross-modal patterns such as images with audio or symbols. The U-matrix was proposed to imprint the distance information on the map for visualization [105]. The visualization induce SOM (ViSOM) has been proposed to directly preserve distance information on the map so as to visualize data structures and distributions [118, 119].

The Temporal Kohonen map [11] and its improved version, the Recurrent SOM [108], as well as the Recursive SOM [110] have extended the SOM for mapping temporal data such as time series, or sequential data such as protein sequences. Extensions along this direction continue to be a focus of research. Extension on probabilistic approaches which enhances the scope and capability of SOM include the Self-Organizing Mixture Network (SOMN) [128], Kernel-based topographic maps [106, 107]; and the generic topographic mapping (GTM) [8]. There are many extensions developed in recent years – too many to completely list here. Recent extensions are also proposed for handling non-vectorial [55] and qualitative data [35]. For more comprehensive lists and recent developments, please refer to [1, 13, 38, 40, 53, 83].

The remainder of this Section covers extensions of SOM and their associations with data visualization, manifold mapping, density modeling and kernel methods.

4.1 SOM, Multidimensional Scaling and Principal Manifolds

The SOM is often associated with VQ and clustering. However it is also associated with data visualization, dimensionality reduction, nonlinear data projection, and manifold mapping. A brief review on various data projection methods and their relationships has been given before [121].

Multidimensional Scaling

Multidimensional scaling (MDS) is a traditional study related to dimensionality reduction and data projection. MDS tries to project data points onto an (often two-dimensional) sheet by preserving as closely as possible the inter-point metrics [14]. The projection is generally nonlinear and can reveal the overall structure of the data. A general fitness function or the so-called stress function is defined as,

$$S = \frac{\sum_{i,j}(d_{ij} - D_{ij})^2}{\sum_{i,j} D_{ij}^2} \tag{26}$$

where d_{ij} represents the proximity of data points i and j in the original data space, and D_{ij} represents the dissimilarity or distance (usually Euclidean) between mapped points i and j in the projected space. Note, that global Euclidean distance is usually used to calculate the inter-point distances. Recently, Isomap was proposed to use geodesic (curvature) distance instead for better nonlinear scaling [102].

MDS relies on an optimization algorithm to search for a configuration that gives as low a stress as possible. A gradient method is commonly used for this purpose. Inevitably, various computational problems – such as local minima and divergence – may occur due to the optimization process itself. The methods are also often computationally intensive. The final solution depends on the starting configuration and parameters used in the algorithm.

Sammon mapping is a widely-known example of MDS [95]. The objective is to minimize the differences between inter-point (Euclidean) distances in the original space and those in the projected plane. In Sammon mapping intermediate normalization (of the original space) is used to preserve good local distributions and at the same time maintain a global structure. The Sammon stress is expressed as,

$$S_{Sammon} = \frac{1}{\sum_{i<j} d_{ij}} \sum_{i<j} \frac{(d_{ij} - D_{ij})^2}{d_{ij}} \tag{27}$$

A second order Newton optimization method is used to recursively solve the optimal configuration. It converges faster than the simple gradient method, but the computational complexity is even higher. It still has local minima and inconsistency problems. Sammon mapping has been shown to be useful for data structure analysis. However, like other MDS methods, the Sammon algorithm is a point-to-point mapping, which does not provide an explicit mapping function and cannot naturally accommodate new data points. It also requires the computation and storage of all the inter-point distances. This proves difficult or even impossible for many practical applications where data arrives sequentially, the quantity of data is large, and/or memory space for the data is limited.

In addition to being computationally expensive, especially for large data sets, and not being adaptive, another major drawback of MDS is lack of an explicit projection function. Thus for any new input data, the mapping has to be recalculated based on all available data. Although some methods have been proposed to accommodate the new arrivals using triangulation [16, 61], the methods are generally not adaptive. However, such drawbacks can be overcome by implementing or parameterizing MDS using neural networks – for example, [65, 71].

Principal Component Analysis (PCA)

Principal Component Analysis (PCA) is a classic linear projection method aiming at finding orthogonal principal directions from a set of data, along which the data exhibiting the largest variances. By discarding the minor components, PCA can effectively reduce data variables and display the dominant ones in a linear, low dimensional subspace. It is an optimal linear projection in the sense of the mean-square error between original points and projected ones, in other words,

$$\min \sum_{\mathbf{x}} \left[\mathbf{x} - \sum_{j=1}^{m} (\mathbf{q}_j^T \mathbf{x}) \mathbf{q}_j \right]^2 \tag{28}$$

where $\{\mathbf{q}, j = 1, 2, \cdots, m, m \leq n\}$ are orthogonal eigenvectors representing principal directions. They are the first m principal eigenvectors of the covariance matrix of the input. The second term in the above bracket is the reconstruction or projection of x onto these eigenvectors. The term $\mathbf{q}_j^T\mathbf{x}$ represents the projection of x onto the jth principal dimension.

Traditional methods for solving the eigenvector problem involve numerical methods. Though fairly efficient and robust, they are not usually adaptive and often require presentation of the entire data set. Several Hebbian-based learning algorithms and neural networks have been proposed for performing PCA, such as the subspace network [81] and the generalized Hebbian algorithm [96]. The limitation of linear PCA is obvious, as it cannot capture nonlinear relationships defined by higher than second-order statistics. If the input dimension is much higher than two, the projection onto the linear principal plane will provide limited visualization power.

Nonlinear PCA and Principal Manifolds

Extension to nonlinear PCA (NLPCA) is not unique, due to the lack of a unified mathematical structure and an efficient and reliable algorithm, and in some cases due to excessive freedom in selection of representative basis functions [39, 70]. Several methods have been proposed for nonlinear PCA, such as the five-layer feedforward associative network [56] and the kernel PCA [97]. The first three layers of the associative network project the original data onto a curve or surface, providing an activation value for the bottleneck node. The last three layers define the curve and surface. The weights of the associative NLPCA network are determined by minimizing the following objective function,

$$\min \sum_{\mathbf{x}} \parallel \mathbf{x} - \mathbf{f}\{s_f(\mathbf{x})\} \parallel^2 \tag{29}$$

where $\mathbf{f} : R^1 \to R^n$ (or $R^2 \to R^n$). The function modelled by the last three layers defines a curve (or a surface), $s_f : R^n \to R^1$ (or $R^n \to R^2$); the function modelled by the first three layers defines the projection index.

The kernel-based PCA uses nonlinear mapping and kernel functions to generalize PCA to NLPCA and has been used for various pattern recognition tasks. The nonlinear function $\Phi(x)$ maps data onto high-dimensional feature space, where the standard linear PCA can be performed via kernel functions: $k(\mathbf{x}, \mathbf{y}) = (\Phi(\mathbf{x}) \cdot \Phi(\mathbf{y}))$. The projected covariance matrix is then,

$$Cov = \frac{1}{N} \sum_{i=1}^{N} \Phi(\mathbf{x}_i)\Phi(\mathbf{x}_i)^T \tag{30}$$

The standard linear eigenvalue problem can now be written as $\lambda\mathbf{V} = \mathbf{K}\mathbf{V}$, where the columns of $\mathbf{V}$ are the eigenvectors, and $\mathbf{K}$ is a $N \times N$ matrix with elements as kernels $k_{ij} := k(\mathbf{x}_i, \mathbf{x}_j) = (\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_i))$.

The principal curves and principal surfaces [28, 60] are the principal nonlinear extensions of PCA. The principal curve is defined as a smooth and self-consistency curve, which does not intersect itself. Denote $\mathbf{x}$ as a random vector in R^n with density p and finite second moment. Let $f(\cdot)$ be a smooth unit-speed curve in R^n, parameterized by the arc length (from one end of the curve) over $\Lambda \in R$, a closed interval.

For a data point $\mathbf{x}$ its projection index on f is defined as

$$\rho_f(\mathbf{x}) = \sup_{\rho \in \Lambda}\{\rho : \parallel \mathbf{x} - f(\rho) \parallel = \inf_{\theta} \parallel \mathbf{x} - f(\theta) \parallel\} \tag{31}$$

The curve is called self-consistent or a principal curve of ρ if

$$f(\rho) = E[\mathbf{X} \mid \rho_f(\mathbf{X}) = \rho] \tag{32}$$

The principal component is a special case of the principal curves if the distribution is ellipsoidal. Although principal curves have been mainly studied, extension to higher dimensions – for example principal surfaces or manifolds – is feasible in principle. However, in practice, a good implementation of principal curves/surfaces relies on an effective and efficient algorithm. The principal curves/surfaces are more of a concept that invites practical implementations. The HS algorithm is a nonparametric method [28], which directly iterates the two steps of the above definition. It is similar to the standard LGB VQ algorithm [63], combined with some smoothing techniques.

Algorithm 2 The Hastie and Stuetzle (HS) algorithm

```
Initialization: choose the first linear principal component as the initial curve, $f^{(0)}(\mathbf{x})$.
repeat
  Projection: project the data points onto the current curve and calculate the projections index – that is $\rho^{(t)}(x) = \rho_{f(t)}(\mathbf{x})$.
  Expectation: for each index, take the mean of the data points projected onto it as the new curve point – in other words, $f^{t+1}(\rho) = E[\mathbf{X} \mid \rho_{f(t)}\mathbf{X} = \rho]$.
until a convergence criterion is met (for example, when the change of the curve between iterations falls below a threshold).
```

For a finite data set, the density p is often unknown, and the above expectation is replaced by a smoothing method such as the locally weighted running-line smoother or smoothing splines. For kernel regression, the smoother is,

$$f(\rho) = \frac{\sum_{i=1}^{N} \mathbf{x}_i \mathcal{K}(\rho, \rho_i)}{\sum_{i=1}^{N} \mathcal{K}(\rho, \rho_i)} \tag{33}$$

The arc length is simply computed from the line segments. There are no proofs of convergence for the algorithm, but no convergence problems have been reported, although the algorithm is biased in some cases [28]. Banfield and Reftery have modified the HS algorithm by taking the expectation of the residual of the projections in order to reduce the bias [5]. [42] have proposed an incremental – for example, segment-by-segment – and arc length constrained method for practical construction of principal curves.

Tibshirani has introduced a semi-parametric model for the principal curve [103]. A mixture model was used to estimate the noise along the curve; and the expectation-maximization (EM) method was employed to estimate the parameters. Other options for finding the nonlinear manifold include the Generic Topographic Map [8] and probabilistic principal surfaces [10]. These methods model the data by a means of a latent space. They belong to the semi-parameterized mixture model, although types and orientations of the local distributions vary from method to method.

Visualization induced SOM (ViSOM)

For scaling and data visualization, a direct and faithful display of data structure and distribution is highly desirable. ViSOM has been proposed to extend the SOM for direct distance preservation on the map [118, 119], instead of using a colouring scheme such as U-matrix [105], which imprints qualitatively the inter-neuron distances as colours or grey levels on the map. For the map to capture the data structure naturally and directly, (local) distance quantities must be preserved on the map, along with the topology. The map can be seen as a smooth and graded mesh, or manifold embedded into the data space onto which the data points are mapped and the inter-point distances are approximately preserved.

