Biol. Cybern. 67, 47–55 (1992)

Biological Cybernetics

Self-organizing maps: ordering, convergence properties and energy functions

E. Erwin, K. Obermayer, and K. Schulten

Beckman Institute and Department of Physics, University of Illinois at Urbana – Champaign, 405 North Mathews Avenue, Urbana, IL 61801, USA

Received July 22, 1991/Accepted in revised form December 18, 1991

Abstract. We investigate the convergence properties of the self-organizing feature map algorithm for a simple, but very instructive case: the formation of a topographic representation of the unit interval [0,1] by a linear chain of neurons. We extend the proofs of convergence of Kohonen and of Cottrell and Fort to hold in any case where the neighborhood function, which is used to scale the change in the weight values at each neuron, is a monotonically decreasing function of distance from the winner neuron. We prove that the learning dynamics cannot be described by a gradient descent on a *single* energy function, but may be described using a set of potential functions, one for each neuron, which are independently minimized following a stochastic gradient descent. We derive the correct potential functions for the one- and multi-dimensional case, and show that the energy functions given by Tolat (1990) are an approximation which is no longer valid in the case of highly disordered maps or steep neighborhood functions.

1 Introduction

The self-organizing feature map (SOFM) algorithm (Kohonen 1982a, b) is a biologically-inspired method for constructing a structured representation of data from an often high-dimensional *input space*. As in vector quantization, the data is represented by prototypes, called *weight vectors*. Unlike vector quantization, however, these weight vectors are associated with selected elements, the *neurons*, of an image space, where metric relationships are defined between the elements. For any given data-set, the SOFM algorithm selects weight vectors and assigns them to neurons in the network. The weight vectors as a function of neuron coordinates are called the *feature map*.

Feature maps generated by the SOFM algorithm are characterized by the fact that weight vectors which are neighbors in the input space are mapped onto neighboring neurons. If the dimensionalities of the input space[1] and the network differ, it is impossible to preserve all similarity relationships among weight vectors in the input space; only the most "important" similarity relationships are preserved and mapped onto neighborhood relationships on the network of neurons, while the less "important" similarity relationships are not retained in the mapping. If the input space and network are of the same dimensionality, the SOFM algorithm can preserve all the similarity relationships and generates a distorted, but topographic map, of the input space, where more "important" regions of the input space are represented with higher resolution.

The low-dimensional, ordered representation of data generated by the SOFM algorithm has proven useful for a variety of technical applications in the areas of pattern classification and function approximation (Favata and Walker 1991; Kohonen 1989; Ritter et al. 1989), as well as knowledge representation (Ritter and Kohonen 1989; Scholtes 1991), and the algorithm has been successfully applied as a model for the development of structural representations in biological neural systems, the so-called brain maps (Obermayer et al. 1990a, b, 1991). However, a general theory of the algorithm has not yet been achieved. It is not clear under what conditions the algorithm may be guaranteed to converge or whether the algorithm works by performing a stochastic gradient descent on some potential function, and problems of important practical interest, like the number and type of the algorithm's stationary states, convergence speed as a function of the algorithm's parameters and the avoidance of sub-optimal representations, are not solved. The intent of this, and a companion paper (Erwin et al. 1992) is to answer some of these questions for a simple, but very instructive case: the formation of a topographic representation of the unit interval by a linear chain of neurons.

Correspondence to: E. Erwin

[1] Actually the dimensionality of the given data manifold, which is embedded in this input space

The formation of topologically ordered feature maps occurs easily in a wide range of situations, where the dimensionalities of the input and image spaces are the same or different and where the interaction function, or neighborhood function, takes a variety of forms. So far no theory exists which can demonstrate how the algorithm actually works to generate topologically correct feature maps in such a variety of conditions. Even for the simplest case, the formation of a topological map of a one-dimensional space by a linear array of neurons, a rigorous proof that an ordered mapping will be formed has only been provided for a very restricted case, where the neighborhood function is a one-unit-wide step-function, and for higher-dimensional cases no proof of ordering has yet been presented.

This paper is organized into five parts. After a short introduction to the algorithm in Sect. 2, we present a proof of ordering for the one-dimensional case which holds for any neighborhood function which is monotonically decreasing with distance. Unfortunately, our proof gives little insight into the way the algorithm itself actually forms the ordered representation.

In Sects. 3 and 4 we investigate the hypothesis that the SOFM algorithm minimizes a potential function by a downhill search, or gradient descent procedure. It turns out that even the one-dimensional SOFM algorithm cannot be derived as a stochastic gradient descent on *any* energy function. A set of "energy" functions, one for each weight vector, seems to be the best description of the dynamics of the algorithm. This approach was first suggested by Tolat (1990), but the expressions given were incorrect. In Sect. 4 we present the correct energy expressions for the SOFM algorithm in one dimension and generalize this approach to arbitrary dimensions. Our findings are summarized in Sect. 5.

In a second paper (Erwin et al. 1992) we consider some consequences of the results of this paper. We provide the conditions under which the energy equations admit non-ordered stationary states, and we characterize these states and study their influence on the rate of convergence of the algorithm towards the ordered map.

2 The SOFM algorithm

The SOFM algorithm employs a set of neurons, which, in the general case, are arranged in a network of a certain dimensionality. The location of any neuron in the network is specified by a position vector $\mathbf{s}$. The weight vector associated with neuron $\mathbf{s}$ in the *feature map* is denoted $\mathbf{w}_\mathbf{s}$.

Feature map formation follows an iterative procedure. Initially the weight vectors are randomly selected, or they are chosen to take advantage of assumed or known properties of the data manifold in the input space. Then at each time step t, a *pattern* $\mathbf{v}$, an element of the data manifold in input space, is chosen at random. The neuron $\mathbf{r}$ whose weight value $\mathbf{w}_\mathbf{r}$ is metrically closest to the pattern $\mathbf{v}$,

$$\|\mathbf{w}_\mathbf{r} - \mathbf{v}\| = \min_\mathbf{s} \|\mathbf{w}_\mathbf{s} - \mathbf{v}\|, \tag{1}$$

is selected. The weight values of all neurons are then changed according to the feature map *update rule* (Kohonen 1982a, b)

$$\mathbf{w}_\mathbf{s}(t+1) - \mathbf{w}_\mathbf{s}(t) + \epsilon h(\mathbf{r}, \mathbf{s})(\mathbf{v}(t) - \mathbf{w}_\mathbf{s}(t)), \tag{2}$$

where ϵ the *learning step width*, is some small constant ($0 < \epsilon \ll 1$.) The function $h(\mathbf{r}, \mathbf{s})$ is called the *neighborhood function*. For most applications it has a maximum value of one for $\mathbf{s} \equiv \mathbf{r}$ and decreases with increasing distance between $\mathbf{s}$ and $\mathbf{r}$.

