ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1996
Seria: GEOMETRIA I GRAFIKA INŻYNIERSKA z. 1 Nr kol. 1318

M. **HOFFMANN**
L. **VÁRADY** AND T. **MOLNÁR**
Lajos Kossuth University
Dept. of Mathematics and Informatics
H-4010 Debrecen

APPROXIMATION OF SCATTERED DATA BY DYNAMIC NEURAL NETWORKS

Summary. A new method of manipulation of two dimensional scattered data is presented in this paper. The scattered input points are ordered by a self-organizing neural network. For this purpose the dynamic version of the Kohonen network is used.

APROKSYMACJA DANYCH ROZPROSZONYCH ZA POMOCĄ DYNAMICZNEJ SIECI KOMÓREK NERWOWYCH

Streszczenie. W pracy przedstawiono nowy sposób porządkowania dwuwymiarowo rozproszonych danych, rozumianych jako zbiór punktów na płaszczyźnie o nieustalonej kolejności łączenia. Sposób polega na wykorzystaniu zdolności samoorganizowania się sieci komórek nerwowych, w szczególności na dynamicznej wersji tzw. modelu Kohonena.

INTRODUCTION

The interpolation and the approximation of scattered data are interesting problems of computer graphics. By scattered data we mean a set of points without any predefined order. Unfortunately all the standard interpolation and approximation methods – like Hermite interpolation, Bezier curves or B-spline - need a sequence of points (for the sake of simplicity the problems will be discussed only in two dimensions), hence if we want to apply these methods we have to order the data.

A good survey of the scattered data interpolation can be found in [1]. In this paper a completely new approach is given where the self-organizing ability of the neural networks will be used to order the points. The Kohonen network [2,3] can be trained by scattered data, that is the points will form the

This research was supported by the Hungarian National Foundation for Scientific Research (OTKA), grant No. F007262, No. 1651 and No. W015124

input of the network, while the weights of the network and their connections give us a polygon, the vertices of which are the input points. However this process needs a good estimation of the number of output neurons in advance, because if this number is too small then some of the input points can be out of the result polygon, while if this number is too large, then the processing time will increase. To avoid these problems we use the dynamic version of the Kohonen network developed by Fritzke [5,6], where the number of output points is growing dynamically through the iterations. During the selforganizing proces some input points will be approached enough by the output neurons. To decrease the number of iterations these imputpoints will not be presented to the net in the future iterations.

In this way the polygon we obtained can be used as the control polygon of a B-spline curve, so finally a standard approximation or interpolation method can be applied for the scattered data.

We begin our discussion with the short definition of Kohonen's neural network.

KOHONEN'S NEURAL NETWORK

TOPOLOGY AND TRAINING RULES OF THE KOHONEN NET

The Kohonen net is a two-layered non-supervised learning network. The first layer of neurons is called input layer and contains the input neurons which pick up the data. The input neurons are entirely interconnected to a second layer, the competitive layer. The weights associated with the connections are adjusted during the training. Only one single neuron can be active at a time and this neuron represents the cluster which the input data set belongs to. The following figure shows the topology of a Kohonen map with two dimensional output layer.

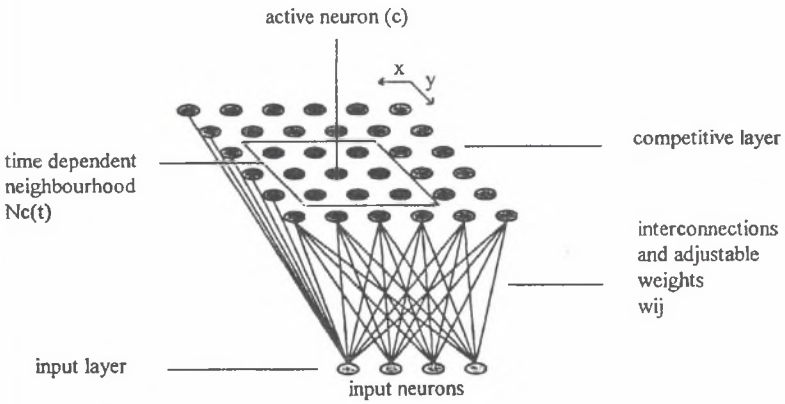

Figure 1

The training of the network is figured out by presenting data vectors x to the input layer of the network whose connection weights w_i of all competitive neurons (i = 1,...,m) are initially chosen as random values. If n is the dimension of the input we choose n input neurons and define a Euclidean distance between x and w_i with

$$d_i = \|x - w_i\| = \sqrt{\sum_{j=1}^{n}(x_j - w_{ij})^2},$$

where x_j denotes the jth coordinate of the presented input vector.