In order to achieve that, the updating force, $\mathbf{x}(t)-\mathbf{w}_k(t)$, of the SOM algorithm is decomposed into two elements $[\mathbf{x}(t)-\mathbf{w}_\nu(t)]+[\mathbf{w}_\nu(t)-\mathbf{w}_k(t)]$. The first term represents the updating force from the winner ν to the input $\mathbf{x}(t)$, and is the same to the updating force used by the winner. The second force is a lateral contraction force bringing neighbouring neuron k to the winner ν. In the ViSOM, this lateral contraction force is constrained or regulated in order to help maintain unified local inter-neuron distances $\parallel \mathbf{w}_\nu(t)-\mathbf{w}_k(t) \parallel$ on the map.

$$\mathbf{w}_k(t+1)=\mathbf{w}_k(t)+\alpha(t)\eta(v,k,t)[\mathbf{x}(t)-\mathbf{w}_\nu(t)]+\beta[\mathbf{w}_\nu(t)-\mathbf{w}_k(t)] \qquad (34)$$

where the simplest constraint can be $\beta=\frac{d_{\nu k}}{(D_{\nu k}\lambda)-1}$, with $d_{\nu k}$ being the distance of neuron weights in the input space, $D_{\nu k}$ the distance of neuron indexes on the map, and λ a (required) resolution constant.

ViSOM regularizes the contraction force so that the distances between nodes on the map are analogous to the distances of their weights in the data space. The aim is to adjust inter-neuron distances on the map in proportion to those in the data space, in other words $D_{vk} \propto d_{vk}$. When the data points are eventually projected onto a trained map, the distance between point i and j on the map is proportional to that of the original space, subject to the quantization error (the distance between a data point and its neural representative). This has a similar effect to MDS, which also aims at achieving this proportionality, $D_{ij} \propto d_{ij}$.

The SOM is shown to be a qualitative scaling, while the ViSOM is a metric scaling [124]. The key feature of ViSOM is that the distances between the neurons (which data are mapped to) on the map (in a neighbourhood) reflect the corresponding distances in the data space. When the map is trained and data points mapped, the distances between mapped data points will resemble approximately those in the original space (subject to the resolution of the map). This makes visualization more direct, quantitatively measurable, and visually appealing. The map resolution can be enhanced (and the computational cost reduced) by interpolating a trained map or incorporating local linear projections [122]. The size or covering range of the neighbourhood function can also be decreased from an initially large value to a final smaller one. The final neighbourhood, however, should not contain just the winner. The rigidity or curvature of the map is controlled by the ultimate size of the neighbourhood. The larger this size, the flatter the final map is in the data space. Guidelines for setting these parameters have been given in [120]. An example on data visualization will be shown in the next Section.

Several authors have since introduced improvements and extensions to ViSOM. For example, in [116], a probabilistic data assignment [26] is used in both the input assignment and the neighbourhood function; also an improved second order constraint is adopted. The resulting SOM has a clearer connection to an MDS cost function. Estévez and Figueora extend the ViSOM to an arbitrary, neural gas type of map structure [21]. Various other variants of SOM, such as hierarchical, growing, and hierarchical and growing structures are readily extendable to the ViSOM for various application needs.

The SOM has been related to the discrete principal curve/surface algorithm [91]. However differences remain in both the projection and smoothing processes. In the SOM the data are projected onto the nodes rather than onto the curve. The principal curves perform the smoothing entirely in the data space – see Eqn. (33). The smoothing process in SOM and ViSOM, as a convergence criterion, is [120],

$$\mathbf{w}_k = \frac{\sum_{i=1}^{L} \mathbf{x}_i \eta(\nu, k, i)}{\sum_{i=1}^{L} \eta(\nu, k, i)} \tag{35}$$

Smoothing is governed by the indexes of the neurons in the map space. The kernel regression uses the arc length parameters (ρ, ρ_i) or $\| \rho - \rho_i \|$ exactly, while the neighbourhood function uses the node indexes (ν, k) or $\| \mathbf{r}_\nu - \mathbf{r}_k \|$. Arc lengths reflect the curve distances between the data points. However, node indexes are integer numbers denoting the nodes or positions on the map grid, not the positions in the input space. So $\| \mathbf{r}_\nu - \mathbf{r}_k \|$ does not resemble $\parallel \mathbf{w}_\nu - \mathbf{w}_k \parallel$ in the common SOM. In the ViSOM, however, as the local inter-neuron distances on the map represent those in the data space (subject to the map resolution), the distances of nodes on the map are in proportion to the difference of their positions in the data space, that is $\parallel \mathbf{r}_\nu - \mathbf{r}_k \parallel \sim \parallel \mathbf{w}_\nu - \mathbf{w}_k \parallel$. The smoothing process in the ViSOM resembles that of the principal curves as shown below,

$$\mathbf{w}_k = \frac{\sum_{i=1}^{L} \mathbf{x}_i \eta(\nu, k, i)}{\sum_{i=1}^{L} \eta(\nu, k, i)} \approx \frac{\sum_{i=1}^{L} \mathbf{x}_i \eta(\mathbf{w}_\nu, \mathbf{w}_k, i)}{\sum_{i=1}^{L} \eta(\mathbf{w}_\nu, \mathbf{w}_k, i)} \tag{36}$$

This shows that ViSOM is a better approximation to the principal curves/surfaces than the SOM. SOM and ViSOM are similar only when the data are uniformly distributed, or when the number of nodes becomes very large, in which case both SOM and ViSOM will closely approximate the principal curves/surfaces.

4.2 SOM and Mixture Models

The SOM has been linked with density matching models and the point density that the SOM produces is related to the density of the data. However the SOM does not exactly follow the data density. Such properties have been studied and treated under the VQ framework [17,54,67,89,117].

The self-organizing mixture network (SOMN) [128] extends and adapts the SOM to a mixture density model, in which each node characterizes a conditional probability distribution. The joint probability density of the data or the network output is described by a mixture distribution,

$$p(\mathbf{x} \mid \Theta) = \sum_{i=1}^{K} p_i(\mathbf{x} \mid \theta_i) P_i \tag{37}$$

where $p_i(\mathbf{x} \mid \theta_i)$ is the ith component-conditional density, and θ_i is the parameter for the ith conditional density, $i = 1, 2, \cdots, K; \Theta = (\theta_1, \theta_2, \cdots, \theta_K)^T$, and P_i is the prior probability of the ith component or node and is also called the mixing weights. For example, a Gaussian mixture has the following the conditional densities respectively,

$$p_i(\mathbf{x} \mid \theta_i) = \frac{1}{(2\pi)^{d/2} \mid \sum_i \mid^{1/2}} exp \left[-\frac{1}{2} (\mathbf{x} - \mathbf{m}_i)^T {\sum_i}^{-1} (\mathbf{x} - \mathbf{m}_i) \right] \tag{38}$$

where $\theta_i = \{\mathbf{m}_i, \sum_i\}$ are the mean vector and covariance matrix, respectively.

Suppose that the true environmental data density function and the estimated one are $p(\mathbf{x})$ and $\hat{p}(\mathbf{x})$, respectively. The Kullback-Leibler information distance measures the divergence between these two, and is defined as,

$$\mathbf{I} = -\int log\frac{\hat{p}(\mathbf{x})}{p(\mathbf{x})}p(\mathbf{x})d\mathbf{x} \tag{39}$$

It is always positive and is equal to zero only when two densities are identical.

When the estimated density is modelled as a mixture distribution, one can seek the optimal estimate of the parameters by minimizing the Kullback-Leibler divergence via its partial differentials in respect to model parameters, more specifically,

$$\frac{\partial \mathbf{I}}{\partial \theta_i} = -\int \left[\frac{1}{\hat{p}(\mathbf{x} \mid \hat{\Theta})}\frac{\partial \hat{p}(\mathbf{x} \mid \hat{\Theta})}{\partial \theta_i}\right] p(\mathbf{x})d\mathbf{x}, \qquad i = 1, 2, \cdots, K \tag{40}$$

As the true data density is not known, the stochastic gradient is used for solving these non-directly solvable equations. This results in the following adaptive update rules for the parameters and priors [129],[2]

$$\begin{aligned}\hat{\theta}_i(t+1) &= \hat{\theta}_i(t) + \alpha(t)\eta(\nu(\mathbf{x}), i)\left[\frac{1}{\hat{p}(\mathbf{x} \mid \hat{\Theta})}\frac{\partial \hat{p}(\mathbf{x} \mid \hat{\Theta})}{\partial \theta_i}\right] \\ &= \hat{\theta}_i(t) + \alpha(t)\eta(\nu(\mathbf{x}), i)\left[\frac{\hat{P}_i(t)}{\sum_j \hat{P}_i(t)\hat{p}_j(\mathbf{x} \mid \theta_j)}\frac{\partial \hat{p}_i(\mathbf{x} \mid \hat{\Theta}_i)}{\partial \theta_i}\right]\end{aligned} \tag{41}$$

and

$$\begin{aligned}\hat{P}_i(t+1) &= \hat{P}_i(t) + \alpha(t)\left[\frac{\hat{p}_i(\mathbf{x} \mid \hat{\theta}_i)\hat{P}_i(t)}{\hat{p}(\mathbf{x} \mid \hat{\Theta})} - \hat{P}_i(t)\right] \\ &= \hat{P}_i(t) - \alpha(t)\eta(\nu(\mathbf{x}), i)\left[\hat{P}(i \mid \mathbf{x}) - \hat{P}_i(t)\right]\end{aligned} \tag{42}$$

where $\alpha(t)$ is the learning coefficient or rate at time step t $(0 < \alpha(t) < 1)$, and decreases monotonically. The winner is found via the maximum *posterior* probability of the node,

$$\hat{P}(i \mid \mathbf{x}) = \frac{\hat{P}_i\hat{p}_i(\mathbf{x} \mid \hat{\theta}_i)}{\sum_j \hat{p}(\mathbf{x} \mid \hat{\Theta})} \tag{43}$$

When the SOMN is limited to the homoscedastic case – namely equal variances and equal priors (non-informative priors) for all components – only the means are the learning variables. The above winner rule becomes,

$$v = \arg\max_i \frac{\hat{p}_i(\mathbf{x} \mid \hat{\theta}_i)}{\sum_j \hat{p}_j(\mathbf{x} \mid \hat{\theta}_j)} \tag{44}$$

[2] A common neighbourhood function, $\eta(\nu(\mathbf{x}), i)$, can be added as in the SOM, but is optional.

