ESANN'1994 proceedings - European Symposium on Artificial Neural Networks
Brussels (Belgium), 20-21-22 April 1994, D-Facto public., ISBN 2-9600049-1-4, pp. 263-269

Self-Organizing Maps Based on Differential Equations

Andreas Kanstein and Karl Goser

Universität Dortmund, Fakultät für Elektrotechnik,
D-44221 Dortmund, Germany.
E-mail: kanst@luzi.e-technik.uni-dortmund.de

Abstract. As an example of real parallel processing in self-organizing maps this paper presents an adaptation algorithm based on a differential equation. This new model of self-organizing maps uses an equation of the type of the Schrödinger wave equation to compute the activation of the map neurons. The amplitude function gives both a fuzzy activation and the interaction of the processing elements. The interaction thereby is not fixed to an invariant shape. The new features of this model are demonstrated in relation to Kohonen maps. In a map enhanced with output units the fuzzy activation offers the possibility of an interpolative computation of the system's output value. The ordering properties of this enhanced network are tested with a well known classification task.

1 Introduction

Self-organizing maps are types of artificial neural networks characterized by competitive learning of the processing elements (neurons) [1]. Therefore they can be used for pattern analysis of complex input signals. Kohonen first introduced self-organizing maps as a model of brain mapping of sensory inputs [2]. Later the physical model was formulated in a more abstract manner, relating the learning mechanism to vector quantisation algorithms (e.g., [3]). In comparison to the physical model, this formulation has a major drawback. Only the best matching element gives the active response of the net, and the interaction of the processing units is fixed to a constant shape. This makes the algorithm mathematically more plausible, but inherits a quite artificial formulation of self-organization. The computation of the best matching element is a process of global interaction, and therefore a fundamental drawback in the fully parallel neural network structure. Furthermore the convergence rate of this adaptation process is slow.

An alternative is a self-organizing map where the activation of the map is described by the solution of a differential equation. With a differential equation, a global function is defined locally and can be calculated in a fully parallel manner. Tryba and Goser [4, 5] first proposed an adaptation scheme which uses a differential equation similar to the time-dependent Schrödinger equation. But the iteration of the complex valued variables is very time consuming and the control of the parameters is difficult. Here this idea becomes more practical by

ESANN'1994 proceedings - European Symposium on Artificial Neural Networks
Brussels (Belgium), 20-21-22 April 1994, D-Facto public., ISBN 2-9600049-1-4, pp. 263-269

the implementation of a time independent, real-valued differential equation, as will be shown in the next section.

The solution of the differential equation is regarded as a fuzzy activation of the map, which determines the adaptation of the weights. This increases the convergence rate [6], as explained in Sect. 3. It is also used to expand the map by a projection layer for real-valued outputs, similar to the projection scheme in the network model of Demartines and Hérault [7]. This will be demonstrated in Sect. 4.

2 The Adaptation Process

Self-organizing maps consist of two neuron layers, the input layer and the map layer. The input vector $\mathbf{x} \in \mathbb{R}^k$ is fed into the net through k input neurons. Every input neuron is connected to all n neurons of the map layer, called processing elements. The connections of the layers (the weights) are therefore described by vectors $\mathbf{w}_i \in \mathbb{R}^k$, $i = 1 \ldots n$.

The processing elements are arranged in a linear or rectangular array. They interact locally with their neighbours, competing for the representation of the input vector. The competition depends on the distances

$$d_i = \|\mathbf{x} - \mathbf{w}_i\| \quad , \quad i = 1 \ldots n \ . \tag{1}$$

Learning takes place iteratively in cycles, in which all training vectors are fed into the network in a random sequence. For simplicity the following considerations refer to the linear case (a linear map), but can be generalized to two-dimensional maps.