Throughout most of this paper we will consider the mapping of the unit interval [0,1] onto a one-dimensional network of N neurons. The patterns v and weight vectors, w_s will simply be scalars, and the indices s of the neurons will be integers between 1 and N. For convenience we define a *state* of the network as a particular set of weight values $\{w_s \mid s = 1,2,\ldots,N;\ w_s \in [0,1]\}$, and a *configuration* as the set of states which are characterized by the same order relations among the (scalar) weights. The update rule (2) reduces to

$$w_s(t+1) = w_s(t) + \epsilon h(r, s)[v(t) - w_s(t)] \tag{3}$$

in the one-dimensional case.

3 Proof of convergence to stable, ordered configurations

For the one-dimensional SOFM algorithm, we define an *ordered* configuration as a map of the input space [0,1] which preserves the distance relationships between input patterns such that

$$\begin{aligned} |r - s| < |r - q| &\Leftrightarrow |w_r - w_s| \\ &< |w_r - w_q|, \quad \forall r, s, q \in \{1,2,\ldots,N\}. \end{aligned} \tag{4}$$

There exist two ordered configurations which have weights arranged in either ascending or descending order.

The update rule (3) may be arranged to give

$$[w_s(t+1) - v(t)] = [1 - \epsilon h(r, s)][w_s(t) - v(t)]. \tag{5}$$

If (4) is fulfilled, if $0 < \epsilon, h(r, s) < 1$, and if the neighborhood function is monotonously decreasing with $|r - s|$, then the factor $[1 - \epsilon h(r, s)]$ is positive, smaller than one, and decreases monotonously with $[w_s - v]$. This application of (5) cannot change the sequence of weights in an ordered configuration, and ordered configurations are absorbing configurations of the mapping algorithm (see Kohonen 1988). But does every initial state eventually end up in one of the two absorbing configurations?

Cortell and Fort (1987) have shown that for a step neighborhood function

$$h(r, s) \equiv H(|r - s|) = \begin{cases} 1, & \text{if } |r - s| \leqslant 1, \\ 0, & \text{otherwise}. \end{cases} \tag{6}$$

the "number of inversions", $N(t)$, defined by

$$N(t) = \mathrm{Card}_s \{s \in \{2, \ldots, N-1\} \,|\, [w_{s+1}(t) - w_s(t)] \times [w_s(t) - w_{s-1}(t)] < 0\}, \quad (7)$$

either decreases or remains constant at each application of the update rule[2]. They used this fact as a motivation to construct a finite sequence of patterns which leads from an arbitrary initial configuration to the final ordered configuration.

For other decreasing neighborhood functions, including the frequently used Gaussian function, however, the value of $N(t)$ *may increase* during some iterations, although the probability for $N(t)$ to decrease or remain the same is usually greater than the probability for it to increase. This point has been overlooked by several authors (Geszti 1990; Hertz et al. 1991) who have interpreted the proof of Cottrell and Fort to apply to any monotonically decreasing neighborhood function. Unfortunately, their proof, although unrelated to the behavior of $N(t)$, cannot be easily extended to include all monotonically decreasing neighborhood functions.

Tolat (1990) introduced a set of energy functions, one for each neuron, and interpreted the dynamics of the one-dimensional SOFM-algorithm as a stochastic gradient descent minimizing these individual energies. He shows that ordered configurations of the weights have lower energies than all other configurations, which proves that the algorithm will necessarily converge to an ordered configuration provided that no metastable states exist. Tolat's proof of ordering is based on energy functions which are only approximately correct (see below). But even using the correct energy functions, his proof would be restricted to the subset of neighborhood functions which do not give rise to metastable states.

For a proof of ordering for arbitrary positive-valued, normalized and decreasing neighborhood functions, we take an approach similar to that outlined by Kohonen (1988) and by Cottrell and Fort (1987). The result of the proof given in Appendix A may be stated formally as

Theorem 1. *Given any set of weights* $\{w_s(t) | s = 1,2, \ldots, N\}$ *at* $t = 0$, *and a neighborhood function* $h(r, s) \equiv (|r - s|)$ *such that*

$$1 \geqslant H(0) \geqslant H(1) > H(2) > \cdots > H(N-2) > H(N-1) \geqslant 0, \quad (8)$$

there exists a sequence of patterns $\{v(t) | t = 1,2, \ldots, T\}$, $T < \infty$, *such that application of the update rule* (3) *with this sequence of values of* $v(t)$ *will result in a set of weight values* $\{w_s(t)\}$ *which fulfills condition* (4) *for* $t \geqslant T$.

From this it follows from the general properties of a Markov process (Van Kampen 1981) and with the $v(t)$ being randomly selected from a continuous distribution, the algorithm will generate an ordered configuration in a finite time with probability one. Unfortunately, the proof of convergence of the algorithm given above for the one-dimensional case cannot easily be extended to higher-dimensional cases.

Lo and Bavarian (1991) have recently attempted to prove convergence for the self-organizing feature map algorithm of *arbitrary* dimensionality. They state that given a monotonically decreasing neighborhood function, the algorithm will converge towards a "topologically ordered" configuration, defined by $\|\mathbf{v} - \mathbf{w}_a\| < \|\mathbf{v} - \mathbf{w}_b\|$ if and only if $\|\mathbf{r} - \mathbf{a}\| < \|\mathbf{r} - \mathbf{b}\|$ where $\mathbf{r}$ is the location of the winner neuron, and $\mathbf{a}$ and $\mathbf{b}$ are the locations of any other two neurons. However, the "topologically ordered" configuration of Lo and Bavarian is not absorbing, as demonstrated by the following counterexample. Consider a "topologically ordered" network given by four neurons, (11, 12, 21, 22), connected in a square lattice whose weight vectors, $\mathbf{w}_{11}$, $\mathbf{w}_{12}$, $\mathbf{w}_{21}$, and $\mathbf{w}_{22}$, project into a square pattern in the input space. If patterns, $\mathbf{v}$, are repeatedly chosen near the initial midpoint of a line segment connecting weights $\mathbf{w}_{11}$ and $\mathbf{w}_{22}$ at diagonally opposed corners of this square, but always slightly closer to $\mathbf{w}_{11}$, then after a sufficient number of iterations of the algorithm the other two weights, $\mathbf{w}_{12}$ and $\mathbf{w}_{21}$, will be closer to each other than they are to $\mathbf{w}_{22}$. Hence "topological order" is not maintained.