When the conditional density function is isotropic or symmetric or is a function of $\| \mathbf{x} - \mathbf{m} \|$, the above winning rule is a function of commonly used Euclidean norm $\| \mathbf{x} - \mathbf{m} \|$. The corresponding weight updating rule is,

$$\mathbf{m}_i(t+1) = \mathbf{m}_i(t) + \alpha(t)\eta(\nu(\mathbf{x}), i)\frac{1}{\sum_j p_j(\mathbf{x} \mid \theta_j)}\frac{\partial p_i(\mathbf{x} \mid \theta_i)}{\partial \mathbf{m}_i} \tag{45}$$

For example, for a Gaussian mixture with equal variance and prior for all nodes, it is easy to show that the winning and mean update rules become,

$$v = \arg\max_i \left[exp\left(-\frac{\| \mathbf{x} - \mathbf{m}_i \|^2}{2\sigma^2} \right) \right] \tag{46}$$

and

$$\mathbf{m}_i(t+1) \\ = \mathbf{m}_i(t) + \alpha(t)\eta(\nu(\mathbf{x}), i)\frac{1}{2\sigma^2}\frac{1}{\sum_j p_j(\mathbf{x} \mid \theta_j)} exp\left(-\frac{\| \mathbf{x} - \mathbf{m}_i \|^2}{2\sigma^2} \right)(\mathbf{x} - \mathbf{m_i}) \tag{47}$$

The winning rule becomes equivalent to the simple distance measure. The update formula bear a similarity to that of the original SOM. The term $exp\left(-\frac{\|\mathbf{x}-\mathbf{m}_i\|^2}{2\sigma^2}\right)$ is playing a similar role as the neighbourhood function, defined by the distances between weights and input instead of node indexes. The SOM approximates it by quantizing it using node indexes. The SOMN is also termed a 'Bayesian SOM', as it applies the Bayesian learning principle to the SOM learning rule [129].

4.3 SOM and Kernel Method

A kernel is a function $\mathcal{K} : X \times X \rightarrow \mathcal{R}$, where X is the input space. This function is a dot product of the mapping function $\phi(\mathbf{x})$ – in other words $\mathcal{K}(\mathbf{x}; \mathbf{y}) = [\phi(\mathbf{x}), \phi(\mathbf{y})]$, where $\phi : X \rightarrow F, F$ being a high dimensional inner product feature space. The mapping function $\phi(\mathbf{x})$ is often nonlinear and not known. All the operations are defined in terms of the kernel function instead of the mapping function. The kernel methodology has become increasingly popular within supervised learning paradigms, with the Support Vector Machine (SVM) being a widely known example. When nonlinearly mapping data or patterns to high dimensional space, the patterns often become linearly separable.

The kernel method has also been applied to the SOM. Following the kernel PCA [97], a k-means based kernel SOM has been proposed [69]. Each data point x is mapped to the feature space via a (unknown or imaginary) nonlinear function $\phi(\mathbf{x})$. In principle each mean can be described as a weighted sum of the observations in the feature space $\mathbf{m}_i = \sum_n \gamma_{i,n}\phi(\mathbf{x}_n)$, where $\{\gamma_{i,n}\}$ are the constructing coefficients. The algorithm then selects a mean or assigns a data point with the minimum distance between the mapped point and the mean,

$$\begin{aligned}\| \phi(\mathbf{x} - \mathbf{m}_i \|^2 &= \| \phi(\mathbf{x} - \sum_n \gamma_{i,n}\phi(\mathbf{x}_n) \|^2 \\ &= \mathcal{K}(\mathbf{x},\mathbf{x}) - 2\sum_n \gamma_{i,n}\mathcal{K}(\mathbf{x},\mathbf{x}_n) + \sum_{n,m} \gamma_{m,n}\mathcal{K}(\mathbf{x}_n,\mathbf{x}_m) \quad (48)\end{aligned}$$

The update of the mean is based on a soft learning algorithm,

$$\mathbf{m}_i(t+1) = \mathbf{m}_i(t) + \Lambda[\phi(\mathbf{x} - \mathbf{m}_i(t)] \quad (49)$$

where Λ is the normalized winning frequency of the ith mean and is defined as,

$$\Lambda = \frac{\zeta_{i(\mathbf{x}),j}}{\sum_{n=1}^{t+1} \zeta_{i,n}} \quad (50)$$

where ζ is the winning counter and is often defined as a Gaussian function between the indexes of the two neurons.

As the mapping function $\phi(\mathbf{x})$ is not known, the update rule (Eqn. (49)) is further elaborated and leads to the following updating rules for the constructing coefficients of the means [69],

$$\gamma_{i,n}(t+1) = \begin{cases} \gamma_{i,n}(t)(1-\zeta), & \text{for} \quad n \neq t+1 \\ \zeta, & \text{for} \quad n = t+1 \end{cases} \quad (51)$$

Note that these constructing coefficients, $\{\gamma_{i,n}\}$, together with the kernel function, effectively define the kernel SOM in feature space. The winner selection – that is, Eqn. (48) – operates on these coefficients and the kernel function. No *explicit* mapping function $\phi(\mathbf{x})$ is required. The exact means or neurons' weights – $\{\mathbf{m}_i\}$ – are not required.

There is another, direct way to kernelize the SOM by mapping the data points and neuron weights, both defined in the input space, to a feature space, then applying the SOM in the mapped dot product space. The winning rules of this second type of kernel SOM have been proposed as follows, either in the input space [85],

$$v = \arg\min_i \| \mathbf{x} - \mathbf{m}_i \| \quad (52)$$

or in the feature space [4],

$$v = \arg\min_i \| \phi(\mathbf{x}) - \phi(\mathbf{m}_i) \| \quad (53)$$

It will soon become clear that these two rules are equivalent for certain kernels, such as the Gaussian. The weight update rule proposed by [4] is,

$$\mathbf{m}_i(t+1) = \mathbf{m}_i(t) + \alpha(t)\eta(v(\mathbf{x}), i)\Delta J(\mathbf{x}, \mathbf{m}_i) \tag{54}$$

where $J(\mathbf{x}, \mathbf{m}_i) = \parallel \phi(\mathbf{x}) - \phi(\mathbf{m}_i) \parallel^2$ is the distance function in the feature space or the proposed instantaneous or sample objective function. and are the learning rate and neighbourhood function, respectively.

Note that,

$$J(\mathbf{x}, \mathbf{m}_i) = \parallel \phi(\mathbf{x}) - \phi(\mathbf{m}_i) \parallel^2 = \mathcal{K}(\mathbf{x}, \mathbf{x}) + \mathcal{K}(\mathbf{m}_i, \mathbf{m}_i) - 2\mathcal{K}(\mathbf{x}, \mathbf{m}_i) \tag{55}$$

and,

$$\nabla J(\mathbf{x}, \mathbf{m}_i) = \frac{\partial \mathcal{K}(\mathbf{m}_i, \mathbf{m}_i)}{\partial \mathbf{m}_i} - 2\frac{\partial \mathcal{K}(\mathbf{x}, \mathbf{m}_i)}{\partial \mathbf{m}_i} \tag{56}$$

Therefore this kernel SOM can also be operated entirely in the feature space with the kernel function. As the weights of the neurons are defined in the input space, they can be explicitly resolved.

These two kernel SOMs have been proved equivalent ([59], [123]); they can all be derived from applying the energy function (Eqn. (18)) on the mapped feature space,

$$E_F = \sum_i \int_{v_i} \sum_j \eta(i,j) \parallel \phi(\mathbf{x}) - \phi(\mathbf{m}_j) \parallel^2 p(\mathbf{x}) d\mathbf{x} \tag{57}$$

The kernel SOM can be seen as a result of directly minimizing this transformed energy stochastically, in other words, by using the sample gradient on $\sum_j \eta(v(\mathbf{x}), j) \parallel \phi(\mathbf{x}) - \phi(\mathbf{m}_j) \parallel^2$,

$$\frac{\partial \hat{E}_F}{\mathbf{m}_i} = \frac{\partial}{\mathbf{m}_i} \sum_j \eta(\nu(\mathbf{x}), j) \parallel \phi(\mathbf{x}) - \phi(\mathbf{m}_j) \parallel^2 = -2\eta(\nu(\mathbf{x}), i)\nabla J(\mathbf{x}, \mathbf{m}_j) \tag{58}$$

This leads to the same weight update rule of the kernel SOM as Eqn. (54).

Various kernel functions such as Gaussian (or radial basis function), Cauchy and polynomial, are readily applicable to the kernel SOM [59]. For example, for Gaussian kernel, the winning and weight update rules are,

$$\begin{aligned} v = \arg\min_i J(\mathbf{x}, \mathbf{m}_i) &= \arg\min_i [-2\mathcal{K}(\mathbf{x} - \mathbf{m}_i)] \qquad (59) \\ &= \arg\min_i \left[-exp\left(-\frac{\parallel \mathbf{x} - \mathbf{m}_i \parallel^2}{2\sigma^2} \right) \right] \end{aligned}$$

and,

$$\mathbf{m}_i(t+1) = \mathbf{m}_i(t) + \alpha(t)\eta(\nu(\mathbf{x}), i)\frac{1}{2\sigma^2} exp\left(-\frac{\parallel \mathbf{x} - \mathbf{m}_i \parallel^2}{2\sigma^2} \right) (\mathbf{x} - \mathbf{m}_i) \tag{60}$$

respectively. Please note for Gaussian kernel functions, although the winning rule (Eqn. (59)) is derived from the feature space, it is equivalent to that of the original SOM and is conducted in the input space.

Comparing the kernel SOM algorithm (Eqns. (59) and (60)) with those of the SOMN (Eqns. (46) and (47)), it can be easily seen that the two methods are the same [123]. That is, the kernel SOM (with Gaussian kernels) is implicitly applying a Gaussian mixture to model the data. In other words, the SOMN is a kind of kernel method. As the SOM is seen as a special case of the SOMN, the original SOM has a certain effect of the kernel method.

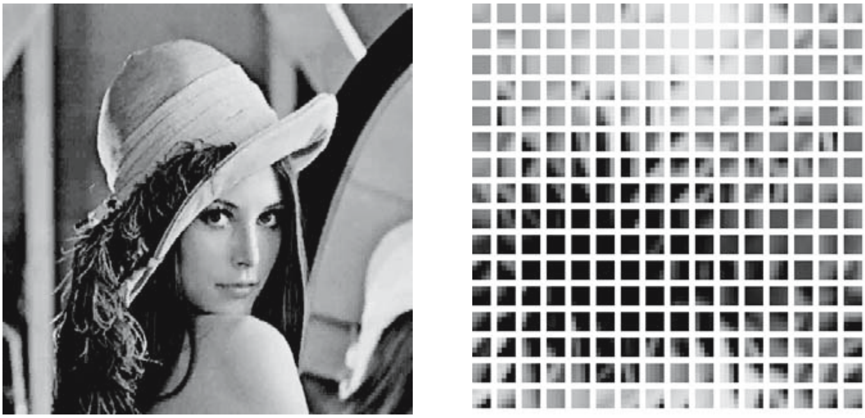
5 Applications and Case Studies

Thousands of applications of the SOM and its variants have been reported since its introduction [40,53,83] – too many to list here. There is a dedicated international Workshop on SOMs (WSOM), as well as focused sessions in many neural network conferences. There have also been several special journal issues dedicated to advances in SOM and related topics [1,13,38]. Moreover, many new applications are being reported in many relevant journals today. SOMs will remain an active topic in their continued extension, combination and applications in the years to come.

In this Section, several typical applications are provided as case studies. They include image and video processing; density or spectrum profile modeling; text/document mining and management systems; gene expression data analysis and discovery; and high dimensional data visualizations. Other typical applications not discussed here include image/video retrieval systems – for instance, PicSOM [57]; nonlinear ICA (Nonlinear PCA and ICA) [27,31,37,82,84]; classification (LVQ) [53]; cross-modal information processing and associations [76,77]; novelty detection [73]; robotics [6]; hardware implementation [98]; and computer animation [113].