2.1 Original Algorithm

The original adaptation algorithm of Kohonen [3] is formulated as follows. For every input $\mathbf{x}$, the weight vectors are adapted as

$$\mathbf{w}_i \rightarrow \mathbf{w}_i + h_{ci}(t)\,(\mathbf{x} - \mathbf{w}_i) \quad , \quad i = 1 \ldots n \ . \tag{2}$$

Therein $c = \mathrm{argmin}_i\{\|\mathbf{x} - \mathbf{w}_i\|\}$ is the index of the best matching parameter vector. The lateral interaction of the processing elements is given by the interaction strength $h_{ci}(t)$, which depends on time, i.e. the interaction strength decreases with time, and on the distance of the weight vectors $\mathbf{w}_i$ and $\mathbf{w}_c$ within the map. The lateral interaction is essential for the spatial order of the weight vectors.

In respect to a fast parallel implementation, this algorithm has certain disadvantages. First, the time-invariant interaction shape is insensitive to the structural knowledge stored in the map. Therefore the topology of the neighbourhood is often redefined, especially during the first learning cycles. Previously learned structures are eliminated, or at least distorted. Second, the calculation of the best matching element needs a global communication within the map. This is

ESANN'1994 proceedings - European Symposium on Artificial Neural Networks
Brussels (Belgium), 20-21-22 April 1994, D-Facto public., ISBN 2-9600049-1-4, pp. 263-269

uncommon to neural network algorithms, because neurons communicate directly with each other. In addition, for effective parallel computation there should be only local connections of neurons.

2.2 Algorithm with Differential Equation

Differential equations fulfil the requirement of parallel computation in an ideal way, because they describe functions by the relationship of every function value to its infinitesimal surrounding. The aim of the new adaptation algorithm is to describe the activation shape with a differential equation. It follows from the original algorithm that the activation should be highest where the distance value is lowest. Furthermore it should decrease slowly with the lateral distance of processing elements, thereby describing the interaction.

In simulations model equations like the diffusion and the wave equation did not show these properties. Only equations derived from the Schrödinger wave equation proved to be successful, because of the phenomena of tunneling of quantum mechanical particles. Quantum mechanical particles will always find the location with lowest potential because they "tunnel" through higher energy barriers [8]. Of the solutions of the Schrödinger wave equation here only the solution with lowest energy value is interesting. This amplitude function is bounded and does not have nodes, i.e. it has only positive values. Its maximal value is located at or near the absolute minimum of the potential.

Because of this property the amplitude function is suitable to describe the activation of the processing elements. Tryba and Goser [4] presented an adaptation algorithm which is based on the time dependent Schrödinger equation. But simulations of that algorithm are computationally very expensive because of the complex-valued wave function and the iteration of the time variable. With a differential equation similar to the time independent Schrödinger equation the algorithm becomes quite faster. In

$$-\frac{1}{\gamma^2}\frac{\partial^2}{\partial x^2}\psi(x) + d(x)\psi(x) = \epsilon\,\psi(x) \tag{3}$$

the distance function $d(x)$ replaces the potential function. The interpretation of the distance function as a potential is common for physical models of neural networks.

Because the distance is known only at single locations, at the processing elements, (3) is used in the discrete form (approximation by Taylor series expansion)

$$-\alpha\,(\psi_{i-1} - 2\psi_i + \psi_{i+1}) + d_i\,\psi_i = \epsilon\,\psi_i \tag{4}$$

with $\alpha := 1/\gamma^2\Delta_x$.

To solve (4) for all $i = 1 \ldots n$, boundary conditions for ψ have to be chosen. There are mainly two possibilities, either $\psi_0 = \psi_2$ and $\psi_{n+1} = \psi_{n-1}$ [4], or $\psi_0 = \psi_n$ and $\psi_{n+1} = \psi_1$. Here the second possibility is selected, because the map is then circular and mathematically ψ and d are periodically continued.

ESANN'1994 proceedings - European Symposium on Artificial Neural Networks
Brussels (Belgium), 20-21-22 April 1994, D-Facto public., ISBN 2-9600049-1-4, pp. 263-269

As mentioned before, only the solution with lowest eigenvalue ϵ has the properties of the activation being sought. With this amplitude function, the adaptation of the new algorithm takes the form

$$\mathbf{w}_i \rightarrow \mathbf{w}_i + \psi_i(t)\,(\mathbf{x} - \mathbf{w}_i) \quad , \quad i = 1 \ldots n \ . \tag{5}$$

The activation function $\psi(t)$ depends on time, because the parameter α in (4) will change with time. The selection of the parameter values is discussed in the next section.