Lo and Bavarian also attempted to prove that if the neighborhood function is monotonically decreasing and the weight vectors are initially chosen to all be equal, then the update rule will establish the topological order in one iteration. However, if all weight vectors are initially equal the method of selection of the "winner" neuron for the first pattern is formally undefined in the algorithm. Although any implementation of the algorithm will have some default method of selecting a winner in such cases, even with the best possible choice of the patterns and the resulting "winner" neurons, the *minimum* number of iterations required for the development of a topologically ordered map in this case is equal to the dimensionality of the input space, and will in the general case be larger than one.

It seems that in higher dimensions true absorbing states of the algorithm do not exist and that no strict proof of convergence is likely to be possible. However, there seem to exist ordered configurations in the sense that a map will be far more likely to move towards or remain in these configurations after application of the update rule than to leave these states.

4 Energy functions

4.1 Energy functions in one dimension

Does there exist a global function such that the convergence to stationary states can be described as a stochastic gradient descent minimizing this potential? Following Ritter and Schulten (1986, 1989) we will introduce average "forces" acting on the weights and show that, in contrast to the case of a discrete pattern manifold (Ritter 1988), in the general case these forces *cannot* be derived from a potential function. A global

[2] If one plots $w_s(t)$ against s, then $N(t)$ is just the number of "turns" or "kinks" in the graph

energy function does not exist and the best we can do is to derive a set of single-neuron energy functions which allow a description of the single-neuron dynamics by conservative forces. The energy functions proposed by Tolat (1990) are shown to be an approximation which is no longer valid in the case of highly disordered maps and a steep neighborhood function.

For the following it will be convenient to relabel the weight values so that their indices are arranged in ascending order in the input space. To avoid confusion we introduce new symbols u_x to refer to weight values labeled such that $x < y \rightarrow u_x < u_y$, $(x, y \in [1,N])$ and use the "old" symbols w_s to label weight values by the position of the corresponding neurons in the network (see Fig. 1a). "New" indices can be converted to "old" indices by a permutation function $s = \mathscr{P}(x)$, which, however, is different for each configuration of the network (see Fig. 1b). Thus we may write:

$$u_x \equiv w_{\mathscr{P}(x)} \equiv w_s. \tag{9}$$

Note that the arguments of the neighborhood function are always the indices s of w_s, since the neighborhood function is defined by neighborhood relationships in the image space (network), not in the input space.

Let us denote the probability density of choosing a pattern v by $P(v)$. Then the average change $V_x[u] \equiv \langle u_x(t+1) - u_x(t) \rangle$ of the weight value u_x in one iteration, with the average taken over all possible patterns v, is given by

$$V_x[u] = \epsilon \int_0^1 \hat{h}(x, y)(v - u_x)P(v)\, \mathrm{d}v \tag{10}$$

where y is the label of the winner neuron, and we have used the abbreviation $\hat{h}(x, y)$ for $h(\mathscr{P}(x), \mathscr{P}(y))$. The quantity $V_x[u]$ may be interpreted loosely as the average force acting to either increase or decrease the value of the weight u_x at the next iteration. Expanding (10) into a sum of integrals over all possible winner neurons y yields

$$V_x[u] = \epsilon \sum_{y=1}^{N} \hat{h}(x, y) \int_{v \in \Omega(y)} (v - u_x)P(v)\, \mathrm{d}v \tag{11}$$

where each integral is evaluated only over the *Voroni tesellation cell* of the neuron y, i.e. the area of the input space mapped onto neuron y. The Voroni tessellation cell may be expressed as

$$\left.\begin{aligned} &\Omega(1) = \{v | 0 < v < \tfrac{1}{2}(u_1 + u_2)\}, \\ &\Omega(y) = \{v | \tfrac{1}{2}(u_{y-1} + u_y) < v < \tfrac{1}{2}(u_y + u_{y+1})\}, \\ &\qquad\qquad \text{for } 1 < y < N, \\ &\Omega(N) = \{v | \tfrac{1}{2}(u_{N-1} + u_N) < v < 1\}. \end{aligned}\right\} \tag{12}$$

Let us consider the simplest case, where $P(x)$ is a constant. Performing the integrations in (10) we obtain

$$\begin{aligned} V_x[u] = \sum_{y=2}^{N-1} \hat{h}(y, x)[(u_{y+1}^2 - u_{y-1}^2)/8 \\ + u_y(u_{y+1} - u_{y-1})/4 + u_x(u_{y-1} - u_{y+1})/2 \\ + \hat{h}(1,x)[(u_1 + u_2)^2/8 - u_x(u_1 + u_2)/2] \\ + \hat{h}(N, x)[1/2 - u_x - (u_N + u_{N-1})^2/8 \\ + u_x(u_N + u_{N-1})/2] \end{aligned} \tag{13}$$

The quantity $V_x[u]$ may be interpreted loosely as the average force acting on the weight u_x. Positive values of $V_x[u]$ indicate that the value of u_x is more likely to increase at the next iteration; negative values indicate that u_x is more likely to decrease. Analysis of the behavior of the mapping algorithm would be greatly simplified if the forces (13) could be derived from a potential function $E[u]$ such that

$$\partial E[u]/\partial u_x = -V_x[u]. \tag{14}$$

Unfortunately, this is not the case. The necessary and sufficient condition (Moon and Spencer 1969) for the forces to obey even a more general relation of the type