5.1 Vector Quantization and Image Compression

The SOM is an optimal VQ when the neighbourhood eventually shrinks to just the winner, as it will satisfy the two necessary conditions for VQ (Voronoi partition and centroid condition). The use of the neighbourhood function makes the SOM superior to common VQs in two main respects. Firstly, the SOM is better at overcoming the under- or over-utilization and local minima problem. The second is that the SOM will produce a map (codebook) with some ordering (even when the neighbourhood eventually vanishes) among the code vectors, and this gives the map an ability to tolerate noise in the input or retrieval patterns. An example is provided in Fig. 6, in which (b) shows the 16×16 codebook trained on the Lena test image of 512×512 pixels by SOM with distinctive ordering found among the code vectors; and (a) shows the quantized Lena image by the trained codebook. The code vectors are of 4×4 pixel blocks.



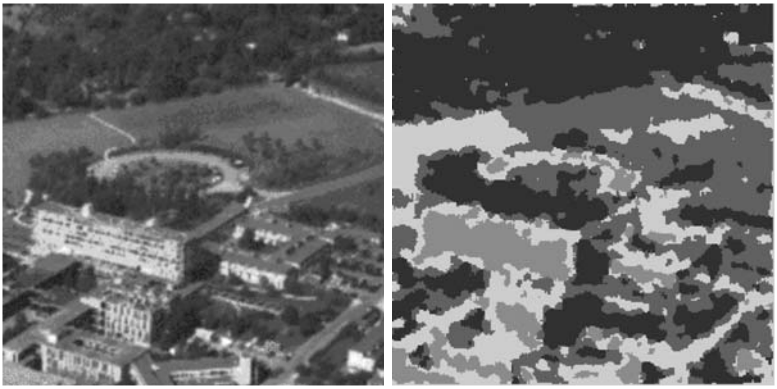
(a) (b)

Fig. 6. (**a**) Quantized Lena image; (**b**) the SOM codebook (map)

It has been found that SOMs generally perform better than other VQs especially in situations where local optima are present [117]. The robustness of SOM has been further improved by introducing a constraint on the learning extent of a neuron based on the input space variance it covers. The algorithm is aiming to achieve global optimal VQ by limiting and unifying the distortions from all nodes to approximately equal amounts – the asymptotic property of the global optimal VQ (in other words, for a smooth underlying probability density and large number of code vectors as all regions in an optimal Voronoi partition have the same within region variance). The constraint is applied to the scope of the neighbourhood function so that the node covering a large region (thus having a large variance) has a large neighbourhood. The results show that the resulting quantization error is smaller. Such a SOM-based VQ has also been applied to video compression [2, 22] for improved performance at low bit rates.

5.2 Image Segmentation

The SOM has been used in a hierarchical structure, together with the Markov random field (MRF) model, for the unsupervised segmentation of textured images [125]. The MRF is used as a measure of homogeneous texture features from a randomly placed local region window on the image. Such features are noisy and poorly known. They are input to a first SOM layer, which learns to classify and filter them. The second local-voting layer – a simplified SOM – produces an estimate of the texture type or label for the region. The hierarchical network learns to progressively estimate the texture model, and classify the various textured regions of similar type. Randomly positioning the local window at each iteration ensures that the consecutive inputs to the SOMs are uncorrelated. The size of the window is large at the beginning to capture patch-like texture homogeneities and shrinks with time to reduce the



(a) (b)

Fig. 7. (**a**) An aerial image; (**b**) segmented image using the SOM and Markov random field

estimation parameter noise at texture boundaries. The weights of the neurons in the first layer will converge to the MRF model parameters of various texture types, whilst the weights of the second layer will be the prototypes of these types – that is, the segmented image. The computational form of the entire algorithm is simple and efficient. The theoretical analysis of the algorithm shows that it will converge to the maximum likelihood segmentation. Figure 7 shows a typical example of such applications. The number of texture types was subjectively assumed as four. Interestingly, the algorithm has segmented the image into four meaningful categories: 'trees', 'grass', 'buildings', and 'roads'.

5.3 Density Modeling

Some function profiles (such as spectra) can be considered as density histograms. If a spectrum consists of many components, then the SOMN described in Sect. 4.2 can be used to estimate the component profiles of the spectrum [128, 129]. Re-sampling the observed spectrum will provide distribution data for training. The x-ray diffraction patterns of crystalline complex organic molecules (such as proteins) consist of a large number of Bragg diffraction spots. These patterns represent the intensity Fourier transform of the molecular structure (actually the electron density maps); the crystallographers need to determine the precise position of each spot together with its magnitude (namely, integrated spot intensity). The patterns exhibit relatively high background noise together with spot spreading (due to shortcomings in the experiments or limitations in the detection processes), which results in overlapping spots. Automatic analysis of these patterns is a non-trivial task. An example of such a pattern image is shown in Fig. 8(a), which is an 8-bit greyscale and of size 88 × 71 pixels. In order for the SOMN to learn the profiles of these diffraction spots, the image (diffraction intensity function) has to be

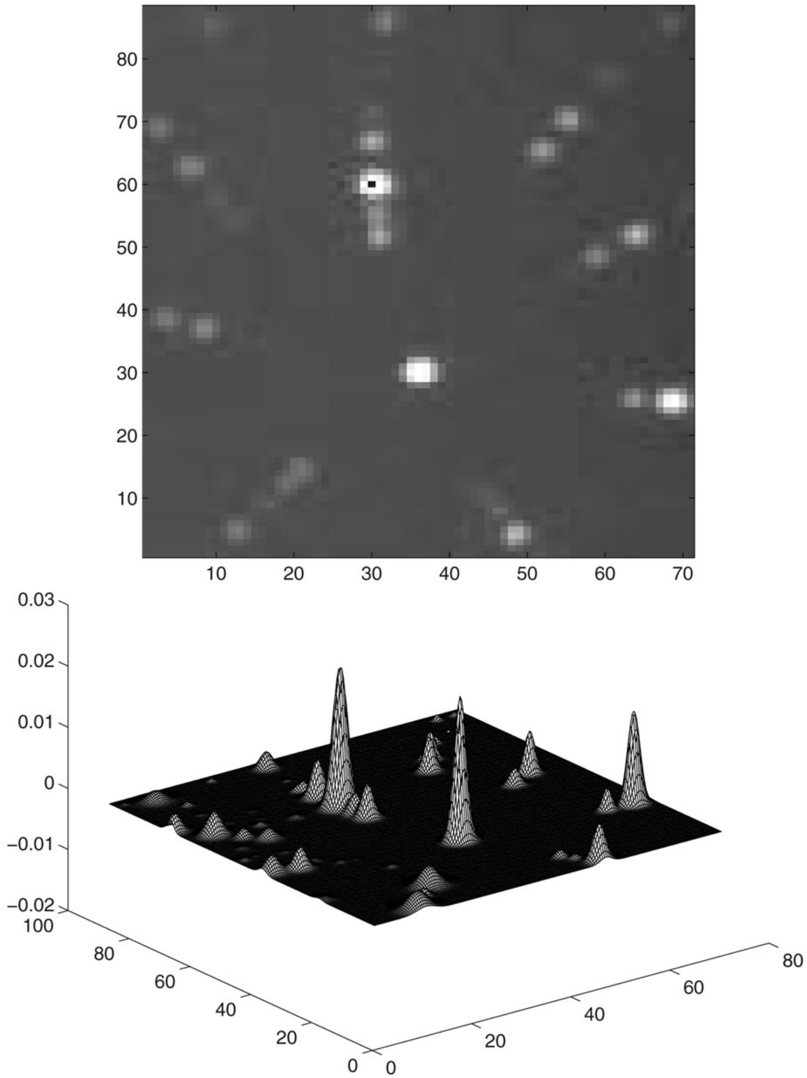


Fig. 8. (**a**) X-ray diffraction pattern (part); (**b**) modelled profiles by a 20 × 20 SOMN (from [129])

re-sampled to provide distribution data. A set of training data (10,647 points in total) was obtained by double sampling this image. A 400-neuron SOMN, arranged in a 20 × 20 grid, was used to learn this density. In this case, the number of spots (peaks or components) in a pattern (a mixture) will not generally be known *a priori*. The initial positions of the neurons were regularly placed inside the data space – in other words, a $[1, 88] \times [1, 71]$ rectangular grid. The initial variances were assigned equally to a diagonal matrix with the diagonal values equal to a fraction of the grid size, and the initial mixing priors were assigned equally to 1/400. The grid was pre-ordered to save unnecessary computational cost as this is a mapping with the same dimension.

After a few learning cycles, the SOMN allocated the spots to the Gaussian kernels and decomposed the overlapping ones. Individual neurons and their parameters provide centre (mean vectors) and width (covariance matrices) information for relevant spots. The total intensity of each peak is readily obtainable and is simply related to its mixing weight. The result of the estimation after five epochs is shown in Fig. 8(b). The number of active nodes (that is, surviving ones) is much less than the initial guess of 400. The SOMN has dynamically fitted to the correct mixture number and suppressed others. As the updating at each input was limited to a small area (3×3 – the winner and its first order neighbourhood, in this example), the SOMN required a much lighter computational effort than updating the entire network at each input (as the EM algorithm would). This becomes particularly advantageous when the number of the nodes is large. In this example, The EM algorithm of the same size would require approximately $400/(3 \times 3) \approx 44$ times more computing effort than the SOMN.

5.4 Gene Expression Analysis

The SOM has been applied as a valid and useful tool for clustering gene expressions [80,104]. Several attempts have been made to deal with ordered sequence or temporal sequences using SOM. A common approach is to use the trajectories of consecutive winning nodes on the SOM. Other methods are based on modification of the learning topology by introducing recurrent connections, for example the Temporal Kohonen Maps (TKM) or Recurrent SOM (RSOM) mentioned in Sect. 4. In TKM the participation of earlier input vectors in each unit is represented by using a recursive difference equation which defines the current unit activity as a function of the previous activations and the current input vector. In the RSOM, which is a modification of the TKM, the scalar node activities of the TKM are replaced by difference vectors defined as a recursive difference equation of the new input, the previous difference vectors, and the weight vectors of the units. One potential problem with recurrent models is stability. In the case of temporal gene expression clustering, the data items presented to the map are not a spatial vector, but a sequence with time order in itself. They are time-series corresponding to the expression levels over time of a particular gene. Therefore, if a common 2-D SOM is used, the trained map can then be used to mark the trajectories of the expressions of the genes for comparison purposes.

We approach the temporal extension of SOM from another perspective, this being the similarity metric. If the similarity metric takes into account temporal properties, then the neurons in the resultant map will exhibit temporal relationships. As time is one dimensional, a 1-D SOM is more appropriate.

In addition, a circular (that is, a closed 1-D SOM), can further detect cyclic temporal characteristics [80]. A novel temporal metric termed the co-expression coefficient has been defined as [79],

$$ce(x,y) = \frac{\int x' y' dt}{\sqrt{\int x'^2 dt \int y'^2 dt}} \tag{61}$$

where x and y are two (often modelled, thus smoothed) gene expression profiles; x' and y' are their derivatives. It can be seen that the co-expression coefficient is the correlation coefficient of the derivatives of the profiles. Comparing the derivatives of the two better profiles rather than directly on the two profiles captures their temporal properties.