3 Influence of the Adaptation Parameters

The activation function ψ here also describes the lateral interaction of the processing elements. During training, the interaction should vary in the following way: In the beginning the aim of adaptation is the topological order, therefore the interaction (activation) is wide. Later, close to the end of learning, the aim is to improve the matching of the input and weight vectors. The activation is concentrated at the absolute minimum of the distance.

The width of the activation is controlled by the parameter α in (4). On the left hand side of the equation there are two summands. The first one gives the influence of the curvature of ψ, the second the influence of the distance d. The parameter α determines the ratio of these two portions. If the curvature is predominant (large α), then ψ will be wide. If α gets smaller, ψ becomes more concentrated around low values d (Fig. 1). Obviously the adaptation ψ_i

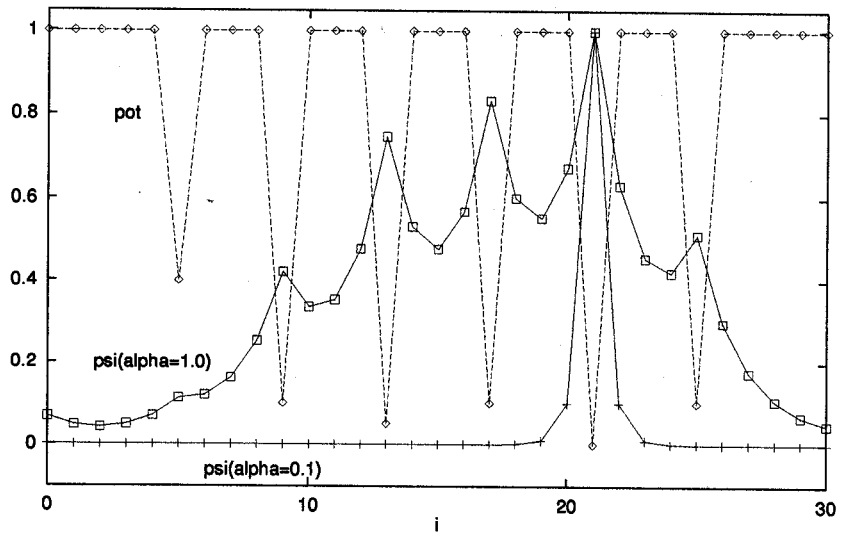

Fig. 1. Examples of solutions ψ_i (psi) of (4) for arbitrary d_i (pot) and two values of α, 1.0 and 0.1. The value of ϵ is 0.70 for $\alpha = 1.0$ and 0.18 for $\alpha = 0.1$.

ESANN'1994 proceedings - European Symposium on Artificial Neural Networks
Brussels (Belgium), 20-21-22 April 1994, D-Facto public., ISBN 2-9600049-1-4, pp. 263-269

strongly depends on the values d_i. Therefore previously learnt weights are taken into consideration and the convergence speed increases [6].

To control adaptation during training, α will be large in the beginning and getting smaller to the end. In addition ψ is scaled down after every training cycle, so $\max(\psi)$ and the training strength will decrease. The weight vectors are initialized either randomly or linearly at the beginning of training.

4 Fuzzy Activation

The activation ψ can be interpreted as a *fuzzy activation.* Unlike in Kohonen's self-organizing map the activation of the map layer spreads over some of the processing elements. This feature is used to implement a nonlinear mapping $\mathbb{R}^k \rightarrow \mathbb{R}^l$ in a projection network that consists of the new self-organizing map expanded by an output layer (Fig. 2). The adaptation of the output weight vectors $\mathbf{x}_{\text{out}} \in \mathbb{R}^l$ is similar to that of the input vectors. During training, a vector $\mathbf{x}_{\text{in}}$ and a corresponding output $\mathbf{x}_{\text{out}}$ are fed into the network. After calculation of the distances $d_i = \|\mathbf{x}_{\text{in}} - \mathbf{w}_{\text{in}i}\|$ and the activation ψ both $\mathbf{x}_{\text{in}}$ and $\mathbf{x}_{\text{out}}$ are adapted according to (5).