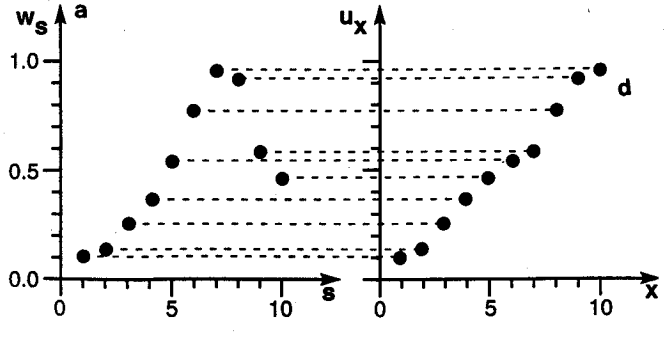
$$\partial E'[u]/\partial u_x = -\mu[u]V_x[u], \tag{15}$$

where $\mu[u]$ denotes some unknown function of the weight values, i.e. an integrating factor, is given by:

$$V_x\left[\frac{\partial V_z}{\partial u_y} - \frac{\partial V_y}{\partial u_z}\right] + V_y\left[\frac{\partial V_x}{\partial u_x} - \frac{\partial V_z}{\partial u_x}\right]$$

$$+ V_z\left[\frac{\partial V_x}{\partial u_y} - \frac{\partial V_y}{\partial u_x}\right] = 0, \quad \forall\{x, y, z\} \in \{1,2, \ldots, N\}. \tag{16}$$

By inserting (13) into (16) it is straightforward to prove that for the one-dimensional case with constant $P(v)$, condition (16) is not met in general, and thus an energy function does not exist.



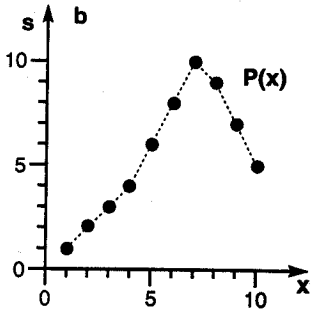


Fig. 1. **a** The indices s of the weights w_s are arranged in increasing order in the image space. The weights may be relabelled u_x with indices x arranged such that $x < y \Rightarrow u_x < u_y$. **b** The indices s of the weights w_s may be converted into the indices x of the weights u_x by the permutation function $\mathscr{P}(x)$. The permutation function is uniquely defined for each possible configuration of the weights

It has been suggested (Tolat 1990) that the dynamics of the SOFM algorithm can instead be described by a set of individual energy-functions

$$\partial E_x[u]/\partial u_x = -V_x[u], \tag{17}$$

one for each neuron x, where each weight value u_x performs a gradient descent to reduce, on the average, the value of its associated energy E_x. The motion of the weight values are coupled, however, such that it is not generally possible for the weights to change in directions which would correspond to the steepest descent of a sum of the individual energy functions.

The individual energy functions consist of two terms:

$$\begin{aligned} E_x[u] &\equiv \tilde{E}_x[u] + X_x[u] \\ &= \epsilon \sum_{y=1}^{N} \hat{h}(x,y) \int_{v \in \Omega(y)} \frac{1}{2}(v-u_x)^2 P(v)\, dv \\ &\quad + \frac{\epsilon}{48} \sum_{y=2}^{N} (u_y - u_{y-1})^3 (1 - \hat{h}(y, y-1)) \\ &\quad + P\left(\frac{1}{2}(u_y + u_{y-1})\right) \end{aligned} \tag{19}$$

which can be proven by differentiation and comparison with the forces (13). A less straightforward, but more general proof is given in Appendix B. ($P(u)$ has been assumed to be a continuous function.) The first term $\tilde{E}_x[u]$ corresponds to Tolat's ansatz for the energy function, the second term $X_x[u]$ is a correction to account for the movement of the borders of the tessellation cells during an iteration[3]. The given form of $X_x[u]$ is not unique. Other choices of $X_x[u]$ are possible, but this particular form of $X_x[u]$ was chosen because it is independent of x, i.e., it is the same for all neurons.

The major contribution to the energies comes from the integral $\tilde{E}_x[u]$. The correction terms, $X_x[u]$ are generally small; they are largest for highly disordered maps, where $(u_y - u_{y-1})$ is near one and $\hat{h}(y, y-1)$ is near zero for some values of y. The correction terms can be neglected and the approximation $E[u] \approx \tilde{E}[u]$ of Tolat becomes valid for ordered maps with sufficiently broad neighborhood functions.

4.2 Generalization to higher dimensions

The discussion above can be generalized to maps in higher dimensions. To keep the notation simple, let $\mathbf{v}$ and $\mathbf{u}_x$ represent the n-dimensional pattern and weight vectors, respectively, with each weight vector referenced by a unique, arbitrary scalar index x. We will again use the abbreviation $\hat{h}(x, y)$ to indicate that in the neighborhood function the distance between neurons arbitrarily labelled x and y must be computed from their relative positions in the network of neurons, not in the input space. For an n-dimensional input space and an image space of arbitrary dimensionality the forces are given by

$$V_x[\mathbf{u}] = \epsilon \int \hat{h}(x,y)(\mathbf{v} - \mathbf{u}_x) P(\mathbf{v})\, d^n\mathbf{v}. \tag{20}$$

The corresponding energy functions are given by (see Appendix C)

$$\begin{aligned} E_x[\mathbf{u}] &\equiv \tilde{E}_x[\mathbf{u}] + X_x[\mathbf{u}] \\ &= \lim_{\beta \to \infty} \epsilon \int \langle \hat{h}(x,y) \rangle_y \tfrac{1}{2} (\mathbf{v} - \mathbf{u}_x)^2 P(\mathbf{v})\, d^n\mathbf{v} \\ &\quad - \epsilon \int d^n\mathbf{u}_x \lim_{\beta \to \infty} \beta \int d^n\mathbf{v}\, (\mathbf{v} - \mathbf{u}_x)^3 (1 - \langle \hat{h}(x,y) \rangle_y) \\ &\quad \times \frac{\exp(-\beta(\mathbf{v} - \mathbf{u}_x)^2)}{\sum_y \exp(-\beta(\mathbf{v} - \mathbf{u}_y)^2)} P(\mathbf{v}). \end{aligned} \tag{21, 22}$$

where we have introduced the "generalized neighborhood function"

$$\langle \hat{h}(x,y) \rangle_y = \frac{\sum_y \hat{h}(x,y) \exp(-\beta(\mathbf{v} - \mathbf{u}_y)^2)}{\sum_y \exp(-\beta(\mathbf{v} - \mathbf{u}_y)^2)} \tag{23}$$

In the limit $\beta \to \infty$, (23) reduces to $\hat{h}(x, y')$, where y' is the index of the weight value $\mathbf{u}_{y'}$ closest pattern space to $\mathbf{v}$. The generalized neighborhood function (23) is introduced to remove the discontinuities in the integrand of (20) so that $V_x[\mathbf{u}]$ may be integrated to give $E_x[\mathbf{u}]$.