Two yeast cell cycle datasets (208 and 511 genes) were modelled using RBFs and then the modelled profiles were differentiated [80]. The Bayesian information criterion was used to validate the number of clusters obtained by the circular SOM. Each node presents the smaller distance only to its two neighbouring nodes in a chain-ordered fashion, this implies that characteristic traits are split or merged with larger or fewer number of clusters without changing the order or relation between them. Figure 9 presents the resulting SOMs and prototype profiles of the clusters. It can be easily seen that topology exists among the profiles. The topological order here refers to the time shift. This demonstrates that the proposed method is able to group profiles based on their temporal characteristics and can automatically order the groups based on their periodical properties.

Genes identified as cell-cycle-regulated by traditional biological methods have been used to evaluate the performance of the proposed technique. The result shows that the obtained clusters have high relevance to the distribution of these genes among the cell cycle phases identified by biological methods, compared with other clustering methods [80].

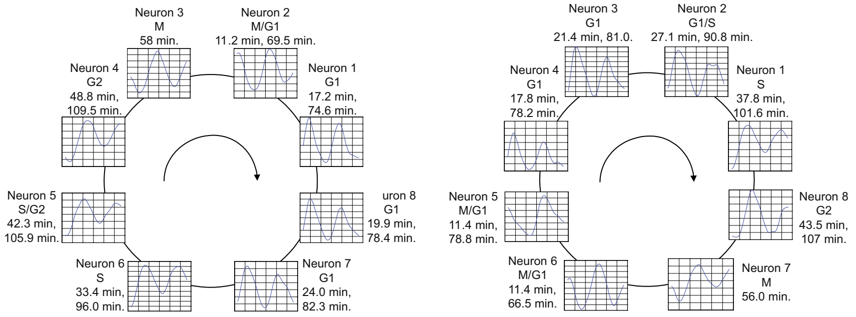


Fig. 9. Circular SOMs for clustering temporal gene expressions (yeast cell cycle dataset): (**a**) 208-gene dataset; (**b**) 511-gene dataset (from [80])

5.5 Data Visualization

Data projection and visualization has become a major application area for neural networks, in particular for the SOMs [53], as its topology preserving property is unique among other neural models. Good projection and visualization methods help to identify clustering tendency, to reveal the underlying functions and patterns, and to facilitate decision support. A great deal of research has been devoted to this subject, and a number of methods have been proposed. A recent review on this subject can be found in [121].

The SOM has been widely used as a visualization tool for dimensionality reduction (for instance, [34,41,53,105]). The SOM's unique topology preserving property can be used to visualize the relative mutual relationships among the data. However, the SOM does not directly apply to scaling, which aims to reproduce proximity in (Euclidean) distance on a low visualization space, as it has to rely on a colouring scheme (for example, the U-matrix method [105]) to imprint the distances crudely on the map. Often the distributions of the data points are distorted on the map. The recently proposed ViSOM [118–120], described in Sect. 4.1, constrains the lateral contraction force between the neurons in the SOM and hence regularizes the inter-neuron distances with respect to a scalable parameter that defines and controls the resolution of the map. It preserves the data structure as well as the topology as faithfully as possible. ViSOM provides a direct visualization of both the structure and distribution of the data. An example is shown in Fig. 10, where a 100 × 100

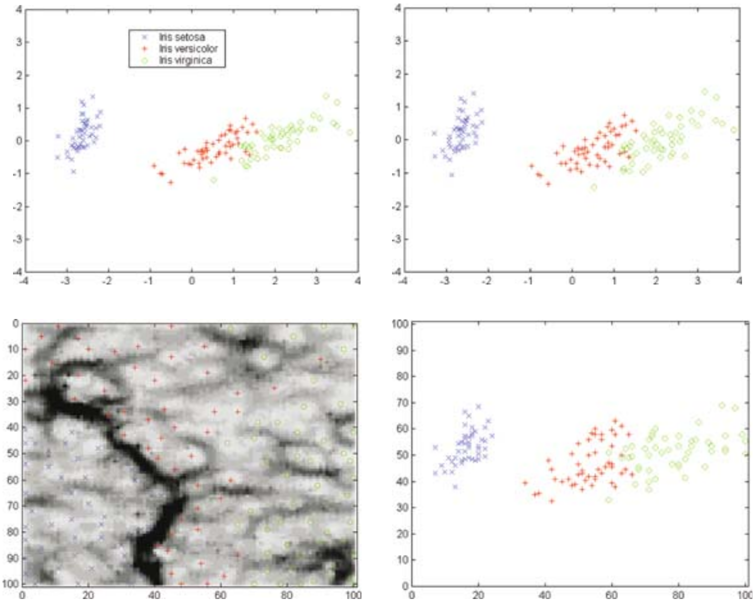


Fig. 10. Mapping and visualization of the iris data set: (*top left*) PCA, (*top right*) Sammon mapping; (*bottom left*) SOM with U matrix Colouring, (*bottom right*) ViSOM

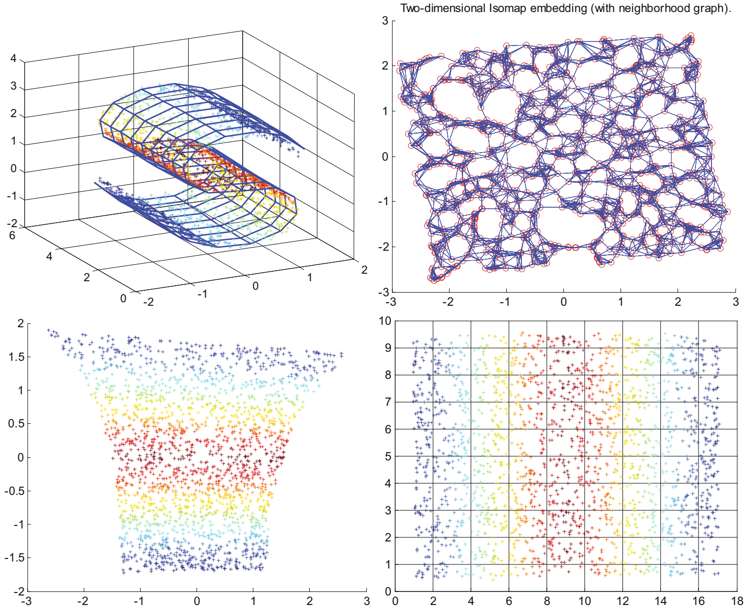


Fig. 11. Manifold mapping by various methods: (*top left*) original S-shape data and ViSOM embedding, (*top right*) Isomap projection; (*bottom left*) LLE projection, (*bottom right*) ViSOM projection

(hexagonal) ViSOM was used to map the 4-D Iris data set; it gives direct visualization of data distribution, similar to Sammon mapping. Although, the SOM with colouring can show the gap between iris setosa and the rest, it is impossible to capture the data structure and represent the data proximity on the map.

Usually for a fine mapping, the resolution parameter needs to be set to a small value. Moreover, a large number of nodes, that is a large map, is required, as for all discrete mappings. However such a computational burden can be greatly reduced by interpolating a trained map [127], or by incorporating a local linear projection on the trained low resolution map [122].

A comparison with other mapping methods, such as PCA, Sammon mapping, Isomap and Local Linear Embedding (LLE) [93] on a highly nonlinear 'S' shape manifold is also shown in Fig. 11. In this example, the resolution of the ViSOM is enhanced [122].

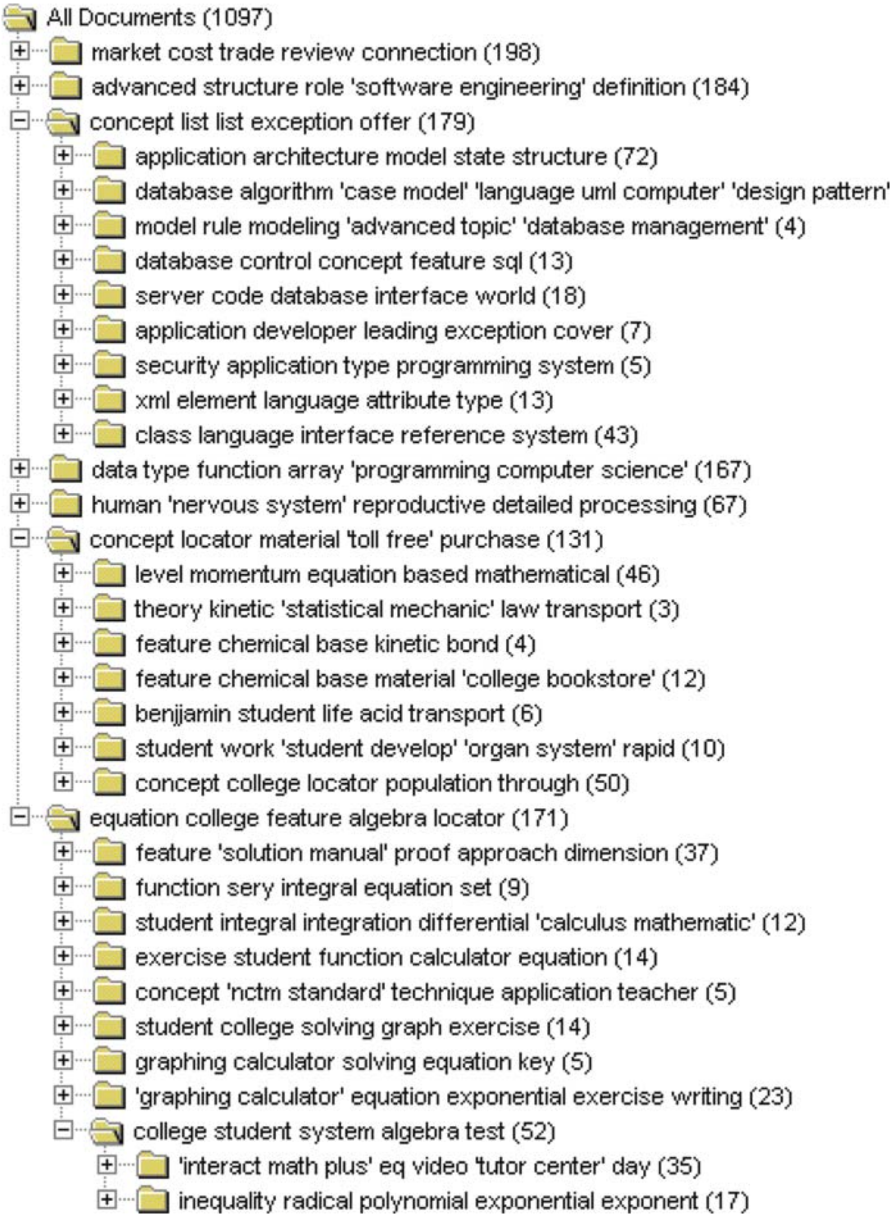
5.6 Text Mining and Information Management

With drastically increasing amounts of unstructured content available electronically within an enterprise or on the web, it is becoming inefficient if not impossible to rely on human operators to manually annotate electronic documents. (Web) content management systems have become an important area of research for many applications, such as e-libraries, enterprise portals, e-commerce, software content management, document management, and knowledge discovery. The documents, generated in an enterprise either centrally or locally by employees, are often unstructured or arranged in *ad hoc* manner (for example, emails, reports, web pages, presentations). Document management addresses many issues, such as storage, indexing, security, revision control, retrieval and organization of documents. Many existing full-text search engines return a large ranked list of documents, many of which are irrelevant. This is especially true when queries are short and very general words are used. Hence document organization has become important in information retrieval and content management.