The neuron c with the minimum distance is then activated, where $d_c = \min_i (d_i)(i = 1,...,m)$. The updating of the weights w_{ij} associated to the neurons is only performed within a neighbourhood ($i \in N_c(t)$) of c, (j = 1,...,n). This neighbourhood $N_c(t)$ is reduced with training time t. This updating follows the equation

$$w_{ij}^{(t+1)} = w_{ij}^{(t)} + \Delta w_{ij}^{(t)}, \quad (i = 1,...m;\ j = 1,...n)$$

$$\Delta w_{ij}^{(t)} \begin{cases} \eta(t) \cdot (x_j - w_{ij}^{(t)}, & \text{if neuron } i \in N_c(t) \\ 0 & \text{otherwise} \end{cases}$$

$$\eta(t) = \eta_0 \cdot \left(1 - \frac{t}{T}\right), \quad \text{where } t \in [0,...,T]$$

where $\eta(t)$ represents a time-dependent learning rate which is decreasing in time. The term $\eta(t)$ can be chosen as the Gaussian function $\eta(t) = \frac{1}{\sqrt{2 \cdot \pi}} \cdot e^{\frac{-t^2}{2}}$.

The time dependent neighbourhood $N_c(t)$ can be described as a rectangular areas in the following way:

$$x_c - d(t) < x < x_c + d(t)$$
$$y_c - d(t) < y < y_c + d(t)$$
$$d(t) = d_0 \left(1 - \frac{t}{T}\right), t \in [0,...,T]$$

After updating the weights w_{ij} a new input is presented and the next iteration starts.

Considering the above description, the Kohonen net has the following important feature:

- After sufficiently many inputs have been presented the net performs a *topological ordering* of the competitive neurons such that the neighbouring neurons in the layer represent similar

clusters in multidimensional space. In other words, weights will specify cluster centers that sample the input space such that the distribution of the cluster centers approximates the distribution of the input data. This is why the Kohonen net can be used for ordering the scattered points.

THE DYNAMIC KOHONEN NET

Now we discuss how the structure we described above can be extended by the possibility of adding and removing new neurons (for a more detailed description see [5,6]). This procedure will let us to overcome the restriction the number of output neurons being constant.

To grow the structure an error-value is associated to every output neuron. If a neuron is active in a training step, then we increase its error-value by the (Euclidean-) distance of the active neuron and the input point. After n training step we can find the output neuron m with the maximal error-value. Now we insert a new neuron between m and its furthest direct neighbour d. The position of the new neuron c will be:

$$\text{pos}(c) = 0.5\,(\text{pos}(m) + \text{pos}(d))$$

and the neuron will be connected with the surrounding neurons. Its error-value will be the average of the error-values of its neighbours, while the error-variable of the neighbours will decrease.

To shrink the structure we try to find the neurons which are not active for a long time. If for a neuron the number of training steps without being active exceeds a certain limit, then the neuron is removed and the connections are modified to rebuild the original structure.

DESCRIPTION OF THE PROBLEM

Let a set of points (scattered data) be given on the plane. Our purpose is to fit (interpolation or approximation) a curve to them. Thus our first task is to determine the order of the points for the interpolating or approximating methods.

The Kohonen net is used to order the points for the above methods. It is obvious that this order should be adequate in the sense that curves fitted to the points are „well shaped". In this paper we use B-spline curve for approximating the points in the specified order.

ADAPTING THE KOHONEN NET TO THIS PROBLEM

Input vectors:

Let a set $\{x^1,..., x^l\}$ of two dimensional vectors be given. The vectors $x^1,..., x^l$ are called input vectors. The coordinates of these vectors are submitted to the input layer which contains two neurons.