The second integral in (22), which is again a correction $X_x[u]$ to the naïve energy $\tilde{E}_x[u]$, is over all pattern space; however, due to the factor $(1 - \langle \hat{h}(x,y) \rangle_y)$, the only contributions to the integral in the limit of large β come from the border regions between the tessellation cells of neighboring neurons. The subsequent factor insures that only contributions from the borders of the cell around the *particular* neuron x survive in the limit. The correction terms are generally small. They again are largest for highly disordered maps, where $\langle \hat{h}(x,y) \rangle_y$ is near zero for some values of x, and they can be neglected for ordered maps if the neighborhood function is sufficiently broad.

Evaluation of (22) is difficult in general, but in Appendix B we demonstrate that for the one-dimensional case (22) reduces to (19).

5 Summary and discussion

We have shown here that the one-dimensional SOFM algorithm using any monotonically decreasing neighborhood function and a constant learning step size can be guaranteed to converge to an ordered mapping. In practical applications, however, it is observed that the rate at which the algorithm converges depends heavily on the shape of the neighborhood function. We investigate the effect of the shape of the neighborhood function on the convergence rate in a companion article (Erwin et al. 1992) where we show that fastest convergence time may be achieved using a class of neighborhood functions called "convex". For all other neighborhood functions, the equations of motion admit stable stationary states in non-ordered configurations

[3] $\Omega(y)$ is a function of $\{u_x | x = 1,2,\ldots,N\}$

which may trap maps for many iterations before finding the ordered state.

We have also shown that the update rule (3) of the self-organizing feature map algorithm does not follow a gradient descent of any energy function. One might then wonder whether it is possible to design a self-organizing algorithm, using the method of stochastic approximation (Robbins and Munro 1951), which does perform a gradient descent on a suitable potential. The gratest obstacle to finding such a method is the difficulty of determining what energy function would be appropriate. It is easy to propose a cost function which should be minimized by the ordered map, but it is much more difficult to find an energy function on which a gradient descent can be guaranteed to lead from any disordered map to a map minimizing the cost function.

The "elastic net" algorithm of Durbin and Willshaw, in the particular form suggested by Yuille, when applied to a discreet pattern manifold does perform a gradient descent on an energy function (Durbin and Willshaw 1987), as does the SOFM algorithm in this case (Ritter 1988). Simic (1989) has shown why this energy function is the function one would like to minimize in applications to the travelling salesman problem. He considers a cost function which is minimized for the shortest path through a space, and then adds terms which enforce the constraint that the path must pass through each of a set of "city" locations. Using techniques from statistical mechanics, Simic develops a "free energy" function from this cost function, and in the limit of a large number of weight vectors relative to the number of cities, this free energy is precisely the energy function on which the elastic net algorithm performs a gradient descent. Additionally, Simic derived a new form of free energy function which is appropriate for developing an "elastic net" type algorithm for solving the travelling saleman problem which does not require the use of a larger number of weight vectors relative to the number of cities. Durbin and Mitchison (1990) have proposed a more general form of the elastic net algorithm which is applicable to networks of neurons with arbitrary dimensions forming a map of either a set of discrete points or a continuous space. It is probable, however, that this more general, continuous form of the elastic net algorithm does not follow a descent on any energy function.

Luttrell (1989) considered the energy function

$$E[\mathbf{w}] = \frac{1}{2}\sum_{\mathbf{r},\mathbf{s}} h(\mathbf{r},\mathbf{s}) \int_{\Omega(\mathbf{r})} P(\mathbf{v})\, \mathrm{d}^n\mathbf{v}\, (\mathbf{v}-\mathbf{w}_\mathbf{s})^2, \qquad (24)$$

and showed that when a certain approximation holds the Kohonen algorithm follows a stochastic gradient descent on this potential[4]. Our analysis has shown, however, that this approximation is only reasonable in the least interesting case, i.e. when the map is already well ordered. Although the approximate energy function (24) can be used to describe the dynamics of the algorithm after an ordered, or mostly ordered map has formed, it is not useful for describing the ordering of an initially highly disordered map, in which case the individual energy functions (22) must be used to describe the dynamics of the original feature map algorithm. The final mapping created by the algorithm does, however, very nearly minimize (24).

It would be interesting to find whether the self-organizing algorithm would more efficiently order an initially disordered map if the update rule were modified to more closely follow a gradient descent on the energy function (24). Kohonen has recently taken this approach (Kohonen 1991), and has shown that for the case of a step neighborhood function one can achieve a self-organizing algorithm which more nearly follows a gradient descent on the potential (24) by adding small additional terms to the update rule (3). These additional terms are similar to the terms in the elastic-net algorithm which act to keep the mapping of nearest-neighbor cortical cells nearby in the input space.

By choosing the step neighborhood function (6), and making the same assumptions as Kohonen, his results can also be derived by using the techniques of (21) and (22). For the one-dimensional case, we can develop an update rule such that a gradient descent of (24) is exactly followed (unpublished results). At present it is unclear whether this approach leads to learning algorithms which represent an improvement over the original version. Although it is clear that we would like to have a learning algorithm which produces maps which minimize the energy function (24), it is not clear whether the best algorithm to use would be one which follows a gradient descent on this potential surface, since it is not clear whether metastable states are present in the energy landscape of (24), or whether a general proof of ordering can be constructed for these algorithms. Although the original update rule does not perform a gradient descent of the energy function (24), the final mapping created by the algorithm does very nearly minimize (24). For the step neighborhood function, Kohonen's revised algorithm creates an ordered mapping in fewer time steps (Kohonen 1991), with a (slightly) increased computational demand for each step. For more general neighborhood functions, the revised algorithm might require more, rather than fewer, time steps, and the computational requirements per step are certainly much greater. By introducing complicated update rules, the analogies with biological self-organizing systems may also be lost. Clearly there are still practical questions to be answered, and further research into the mechanisms driving self-organizing processes is warranted.

Acknowledgements. This research has been supported by the National Science Foundation (grant number NSF 90-15561) and by the National Institute of Health (grant number P41RR05969). Financial support to E. E. by the Beckman Institute and the University of Illinois, and of K. O. by the Boehringer-Ingelheim Fonds is gratefully acknowledged. The authors would like to thank H. Ritter for stimulating discussions and for comments on the manuscript.