The SOM has been applied to organize and visualize vast amounts of textual information. Typical examples include the `Welfaremap` [41] and `WEBSOM` [34]. Many SOM variants have been proposed and applied to document organization, for instance, `TreeGCS` [33] and the growing hierarchical-SOM (GH-SOM) [88]. The main advantage of SOM is the topology preservation of input space, which makes similar topics appear closely on the map. Most of these applications however are based on 2-D maps and grids, which are intuitive for the concept of a digital library. However such a presentation of information (mainly document files) is counter to all existing computer file organizers and explorers, such as `MS Windows Explorer`.

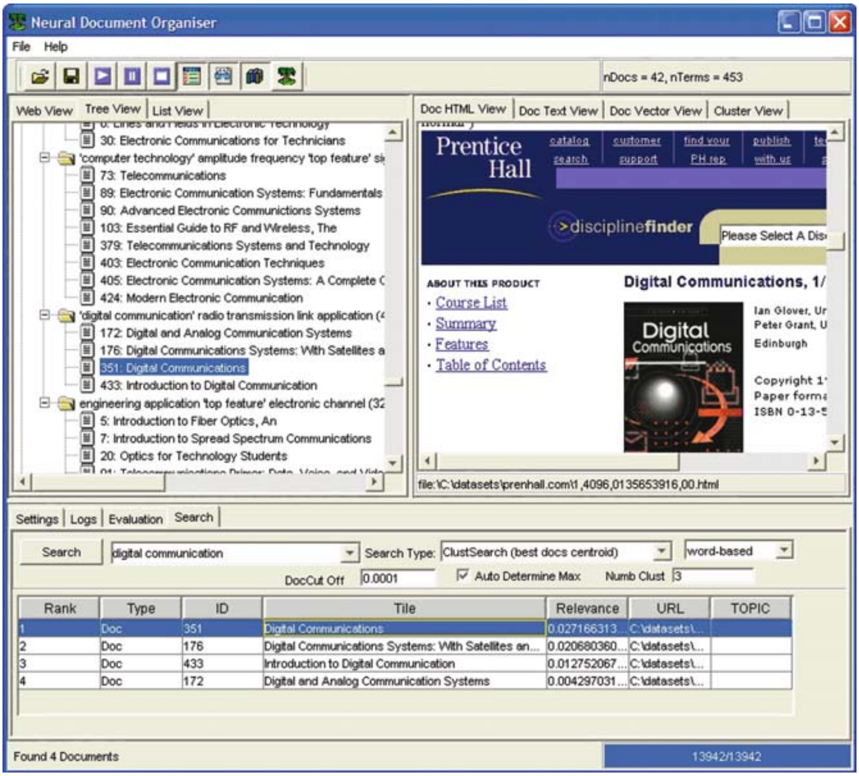
We present a new way of utilizing the SOM as a topology-preserving manifold tree-structure for content management and knowledge discovery [23]. The method can generate a taxonomy of topics from a set of unannotated, unstructured documents. It consists of a hierarchy of self-organizing growing chains, each of which can develop independently in terms of size and topics. The dynamic development process is validated continuously using a proposed entropy-based Bayesian information criterion. Each chain meeting the criterion spawns child chains, with reduced vocabularies and increased specializations. This results in a topological tree hierarchy, which can be browsed like a table of contents directory or web portal. A typical tree is shown in Fig. 12. The approach has been tested and compared with several existing methods on real world web page datasets. The results have clearly demonstrated the advantages and efficiency in content organization of the proposed method in terms of computational cost and representation. The preserved topology provides a unique, additional feature for retrieving related topics and confining the search space.

An application prototype developed based this method is shown in Fig. 13. The left panel displays the generated content tree with various levels and preserved topology on these levels. The right panel shows the details of a selected level or branch or a particular document. The method bears a similar interface to many computer file managers, especially the most popular `MS Windows Explorer` style.



- All Documents (1097)
  - market cost trade review connection (198)
  - advanced structure role 'software engineering' definition (184)
  - concept list list exception offer (179)
    - application architecture model state structure (72)
    - database algorithm 'case model' 'language uml computer' 'design pattern'
    - model rule modeling 'advanced topic' 'database management' (4)
    - database control concept feature sql (13)
    - server code database interface world (18)
    - application developer leading exception cover (7)
    - security application type programming system (5)
    - xml element language attribute type (13)
    - class language interface reference system (43)
  - data type function array 'programming computer science' (167)
  - human 'nervous system' reproductive detailed processing (67)
  - concept locator material 'toll free' purchase (131)
    - level momentum equation based mathematical (46)
    - theory kinetic 'statistical mechanic' law transport (3)
    - feature chemical base kinetic bond (4)
    - feature chemical base material 'college bookstore' (12)
    - benjjamin student life acid transport (6)
    - student work 'student develop' 'organ system' rapid (10)
    - concept college locator population through (50)
  - equation college feature algebra locator (171)
    - feature 'solution manual' proof approach dimension (37)
    - function sery integral equation set (9)
    - student integral integration differential 'calculus mathematic' (12)
    - exercise student function calculator equation (14)
    - concept 'nctm standard' technique application teacher (5)
    - student college solving graph exercise (14)
    - graphing calculator solving equation key (5)
    - 'graphing calculator' equation exponential exercise writing (23)
    - college student system algebra test (52)
      - 'interact math plus' eq video 'tutor center' day (35)
      - inequality radical polynomial exponential exponent (17)

Fig. 12. A typical result of using a topological tree structure for organizing documents



30: Electronic Communications for Technicians
'computer technology' amplitude frequency 'top feature' si
73: Telecommunications
89: Electronic Communication Systems: Fundamentals
90: Advanced Electronic Communictions Systems
103: Essential Guide to RF and Wireless, The
379: Telecommunications Systems and Technology
403: Electronic Communication Techniques
405: Electronic Communication Systems: A Complete C
424: Modern Electronic Communication
'digital communication' radio transmission link application (4
172: Digital and Analog Communication Systems
176: Digital Communications Systems: With Satellites a
351: Digital Communications
433: Introduction to Digital Communication
engineering application 'top feature' electronic channel (32
5: Introduction to Fiber Optics, An
7: Introduction to Spread Spectrum Communications
20: Optics for Technology Students

file:\C:\datasets\prenhall.com\1,4096,0135653916,00.html

Search: digital communication
Search Type: ClustSearch (best docs centroid)
word-based
DocCut Off 0.0001
☑ Auto Determine Max
Numb Clust 3

Rank	Type	ID	Tile	Relevance	URL	TOPIC
1	Doc	351	Digital Communications	0.027166313...	C:\datasets\...	
2	Doc	176	Digital Communications Systems: With Satellites an...	0.020680360...	C:\datasets\...	
3	Doc	433	Introduction to Digital Communication	0.012752067...	C:\datasets\...	
4	Doc	172	Digital and Analog Communication Systems	0.004297031...	C:\datasets\...	

Found 4 Documents
13942/13942

Fig. 13. Screen shot of a document management system developed using a topological tree structure

6 Summary and Future Directions

This Chapter provides an overview and review on the self-organizing map (SOM). First, it reviewed the biological background of SOM and showed that it is a simplified and abstract mathematical model of the retina-cortex mapping based on Hebbian learning and the lateral inhibition phenomena. Then from the mathematics of the algorithm, we discussed and explained its underlying cost function and various measures for mapping quality. Then its variant, the visualization induced SOM (ViSOM), was proposed for preserving local metrics on the map, and reviewed for use in data visualization and nonlinear manifold mapping. The relationships between SOM, ViSOM, multidimensional scaling, principal curve/surface, kernel PCA and several other nonlinear projection methods were analyzed and discussed. Both the SOM and ViSOM are multidimensional scaling methods and produce nonlinear dimension-reduction mapping or manifold of the input space. The SOM was shown to be a qualitative scaling method, while the ViSOM is a metric scaling method and approximates a discrete principal curve/surface. The SOM has also been extended to a probabilistic model and the resulting self-organizing mixture model also reveals that self-organization is an entropy-related optimization process. Furthermore, such a self-organizing model naturally approximates the kernel method. Examples and typical applications of SOM and ViSOM in the context of pattern recognition, clustering, classification, data visualization and mining, and nonlinear manifolds were presented.

Future challenges lie in several areas. First, although the SOM-related methods are finding wide application in more and more fields, to make the methods more efficient, robust and consistent is a key challenge, especially for large-scale, real-world applications. To adapt the methods for various input formats and conditions, such as temporal sequences and qualitative inputs, is also an on-going research focus. For general pattern recognition, the SOM may have more potential than implied by current practice, which often limits the SOM to a 2-D map and empirically chosen model parameters. Ways of applying and extending SOM for optimal clustering and classification also need to be investigated further. Last but not the least, to make this biologically inspired model more biologically relevant is also a key challenge. The model may have to be further extended in order to deal with complex biological signals and networks, for example in handling spikes and more importantly multiple, perhaps inhomogeneous and population spike trains. A synergy with other biologically relevant models seems necessary for modeling large-scale complex biological systems, especially the brain. Neural coding is widely studied under information theory. Probabilistic extensions of the SOM may provide useful tools in deciphering and interpreting the information content and relationships conveyed among stimuli and responses.

References

1. Allinson NM, Obermayer K, Yin H (2002) *Neural Networks* (Special Issue on New Developments in Self-Organising Maps), 15: 937–1155.
2. Allinson NM, Yin H (1999) Self-organising maps for pattern recognition. In: Oja E, Kaski S (eds.) *Kohonen Maps*, Elsevier, Amsterdam, The Netherlands: 111–120.
3. Ameri S-I (1980) Topographic organisation of nerve fields. *Bulletin Mathematical Biology*, 42: 339–364.
4. Andras P (2002) Kernel-Kohonen networks. *Intl. J. Neural Systems*, 12: 117–135.
5. Banfield JD, Raftery AE (1992) Ice floe identification in satellite images using mathematical morphology and clustering about principal curves. *J. American Statistical Association*, 87: 7–16.
6. Barreto GA, Araujo AFR, Ducker C, Ritter H (2002) A distributed robotic control system based on a temporal self-organizing neural network. *IEEE Trans. Systems, Man and Cybernetics – C*, 32: 347–357.