At the end of training every unit holds quantisation vectors of the inputs and the corresponding outputs. For recall, an input vector $\mathbf{x}_{\text{in}}$ is fed into the network and the distances d_i and the fuzzy activation ψ are computed. The output is

$$x_{\text{out}j} = \frac{\sum_i \psi_i w_{\text{out}ij}}{\sum_i \psi_i} \quad , \quad j = 1 \ldots l \ . \tag{6}$$

This network can be used for a fuzzy classification of data sets, which is demonstrated with the following example.

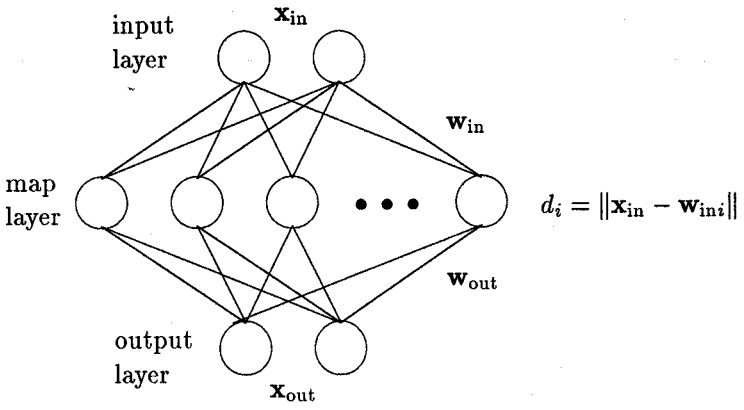

Fig. 2. Structure of a projection network consisting of the new self-organizing map enhanced by an output layer.

ESANN'1994 proceedings - European Symposium on Artificial Neural Networks
Brussels (Belgium), 20-21-22 April 1994, D-Facto public., ISBN 2-9600049-1-4, pp. 263-269

5 Example

A common example for classification algorithms, the classification of Anderson's Iris data [9], will illustrate the ordering properties of the net. The data set has 150 four-dimensional vectors (SL, SW, PL, PW), equally distributed among the classes (subspecies) *setosa*, *versicolor* and *virginica*. The data values were nor-