[4] Luttrell considered this energy function in a different theoretical framework which also allows the number of weight vectors to be infinite and both the labels on the weights and the arguments of the neighborhood function to be real numbers

Appendix A: proof of ordering

In this appendix, we present the full proof of Theorem 1. First we must make a definition:

Definition 1. A k-chain is a set of k weight values obeying either

$$\text{case 1 (ascending):} \begin{cases} w_s < w_{s+1} < w_{s+2} < \cdots < w_{s+k-1} \\ \text{and } w_j < w_s \text{ or } w_j > w_{s+k-1} \\ \text{for all other } w_j, \end{cases}$$

$$\text{or case 2(descending):} \begin{cases} w_s > w_{s+1} > w_{s+2} > \cdots > w_{s+k-1} \\ \text{and } w_j > w_s \text{ or } w_j < w_{s+k-1} \\ \text{for all other } w_j. \end{cases}$$

In the following we will use the expression "v in the vicinity of w_r" to mean that v is closer to w_r than to any other w_j.

Lemma 1. *Applying the update rule (3) with v in the vicinity of w_r for any r does not change the relative order of w_r with respect to any other weight w_s,* i.e.

$$Sign(w_r(t) - w_s(t)) = Sign(w_r(t+1) - w_s(t+1)), \forall s.$$

Lemma 2. *Given any w_r, w_s, and $w_{s'}$, the repeated application of v in the vicinity of w_r leads after a finite number of time steps to the relative ordering of w_r, w_s and $w_{s'}$ such that*

i) $|w_r - w_s| < |w_r - w_{s'}|$ if $H(|r-s|) < H(|r-s'|)$,
ii) $|w_r - w_s| > |w_r - w_{s'}|$ if $H(|r-s|) > H(|r-s'|)$, or
iii) the relative ordering of w_r, w_s and $w_{s'}$ is unchanged if $H(|r-s|) = H(|r-s'|)$.

Lemma 3. *Given $w_s < w_{s+1} < \cdots < w_{s+k-1}$ or $w_s > w_{s+1} > \cdots > w_{s+k-1}$, ($k \geq 3$), with no conditions on the values of the other w_j, a k-chain can be constructed within a finite number of steps.*

Proof. If k is odd, choose v near $w_{s+(k-1)/2}$ repeatedly. Then from Lemmas (1) and (2) we will eventually get our k-chain. If k is even, choose v in the vicinity of $w_{s+(k-2)/2}$ unril a $(k-1)$-chain is constructed from w_s to w_{s+k-2}. Then, if necessary, choose v near w_{s+k-2} until w_{s+k-1} is greater than w_{s+k-2} and closer to it than to any other weight.

Proposition 1. *Given any state $\{w_i | i = 1,2,\ldots,N; w_i \in [0,1]\}$, it is possible to find a sequence of v values such that either an ascending or descending 3-chain can be constructed around any w_s, $s \neq 1$, $s \neq N$, by using the update rule* (3).

Proof. For a given w_s, $s \neq 1, N$, repeatedly choose v in the vicinity of w_s. Lemma 2 then ensures that after a sufficient time

$$|w_s - w_{s+1}| < |w_s - w_j| \text{ and}$$
$$|w_s - w_{s-1}| < |w_s - w_j| \ \forall j \notin \{s-1, s, s+1\}.$$

Depending on the initial values of the weights, one arrives at one of the six following cases:

1. $w_{s-1} < w_s < w_{s+1}$ 2. $w_{s+1} < w_s < w_{s-1}$
3. $w_s < w_{s+1} < w_{s-1}$ 4. $w_s < w_{s-1} < w_{s+1}$
5. $w_{s+1} < w_{s-1} < w_s$ 6. $w_{s-1} < w_{s+1} < w_s$

Cases 1 and 2 are already 3-chains. Lemma 2 ensures that repeated application of (3) with v in the vicinity of w_{s-1} or w_{s+1} will lead to an ascending or descending ordering of the three weights for the cases 6, 3 and 4, 5, respectively. Lemma 3 then ensures that an ascending or descending 3-chain may be constructed.

Proposition 2. *Given any ascending (descending) k-chain ($k < N$), it is possible to find a sequence of $v(t)$ such that an ascending (descending) $(k+1)$-chain will result.*

Proof. We will consider only the ascending case: $w_s < w_{s+1} < \cdots < w_{s+k-1}$. There are three sub-cases:

1. If $w_{s+k} > w_{s+k-1}$, Lemma 3 ensures that an ascending $k+1$-chain can be constructed from w_s to w_{s+k}.

2. If $w_{s-1} < w_s$, Lemma 3 again ensures that an ascending $k+1$-chain can be constructed from w_{s-1} to w_{s+k-1}.

3. If the above conditions do not apply then $w_{s-1} > w_{s+k-1}$ and/or $w_{s+k} < w_s$. If $s > 1$ then choose v near w_s until w_{s-1} is closer to w_s than w_{s+2}. Now choose v near w_{s+2} until $w_{s-1} < w_s$. Now the $k+1$ weights from w_{s-1} to w_{s+k-1} are in ascending order, and from Lemma 3, a $(k+1)$-chain may be constructed. If $s = 1$ then we must work from the other end of the chain. Choose v near w_{s+k-1} until w_{s+k} is closer to w_{s+k-1} than is w_{s+k-3}. Next choose v near w_{s+k-3} until $w_{s+k} > w_{s+k-1}$.

The proof of Theorem 1 follows directly from Propositions 1 and 2.

Appendix B: calculation of $E_x[u]$ in the one-dimensional case

We wish to show that $E_x[u] \equiv \tilde{E}_x[u] + X_x[u]$ is the correct energy function describing the dynamics of the weight vector u_x, for the one-dimensional case. To do this we could explicitly calculate $\partial E_x[u]/\partial u_x$ from (18) and show that this gives $-V_x[u]$, but instead we will demonstrate the usefulness of the general expression (22) by calculating $E_x[u]$ from it.