7. Bauer H-U, Pawelzik KR (1992) Quantifying the neighborhood preservation of self-organizing feature maps. *IEEE Trans. Neural Networks*, 3: 570–579.
8. Bishop CM, Svensén M, Williams CKI (1998) GTM: the generative topographic mapping. *Neural Computation*, 10: 215–235.
9. Bruce V, Green PR (1990) *Visual Perception: Physiology, Psychology and Ecology (2nd ed.)*, Lawerence Erlbraum Associates, East Essex, UK.
10. Chang K-Y, Ghosh J (2001) A unified model for probabilistic principal surfaces. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 23: 22–41.
11. Chappell GJ, Taylor JG (1993) The temporal Kohonen map. *Neural Networks*, 6: 441–445.
12. Cottrell M, Fort JC (1986) A stochastic model of retinotopy: a self-organising process. *Biological Cybernetics*, 53: 405–411.
13. Cottrell M, Verleysen M (2006) *Neural Networks* (Special Issue on Advances in Self-Organizing Maps), 19: 721–976.
14. Cox TF, Cox MAA (1994) *Multidimensional Scaling*, Chapman & Hall, London, UK.
15. de Bolt E, Cottrell M, Verleysen M (2002) Statistical tools to assess the reliability of self-organising maps. *Neural Networks*, 15: 967–978.
16. De Ridder D, Duin RPW (1997) Sammon mapping using neural networks: a comparison. *Pattern Recognition Letters*, 18: 1307–1316.
17. Dersch DR, Tavan P (1995) Asymptotic level density in topological feature maps. *IEEE Trans. Neural Networks*, 6: 230–236.
18. Durbin R, Mitchison G (1990) A dimension reduction framework for understanding cortical maps. *Nature*, 343: 644–647.
19. Erwin E, Obermayer K, Schulten K (1992) Self-organising maps: ordering, convergence properties and energy functions. *Biological Cybernetics*, 67: 47–55.
20. Erwin E, Obermayer K, Schulten K (1992) Self-organising maps: stationary states, metastability and convergence rate. *Biological Cybernetics*, 67: 35–45.
21. Estévez PA, Figueroa CJ (2006) Online data visualization using the neural gas network. *Neural Networks*, 19: 923–934.
22. Ferguson KL, Allinson NM (2004) Efficient video compression codebooks using SOM-based vector quantisation. *Proc. IEE – Vision, Image and Signal Processing*, 151: 102–108.
23. Freeman R, Yin H (2004) Adaptive topological tree structure (ATTS) for document organisation and visualisation. *Neural Networks*, 17: 1255–1271.
24. Gaze RM (1970) *The Information of Nerve Connections*, Academic Press, London, UK.
25. Goodhill GJ, Sejnowski T (1997) A unifying objective function for topographic mappings. *Neural Computation*, 9: 1291–1303.
26. Graepel T, Burger M, Obermayer K (1997) Phase transitions in stochastic self-organizing maps. *Physics Reviews E*, 56: 3876–3890.
27. Haritopoulos M, Yin H, Allinson NM (2002) Image denoising using self-organising map-based nonlinear independent component analysis. *Neural Networks*, 15: 1085–1098.
28. Hastie T, Stuetzle W (1989) Principal curves. *J. American Statistical Association*, 84: 502–516.
29. Haykin S (1998) *Neural Networks: A Comprehensive Foundation (2nd ed.)*, Prentice Hall, Englewood Cliffs, NJ.
30. Hebb D (1949) *Organisation of behavior*, Wiley, New York, NY.

31. Herrmann M, Yang H (1996) Perspectives and limitations of self-organising maps in blind separation of source signals. In: Amari S-I, Xu L, Chan L-W, KIng I, Leung K-S (eds.) *Proc. Intl. Conf. Neural Information Processing (ICONIP'96)*, 24–27 September, Hong Kong. Springer-Verlag, Singapore: 1211–1216.
32. Heskes T (1999) Energy functions for self-organizing maps, In: Oja E, Kaski S (eds.) *Kohonen Maps*, Elsevier, Amsterdam: 303–315.
33. Hodge VJ, Austin J (2001) Hierarchical growing cell structures: TreeGCS. *IEEE Trans. Knowledge and Data Engineering*, 13: 207–218.
34. Honkela T, Kaski S, Lagus K, Kohonen T (1997) WEBSOM-self-organizing maps of document collections. In: *Proc. Workshop on Self-Organizing Maps (WSOM'97)*, 4–6 June, Helsinki, Finland. Helsinkis University of Technology: 310–315.
35. Hsu C-C (2006) Generalising self-organising map for categorical data. *IEEE Trans. Neural Networks*, 17: 294–304.
36. Hyvärinen A, Karhunen J, Oja E (2001) *Independent Component Analysis.* Wiley, New York, NY.
37. Hyvärinen A, Pajunen P (1999) Nonlinear independent component analysis: Existence and uniqueness results. *Neural Networks*, 12: 429–439.
38. Ishikawa M, Miikkulainen R, Ritter H (2004) *Neural Networks* (Special Issue on New Developments in Self-Organizing Systems), 17: 1037–1389.
39. Karhunen J, Joutsensalo J (1995) Generalisation of principal component analysis, optimisation problems, and neural networks. *Neural Networks*, 8: 549–562.
40. Kaski S, Kangas J, Kohonen T (1998) Bibliography of self-organizing map (SOM) papers: 1981–1997. *Neural Computing Surveys*, 1: 1–176.
41. Kaski S, Kohonen T (1996) Exploratory data analysis by the self-organizing map: Structures of welfare and poverty in the world. In: Refenes A-PN, Abu-Mostafa Y, Moody J, Weigend A (eds.) *Neural Networks in Financial Engineering*, World Scientific, Singapore: 498–507.
42. Kegl B, Krzyzak A, Linder T, Zeger K (1998) A polygonal line algorithm for constructing principal curves. *Neural Information Processing Systems (NIPS'98)*, 11: 501–507.
43. Kohonen T (1972) Correlation matrix memory. *IEEE Trans. Computers*, 21: 353–359.
44. Kohonen T (1973) A new model for randomly organised associative memory. *Intl. J. Neuroscience*, 5: 27–29.
45. Kohonen T (1974) An adaptive associative memory principle. *IEEE Trans. Computers*, 23: 444–445.
46. Kohonen T (1982) Self-organised formation of topologically correct feature map. *Biological Cybernetics*, 43: 56–69.
47. Kohonen T (1984) *Self-organization and Associative Memory*, Springer-Verlag, Berlin.
48. Kohonen T (1986) Representation of sensory information in self-organising feature maps, and relation of these maps to distributed memory networks. *Proc. SPIE*, 634: 248–259.
49. Kohonen T (1987) Adaptive, associative, and self-organizing functions in neural computing, *Applied Optics*, 26: 4910–4918.

50. Kohonen T (1991) Self-organizing maps: optimization approaches. In: Kohonen T, Makisara K, Simula O, Kangas J (eds.) *Artificial Neural Networks 2*, North-Holland, Amsterdam, The Netherlands: 981–990.
51. Kohonen T (1995) The adaptive-subspace SOM (ASSOM) and its use for the implementation of invariant feature detection. In: Fogelman-Soulié, Gallinari P (eds.) *Proc. Intl. Conf. Artificial Neural Systems (ICANN'95)*, 9–13 October, Paris, France. EC2 Nanterre, France, 1: 3–10.
52. Kohonen T (1996) Emergence of invariant-feature detectors in the adaptive-subspace self-organizing map. *Biological Cybernetics*, 75: 281–291.
53. Kohonen T (1997) *Self-Organising Maps (2nd ed.)*. Springer-Verlag, Berlin.
54. Kohonen T (1999) Comparison of SOM point densities based on different criteria. *Neural Computation*, 11: 2081–2095.
55. Kohonen T, Somervuo P (2002) How to make a large self-organising maps for nonvectorial data. *Neural Networks*, 15: 945–952.
56. Kramer MA (1991) Nonlinear principal component analysis using autoassociative neural networks. *American Institute Chemical Engineers J.*, 37: 233–243.
57. Laaksonen J, Koskela M, Laakso S, Oja E (2000) PicSOM - content-based image retrieval with self-organizing maps. *Pattern Recognition Letters*, 21: 1199–1207.
58. Lampinen J, Oja E (1992) Clustering properties of hierarchical self-organizing maps. *J. Mathematical Imaging and Vision*, 2: 261–272.
59. Lau KW, Yin H, Hubbard S (2006) Kernel self-organising maps for classification. *Neurocomputing*, 69: 2033–2040.
60. LeBlanc M, Tibshirani RJ (1994) Adaptive principal surfaces. *J. American Statistical Association*, 89: 53–64.
61. Lee RCT, Slagle JR, Blum H (1977) A triangulation method for the sequential mapping of points from n-space to two-space. *IEEE Trans. Computers*, 27: 288–292.
62. Lin S, Si J (1998) Weight-value convergence of the SOM algorithm for discrete input. *Neural Computation*, 10: 807–814.
63. Linde Y, Buzo A, Gray RM (1980) An algorithm for vector quantizer design. *IEEE Trans. Communications*, 28: 84–95.
64. Lo ZP, Bavarian B (1991) On the rate of convergence in topology preserving neural networks. *Biological Cybernetics*, 65: 55–63.
65. Lowe D, Tipping ME (1996) Feed-forward neural networks and topographic mappings for exploratory data analysis. *Neural Computing and Applications*, 4: 83–95.
66. Luttrell SP (1990) Derivation of a class of training algorithms. *IEEE Trans. Neural Networks*, 1: 229–232.
67. Luttrell SP (1991) Code vector density in topographic mappings: Scalar case, *IEEE Trans. Neural Networks*, 2: 427–436.
68. Luttrell SP (1994) A Bayesian analysis of self-organising maps. *Neural Computation*, 6: 767–794.
69. MacDonald D, Fyfe C (2000) The kernel self organising map. In: *Proc. 4th Intl. Conf. Knowledge-based Intelligence Engineering Systems and Applied Technologies*, 30 August – 1 September, Brighton, UK, IEEE Press, Piscataway, NJ: 317–320.
70. Malthouse EC (1998) Limitations of nonlinear PCA as performed with generic neural networks. *IEEE Trans. Neural Networks*, 9: 165–173.