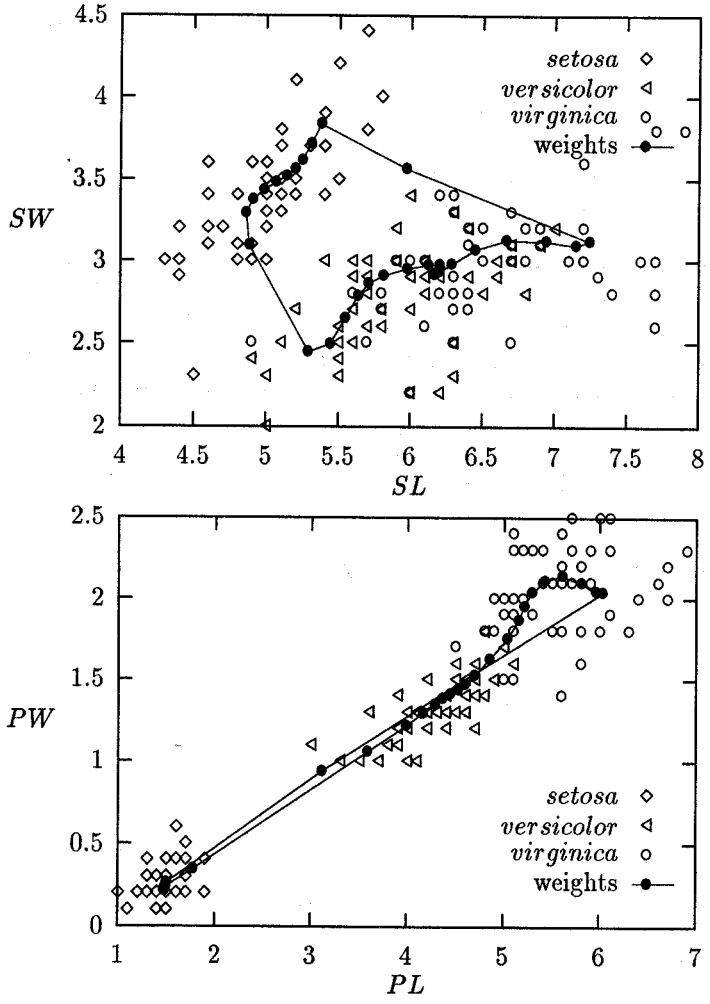

Fig. 3. Location of the 30 weight vectors in respect to the input vectors (Iris-data). The weight vectors try to mimic the distribution of the learning vectors under the constraint of their spatial order. This result was obtained after 10 learning cycles (1500 adaptations) with α decreasing from 1.0 to 0.05 and max(ψ) decreasing from 0.8 to 0.3.

ESANN'1994 proceedings - European Symposium on Artificial Neural Networks
Brussels (Belgium), 20-21-22 April 1994, D-Facto public., ISBN 2-9600049-1-4, pp. 263-269

malized to have real values in $[0, 1]$. The output vectors are three-dimensional. Every unit denotes the membership of the corresponding input vector to a specific class, e.g. $\mathbf{x}_{\text{out}} = (1, 0, 0)$ for input vectors belonging to *setosa*.

This data was used to train a map with 30 processing elements. The spread of the weight vectors in respect to the input vectors is shown in Fig. 3. The weight vectors try to mimic the distribution of the learning vectors. The more concentrated a cluster of input vectors, the closer the weight vectors. With the output calculation described in the preceding section, the reference data could be classified with only 6 misclassifications.

This example has been calculated using a simulation program written in C language. The solution of (4) is computed with Monte-Carlo methods, which makes the simulation quite slow. The adaptation of this example took 7 minutes on a sun sparc10 workstation.

6 Conclusions

With a new kind of adaptation algorithm where a differential equation determines the activation of the processing elements real self-organization is achieved. The new features are the really parallel computation and the fuzzy activation of the processing elements, which increases the convergence rate. At the moment the drawback is the large amount of computation effort, which will be overcome by a new solving algorithm now under test. This new algorithm can easily be implemented in an electronic network of real processing elements.

References

1. T. Kohonen. The self-organizing map. *Proc. of the IEEE*, 78:1464–1480, 1990.
2. T. Kohonen. Automatic formation of topological maps of patterns in a self-organizing system. In *Proceedings of the 2nd Scandinavian Conference on Image Analysis*, pages 1–7, Helsinki, 1981.
3. T. Kohonen. Self-organizing maps: Optimization approaches. In *Artificial Neural Networks*, pages 981–990. Elsevier, 1991.
4. V. Tryba and K. Goser. A modified algorithm for self-organizing maps based on the Schrödinger equation. In *Proceedings of IWANN '91*, Granada, 1991.
5. V. Tryba and K. Goser. Three algorithms for searching the minimum distance in kohonen maps. In *Digest of ESANN '93*, Brussels, 1993.
6. H. Surmann, B. Möller, and K. Goser. A distributed self-organizing fuzzy rule-based system. In *Proceedings of Neuro-Nîmes '92*, pages 187–194, Nîmes, 1992.
7. P. Demartines and J. Hérault. Representation of nonlinear data structures through fast VQP neural network. In *Proceedings of Neuro-Nîmes '93*, Nîmes, 1993.
8. P. Ruján. Searching for optimal configurations by simulated tunneling. *Z. Phys. B – Condensed Matter*, 73:391–416, 1988.
9. J. C. Bezdek. Numerical taxonomy with fuzzy sets. *Journal of Mathematical Biology*, 1:57–71, 1974.