From (21) and (22) we can write $\tilde{E}_x[u]$ for the one-dimensional case by simply changing all vectors to scalars; this accounts for the first sum in (19). We have left to show that $X_x[u]$, given by

$$-\partial X_x[u]/\partial u_x = \lim_{\beta \to \infty} \epsilon\beta \int_0^1 P(v)\, dv\, (v - v_x)^3 \times (1 - \langle \hat{h}(x,y) \rangle_y) \times \frac{\exp(-\beta(v-u_x)^2)}{\sum_y \exp(-\beta(v-u_y)^2)}, \tag{25}$$

will simplify to the second sum in (19). The factor $\exp(-\beta(v-u_x)^2)/\Sigma_y \exp(-\beta(v-u_y)^2)$ will go to zero as β goes to infinite unless v is closer to u_x than to any other weight. However, the factor $(1 - \langle \hat{H}(x,y) \rangle_y)$ goes to $(1 - \hat{h}(x,x)) = 0$ if v is closest to u_x. Thus the

only contribution to the integral comes from the infinitesimally small border regions between the tessellation of u_x and its neighbors.

Let us look at the border between u_x and u_{x-1} ($x \neq 1$). As β goes to infinity, it suffices to integrate over the region $v = (u_x + u_{x-1})/2 - \lambda$ to $v = (u_x + u_{x-1})/2 + \lambda$, where λ is some small number. The value of λ may be chosen arbitrarily, since in the limit of large β the only contribution to the integral comes from the region of pattern space where the distance between v and the border is no larger than the order of $1/\beta$. The contribution to (25) from this region of the input space is then

$$\lim_{\beta\to\infty} \epsilon\beta \int_{(u_x+u_{x-1})/2-\lambda}^{(u_x+u_{x-1})/2+\lambda} P(v)\, dv (v-u_x)^3 \times \left[1 - \frac{\hat{h}(x,x)\exp(-\beta(v-u_x)^2 + \hat{h}(x,x-1)\exp(-\beta(v-u_{x-1})^2)}{\exp(-\beta(v-u_x)^2) + \exp(-\beta(v-u_{x-1})^2}\right] \times \left[\frac{\exp(-\beta(v-u_x)^2)}{\exp(-\beta(v-u_x)^2) + \exp(-\beta(v-u_{x-1})^2)}\right], \quad (26)$$

where we have used the definition of $\langle \hat{h}(x,y)\rangle_y$ and where we have left only the largest terms as $\beta \to \infty$ in the fractions. By making the substitutions $\hat{h}(x,x) = 1$, $v = (u_x + u_{x-1})/2 + \lambda'$ and integrating over λ' instead of v we may write, after some rearrangement,

$$\lim_{\beta\to\infty} \epsilon\beta \int_{-\lambda}^{\lambda} d\lambda' P((u_x+u_{x-1})/2+\lambda') \times (\lambda'(u_x-u_{x-1})/2)^3 \times \left[\frac{(1-\hat{h}(x,x-1))\exp(-\alpha\beta\lambda')}{(1+\exp(-\alpha\beta\lambda'))^2}\right], \quad (27)$$

where $\alpha = 2(u_x - u_{x-1})$. If $P(v)$ is a continuous function then $P((u_x+u_{x-1})/2+\lambda') \approx P((u_x+u_{x-1})/2)$ for small λ'. For any constant value of λ, as $\beta \to \infty$ we find the identity

$$\lim_{\beta\to\infty} \int_{-\lambda}^{\lambda} \frac{(\lambda')^n \beta \exp(-\alpha\beta\lambda')}{(+\exp(-\alpha\beta\lambda'))^2} d\lambda' = \begin{cases} 1/\alpha & \text{for } n = 0 \\ 0 & \text{for } n = 1, 2, 3. \end{cases} \quad (28)$$

Using this identity we may evaluate integral (27) to obtain

$$-(\epsilon/16)(u_x-u_{x-1})^2 P((u_x+u_x-1)/2) \times (1-\hat{h}(x,x-1)). \quad (29)$$

Likewise, by integrating (25) in the region around the border between u_x and u_{x+1} ($x \neq N$) gives a contribution of

$$+(\epsilon/16)(u_{x+1}-u_x)^2 (P((u_{x+1}+u_x)/2) \times (1-\hat{h}(x,x+1)). \quad (30)$$

So overall,

$$-\partial X_x[u]/\partial u_x = -(\epsilon/16)(u_x-u_{x-1})^2 P((u_x+u_{x-1})/2) \times (1-\hat{h}(x,x-1))(1-\delta_{x1}) + (\epsilon/16)(u_{x+1}-u_x)^2 P((u_{x+1}+u_x)/2) \times (1-\hat{h}(x,x+1))(1-\delta_{xN}).$$

Integration over u_x gives

$$X_x[u] = (\epsilon/48)\sum_{y=2}^{N} (u_x - u_{x-1})^3 (1-\hat{h}(x,x-1)) \times P((u_x+u_{x-1})/2), \quad (31)$$

plus an arbitrary constant which we have set to zero. Then $E_x[u] = \tilde{E}_x[u] + X_x[u]$, as in (18).

Appendix C: derivation of $X_x[\mathbf{u}]$ in the multi-dimension case

We wish to show that (21) is the correct definition of the energy $E_x[\mathbf{u}]$ by showing that $\partial E_x[\mathbf{u}]/\partial \mathbf{u}_x = -V_x[\mathbf{u}]$. We start with the uncorrected energy term $\tilde{E}_x[\mathbf{u}]$. Inserting (21) and (22) into $\partial \tilde{E}_x[\mathbf{u}]/\partial \mathbf{u}_x$ gives

$$\frac{\partial \tilde{E}_x[\mathbf{u}]}{\partial \mathbf{u}_x} = \lim_{\beta\to\infty}\left[-\epsilon\int_0^1 \langle \hat{h}(x,y)\rangle_y (\mathbf{v}-\mathbf{u}_x) P(\mathbf{v})\, d^n\mathbf{v} + \frac{\epsilon}{2}\int_0^1 (\mathbf{v}-\mathbf{u}_x)^2 \frac{\partial}{\partial \mathbf{u}_x}\langle \hat{h}(x,y)\rangle_y P(\mathbf{v})\, d^n\mathbf{v}\right]. \quad (32)$$