71. Mao J, Jain AK (1995) Artificial Neural Networks for Feature Extraction and Multivariate Data Projection. *IEEE Trans. Neural Networks*, 6: 296–317.
72. Marr D (1969) A theory of cerebellar cortex. *J. Physiology*, 202: 437–70.
73. Marsland S, Shapiron J, Nehmzow U (2002) A self-organising network that grows when required. *Neural Networks*, 15: 1041–1058.
74. Martinetz TM, Schulten KJ (1991) A "neuralgas" network learns topologies. In: Kohonen T, Mäkisara K, Simula O, Kangas J (eds.) *Artificial Neural Networks*, NorthHolland, Amsterdam, The Netherlands: 397–402.
75. Martinetz TM, Schulten KJ (1994) Topology representing networks. *Neural Networks*, 7: 507–522.
76. Miikkulainen R (1990) Script recognition with hierarchical feature maps. *Connection Science*, 2: 83–101.
77. Miikkulainen R (1997) Dyslexic and category-specific aphasic impairments in a self-organizing feature map model of the lexicon. *Brain and Language*, 59: 334–366.
78. Mitchison G (1995) A type of duality between self-organising maps and minimal wiring. *Neural Computation*, 7: 25–35.
79. Möller-Levet CS, Yin H (2005) Modeling and analysis of gene expression time-series based on co-expression. *Intl. J. Neural Systems*, (Special Issue on Bioinformatics), 15: 311–322.
80. Möller-Levet CS, Yin H (2005) Circular SOM for temporal characterisation of modelled gene expressions. In: Gallagher M, Hogan J, Maire F (eds.) *Proc. Intl. Conf. Intelligent Engineering Data Engineering and Automated Learning Conf. (IDEAL'05)*, 6–8 July, Brisbane, Australia. Lecture Notes in Computer Science 3578, Springer-Verlag, Berlin: 319–326.
81. Oja E (1989) Neural networks, principal components, and subspaces. *Intl. J. Neural Systems*, 1: 61–68.
82. Oja E (1995) PCA, ICA, and nonlinear Hebbian learning. In: Fogelman-Soulié F, Gallinari P (eds.) *Proc. Intl. Conf. Artificial Neural Networks (ICANN'95)*, 9–13 October, Paris, France. EC2, Nanterre, France: 89–94.
83. Oja M, Kaski S, Kohonen T (2003) Bibliography of self-organizing map (SOM) papers: 1998–2001 addendum. *Neural Computing Surveys*, 3: 1–156.
84. Pajunen P, Hyvärinen A, Karhunen J (1996) Nonlinear blind source separation by self-organising maps. In: Amari S-I, Xu L, Chan L-W, King I, Leung K-S (eds.) *Proc. Intl. Conf. Neural Information Processing (ICONIP'96)*, 24–27 September, Hong Kong. Springer-Verlag, Singapore: 1207–1210.
85. Pan ZS, Chen SC, Zhang DQ (2004) A kernel-base SOM classifier in input space. *Acta Electronica Sinica*, 32: 227–231 (in Chinese).
86. Pearson D, Hanna E, Martinez K (1990) Computer-generated cartoons. In: . Barlow H, Blakemore C, Weston-Smith M (eds.) *Images and Understandings*, Cambridge University Press, Cambridge, UK.
87. Ratcliff F (1965) *Mach Bands: Quantitative Studies on Neural Networks in the Retina.* Holden-Day, Inc., San Francisco, CA.
88. Rauber A, Merkl D, Dittenbach M (2002) The growing hierarchical self-organizing map: exploratory analysis of high-dimensional data. *IEEE Trans. Neural Networks*, 13: 1331–1341.
89. Ritter H (1991) Asymptotical level density for class of vector quantisation processes. *IEEE Trans. Neural Networks*, 2: 173–175.

90. Ritter H, Schulten K (1988) Convergence properties of Kohonen's topology conserving maps: fluctuations, stability, and dimension selection. *Biological Cybernetics*, 60: 59–71.
91. Ritter H, Martinetz T, Schulten K (1992) *Neural Computation and Self-organising Maps: An Introduction*. Addison-Wesley, Reading, MA.
92. Robbins H, Monro S (1952) A stochastic approximation method. *Annals Mathematical Statistics*, 22: 400–407.
93. Roweis ST, Saul LK (2000) Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290: 2323–2326.
94. Sakrison DJ (1966) Stochastic approximation: A recursive method for solving regression problems. In: Balakrishnan V (ed.) *Advances in Communication Systems: Theory and Applications 2*, Academic Press, New York, NY: 51–100.
95. Sammon JW (1969) A nonlinear mapping for data structure analysis. *IEEE Trans. Computers*, 18: 401–409.
96. Sanger TD (1991) Optimal unsupervised learning in a single-layer linear feedforward network. *Neural Networks*, 2: 459–473.
97. Schölkopf B, Smola A, Müller KR (1998) Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10: 1299–131.
98. Seiffert U, Michaelis B (2001) Multi-dimensional self-organizing maps on massively parallel hardware. In: Allinson N, Yin H, Allinson L, Slack J (eds.) *Advances in Self-Organising Maps*, Springer-Verlag, London: 160–166.
99. Shepherd GM (1988) *Neurobiology (2nd ed.)*, Oxford University Press, Oxford, UK.
100. Sum J, Leung C-S, Chan L-W, Xu L (1997) Yet another algorithm which can generate topography map. *IEEE Trans. Neural Networks*, 8: 1204–1207.
101. Sutton RS, Barto AG, Williams RJ (1991) Reinforcement learning is direct adaptive optimal control. In: *Proc. American Control Conf.*, 26–28 June, Boston, MA. IEEE Press, Piscataway, NJ: 2143–2146.
102. Tenenbaum JB, de Silva V, Langford JC (2000) A global geometric framework for nonlinear dimensionality reduction. *Science*, 290: 2319–2323.
103. Tibshirani R (1992) Principal curves revisited. *Statistics and Computation*, 2: 183–190.
104. Törönen P, Kolehmainen K, Wong G, Castrén E (1999) Analysis of gene expression data using self-organising maps. *Federation European Biochemical Socities Letters*, 451: 142–146.
105. Ultsch A (1993) Self-organising neural networks for visualisation and classification. In: Opitz O, Lausen B, Klar R (eds.) *Information and Classification*, Springer-Verlag, Berlin: 864–867.
106. Van Hulle MM (1998) Kernel-based equiprobabilitic topographic map formation. *Neural Computation*, 10: 1847–1871.
107. Van Hulle MM (2002) Kernel-based equiprobabilitic topographic map formation achieved with an information-theoretic approach. *Neural Networks*, 15: 1029–1040.
108. Varsta M, del Ruiz Millän J, Heikkonen J (1997) A recurrent self-organizing map for temporal sequence processing. *Proc. ICANN'97*, Lausanne, Switzerland. Springer-Verlag, Berlin: 197–202.
109. Villmann T, Der R, Herrmann M, Martinetz TM (1997) Topology preservation in self-organizing feature maps: exactdefinition and measurement. *IEEE Trans. Neural Networks*, 8: 256–266.

110. Voegtlin T (2002) Recursive self-organizing maps. *Neural Networks*, 15: 979–992.
111. von der Malsburg C, Willshaw DJ (1973) Self-organization of orientation sensitive cells in the striate cortex. *Kybernetik*, 4: 85–100.
112. Walter J, Ritter H (1996) Rapid learning with parametrized self-organizing maps. *Neurocomputing*, 12: 131–153.
113. Wang Y, Yin H, Zhou L-Z, Liu Z-Q (2006) Real-time synthesis of 3D animations by learning parametric gaussians using self-organizing mixture networks. In: King I, Wang J, Chan L, Wang D (eds.) *Proc. Intl. Conf. Neural Information Processing (ICONIP'06)*, 2–6 October, Hong Kong. Lecture Notes in Computer Science 4233, Springer-Verlag, Berlin, II: 671–678.
114. Willshaw DJ, Buneman OP, Longnet-Higgins HC (1969) Non-holographic associative memory. *Nature*, 222: 960–962.
115. Willshaw DJ, von der Malsburg C (1976) How patterned neural connections can be set up by self-organization. *Proc. Royal Society of London – Series B*, 194: 431–445.
116. Wu S, Chow TWS (2005) PRSOM: A new visualization method by hybridizing multidimensional scaling and self-organizing map. *IEEE Trans. Neural Networks*, 16: 1362–1380.
117. Yin H (1996) Self-Organising Maps: Statistical Analysis, Treatment and Applications, *PhD Thesis*, Department of Electronics, University of York, UK.
118. Yin H (2001) Visualisation induced SOM (ViSOM). In: Allinson N, Yin H, Allinson L, Slack J (eds.) *Advances in Self-Organising Maps*, Springer-Verlag, London, UK: 81–88.
119. Yin H (2002) ViSOM-A novel method for multivariate data projection and structure visualisation. *IEEE Trans. Neural Networks*, 13: 237–243.
120. Yin H (2002) Data visualisation and manifold mapping using the ViSOM. *Neural Networks*, 15: 1005–1016.
121. Yin H (2003) Nonlinear multidimensional data projection and visualisation. In: Liu J, Cheung Y, Yin H (eds.) *Proc. Intl. Conf. Intelligent Data Engineering and Automated Learning (IDEAL'03)*, 21–23 March, Hong Kong. Lecture Notes in Computer Science 2690, Springer-Verlag, Berlin: 377–388.
122. Yin H (2003) Resolution enhancement for the ViSOM. In: *Proc. Workshop on Self-Organizing Maps*, 11–14 September, Kitakyushu, Japan. Kyushu Institute of Technology: 208–212.
123. Yin H (2006). On the equivalence between kernel self-organising maps and self-organising mixture density networks. *Neural Networks*, 19: 780–784.
124. Yin H (2007) Connection between self-organising maps and metric multidimensional scaling. In: *Proc. Intl. Joint Conf. Neural Networks (IJCNN2007)*, 12–17 August, Orlando, FL. IEEE Press, Piscataway, NJ: (in press).
125. Yin H, Allinson NM (1994) Unsupervised segmentation of textured images using a hierarchical neural structure. *Electronics Letters*, 30: 1842–1843.
126. Yin H, Allinson NM (1995) On the distribution and convergence of the feature space in self-organising maps. *Neural Computation*, 7: 1178–1187.
127. Yin H, Allinson NM (1999) Interpolating self-organising map (iSOM). *Electronics Letters*, 35: 1649–1650.
128. Yin H, Allinson NM (2001) Self-organising mixture networks for probability density estimation. *IEEE Trans. Neural Networks*, 12: 405–411.
129. Yin H, Allinson NM (2001) Bayesian self-organising map for Gaussian mixtures. *Proc. IEE – Vision, Image and Signal Processing*, 148: 234–240.

Resources

1 Key Books

Kohonen T (1995, 1997, 2001) *Self-Organizing Maps (1st, 2nd and 3rd editions).* Springer-Verlag, Berlin.

Haykin S (1994) *Neural Networks: A Comprehensive Foundation.* Macmillian, New York, NY.

Haykin S (1999) *Neural Networks: A Comprehensive Foundation (2nd ed.).* Prentice Hall, Englewood Cliffs, NJ.

Oja E, Kaski S (eds.) (1999) *Kohonen Maps.* Elsevier, Amsterdam, The Netherlands.

Allinson N, Yin H, Allinson L, Slack J (eds.) (2001) *Advances in Self-Organising Maps.* Springer-Verlag, London, UK.

Ritter H, Martinetz T, Schulten K (1992) *Neural Computation and Self-organising Maps: An Introduction.* Addison Wesley, Reading, MA.

Van Hulle MM (2000) *Faithful Representations and Topographic Maps: From Distortion to Information Based Self-Organization.* Wiley, New York, NY.

2 Key Survey/Review Articles

Allinson NM, Obermayer K, Yin H (eds.) (2002) Special Issue on New Developments in Self-Organising Maps, *Neural Networks,* 15(8–9): 937–1155.

Ishikawa M, Miikkulainen R, Ritter H (eds.) (2004) Special Issue on New Developments in Self-Organizing Systems, *Neural Networks,* 17(8-9): 1037–1389.

Cottrell M, Verleysen M (eds.) (2006) Special Issue on Advances in Self-Organizing Maps, *Neural Networks,* 19(5–6): 721–976.

3 Key International Conferences/Workshops

WSOM – Intl. Workshop on Self-Organizing Maps *(biennial, since 1997).*

IJCNN – Intl. Joint Conference on Neural Networks *(annual, since 1987).*

ICANN – Intl. Conference on Artificial Neural Networks *(annual, since 1991).*

ESANN – European Symp. Artificial Neural Networks *(annual, since 1993).*

ICONIP – Intl. Conf. Neural Information Processing *(annual, since 1994).*

IDEAL – International Conference on Intelligent Data Engineering and Automated Learning *(annual, since 1998).*

ISNN – Intl. Symposium on Artificial Neural Networks *(annual, since 2004).*

4 (Open Source) Software

SOM Toolbox
http://www.cis.hut.fi/projects/somtoolbox/