The first term in (32) is equivalent to $-V_x[\mathbf{u}]$. Performing the derivative on the second term yields

$$\frac{\partial}{\partial \mathbf{u}_x}\langle \hat{h}(x,y)\rangle_y = \frac{\partial}{\partial \mathbf{u}_x}\left[\frac{\sum_y \hat{h}(x,y)\exp(-\beta(\mathbf{v}-\mathbf{u}_y)^2)}{\sum_y \exp(-\beta(\mathbf{v}-\mathbf{u}_y)^2)}\right]$$

$$= \left(\sum_y \exp(-\beta(\mathbf{v}-\mathbf{u}_y)^2)\right)^{-2} \times \left[\left(\sum_y \exp(-\beta(\mathbf{v}-\mathbf{u}_y)^2)\right)\hat{h}(x,x) \times \exp(-\beta(\mathbf{v}-\mathbf{u}_x)^2)(2/\beta(\mathbf{v}-\mathbf{u}_x)) - \left(\sum_y \hat{h}(x,y)\exp(-\beta(\mathbf{v}-\mathbf{u}_y)^2)\right) \times \exp(-\beta(\mathbf{v}-\mathbf{u}_x)^2)(2\beta(\mathbf{v}-\mathbf{u}_x))\right]$$

$$= 2\beta(\mathbf{v}-\mathbf{u}_x)(1-\langle \hat{h}(x,y)\rangle_y)\frac{\exp(-\beta(\mathbf{v}-\mathbf{u}_x)^2)}{\sum_y \exp(-\beta(\mathbf{v}-\mathbf{u}_y)^2)} \quad (33)$$

where we have used the fact $\hat{h}(x, x) \equiv 1$. Substituting (33) and (20) in (32) and reducing the first term to $-V_x[\mathbf{u}]$ gives

$$\partial \tilde{E}_x[\mathbf{u}]/\partial \mathrm{u}_x = -V_x[\mathbf{u}] + \lim_{\beta \to \infty} \epsilon\beta \int_0^1 (\mathbf{v} - \mathbf{u}_x)^3 \times (1 - \langle \hat{h}(x, y) \rangle_y) \times \frac{\exp(-\beta(\mathbf{v} - \mathbf{u}_x)^2)}{\Sigma_y \exp(-\beta(\mathbf{v} - \mathbf{u}_y)^2)} P(\mathbf{v})\, \mathrm{d}^n\mathbf{v}. \quad (34)$$

Comparison of the last term with (22) gives

$$\partial \tilde{E}_x[\mathbf{u}]/\partial \mathbf{u}_x = -V_x[\mathbf{u}] - \partial X_x[\mathbf{u}]/\partial \mathbf{u}_x. \quad (35)$$

Thus

$$E_x[\mathbf{u}] \equiv \tilde{E}_x[\mathbf{u}] + X_x[\mathbf{u}], \quad (36)$$

fulfills $\partial E_x[\mathbf{u}]/\partial \mathbf{u} = -V_x[\mathbf{u}]$.

References

Cottrell M, Fort JC (1987) Etude d'un processus d'auto-organization. Ann Inst Henri Poincaré 23:1–20

Durbin R, Mitchison G (1990) A dimension reduction framework for understanding cortical maps. Nature 343:644–647

Durbin R, Willshaw D (1987) An analogue approach to the travelling salesman problem using an elastic net method. Nature 326:689–691

Erwin E, Obermayer K, Schulten K (1992) Self-organizing maps: Stationary states, metastability and convergence rate. Biol Cybern (this issue)

Favata F, Walker R (1991) A study of the application of Kohonen-type neural networks to the travelling salesman problem. Biol Cybern 64:463–468

Geszti T (1990) Physical models of neural networks. World Scientific, Singapore

Hertz J, Krogh A, Palmer RG (1991) Introduction to the theory of neural computation. Addison-Wesley, Redwood City, California

Kohonen T (1982a) Analysis of a simple self-organizing process. Biol Cybern 44:135–140

Kohonen T (1982b) Self-organized formation of topologically correct feature maps. Biol Cybern 43:59–69

Kohonen T (1988) Self-organization and associative memory. Springer, New York Berlin Heidelberg

Kohonen T (1989) Speech recognition based on topology preserving neural maps. In: Aleksander I (ed) Neural computation. London, Kogan Page

Kohonen T (1991) Self-organizing maps: optimization approaches. In: Kohonen T et al. (eds) Artificial neural networks, vol II. North Holland, Amsterdam, pp 981–990

Lo ZP, Bavarian B (1991) On the rate of convergence in topology preserving neural networks. Biol Cybern 65:55–63

Luttrell SP (1989) Self-organization: a derivation from first principles of a class of learning algorithms. In: Proc. 3rd IEEE Int. Joint Conf. on Neural Networks, vol II. Washington, pp 495–498, IEEE Neural Networks Council

Moon P, Spencer DE (1969) Partial differential equations. Heath, Lexington, Mass

Obermayer K, Ritter H, Schulten K (1990a) Large-scale simulations of self-organizing neural networks on parallel computers: Applications to biological modelling. Parallel Computer 14: 381–404

Obermayer K, Ritter H, Schulten K (1990b) A principle for the formation of the spatial structure of cortical feature maps. Proc Natl Acad Sci USA 87:8345–8349

Obermayer K, Blasdel GG, Schulten K (1991) A neural network model for the formation of the spatial structure of retinotopic maps, orientation- and ocular dominance columns. In: Kohonen T et al. (eds) Artificial neural networks, vol I. North Holland, Amsterdam, pp 505–511

Ritter H (1988) Selbstorganisierende neuronale Karten. PhD thesis. Technische Universität München

Ritter H, Kohonen T (1989) Self-organizing semantic maps. Biol Cybern 61:241–254

Ritter H, Schulten K (1986) On the stationary state of Kohonen's self-organizing sensory mapping. Biol Cybern 54:99–106

Ritter H, Schulten K (1988) Convergence properties of Kohonen's topology conserving mappings: Fluctuatuions, stability and dimension selection. Biol Cybern 60:59–71

Ritter H, Martinetz T, Schulten K (1989) Topology conserving maps for learning visuomotor-coordination. Neural Networks 2:159–168

Robbins H, Munro S (1951) A stochastic approximation method. Ann Math Statist 22:400–407

Scholtes JC (1991) Unsupervised context learning in natural language processing. In: Proc 5th IEEE Int. Joint Conf. on Neural Networks, vol I. Washington, pp 107–112, IEEE Neural Networks Council

Simic PD (1989) Statistical mechanics as the underlying theory of 'elastic' and 'neural' optimizations. Network 1:89–103

Tolat VV (1990) An analysis of Kohonen's self-organizing maps using a system of energy functions. Biol Cybern 64:155–164

van Kampen NG (1981) Stochastic process in physics and chemistry. North-Holland, Amsterdam