Output map and weights

Let the output vectors $o_1,..., o_m$ be two dimensional vectors with the coordinates (w_{i1}, w_{i2}), where w_{ij} denotes the weights between the output vector i and the jth coordinate of the presented input vector. We use the terms „output vector" and „weights of the output neuron" interchangeably. Let the output map be one dimensional. (See figure 2).

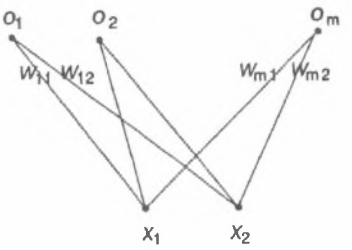

Figure 2

The Kohonen net

After initializing the weights the input vectors are presented sequentially in time.The algorithm determines (using the Euclidean distance) which output vector is the closest to the presented input vector. The coordinates i.e. the weights of this output vector and those of the vectors that are in a certain neighbourhood of the nearest output vector are updated so that these output vectors move closer to the presented input vector. The degree of the updating depends on the gain term and the distance of the output vector and the presented input vector. When the radius (which specifies the neighborhood around an output vector) is large, many output vectors tend towards the presented input. For this reason, initially the output vectors move to places where the density of the input vectors is large, since more input vectors are presented from this areas.

The radius (i.e. the size of the neighborhood) and the gain term is decreasing in time. The latter results in that after enough iterations the locations of the output vectors does not change significantly

(if the gain term is almost zero then the change in the weights is negligible).The gain term should diminish only when the weights are already close to the input vectors.

The output o_k is said to converge to the input x^j if for $\forall \varepsilon > 0$ $\exists t_0$ such that for $\forall\ t > t_0$ $o_k \in N_t$, (x',ε), where $(i = 1,...l)$, $(k = 1,...m)$, $(t = 1,..T)$. A net is said to be *convergent* if for $\forall x^j \exists\ o_k$ such that o_k converges to x^j, where $(i = 1, ..., l)$, $(k = 1, ..., m)$.

In the general case the convergence of the Kohonen net has not been proved yet. Kohonen proved the convergence only in a very simple case when the output is one dimensional and the inputs are the elements of an interval [2].

The probability of the convergence and the number of iterations to the convergence of the net depend on:

1) the initial values of the weights of the output neurons,
2) the method of the presentation of the input vectors,
3) the formulas used to calculate the radius and the gain term,
4) the dynamic characteristic of the Kohonen net.

After reaching a certain level of convergence the input vectors can be ordered according to the order of the output vectors that converge to them. The ordered inputs specify a finite sequence of segments in the plane.

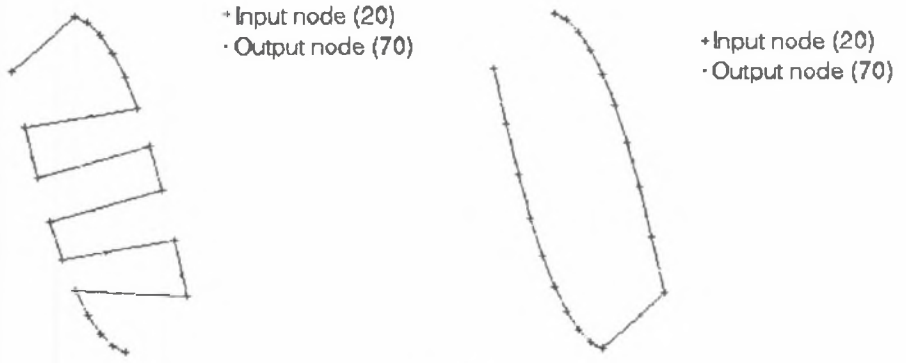

Figure 3

Initializing the weights

1) The simplest initialization of the weights is setting them to small random values. In this case the inputs are far from the outputs.
2) For a better placement of the outputs and for the quicker convergence we can set the weights around the centroid of the inputs (we add small random values to the centroid of the inputs).

3) For input vectors which are bounded in a relatively narrow region and are evenly distributed, the following Initialization can be better than the two above. We select the two inputs that are at the ends of this region and assign them to the weight of the first and last output vectors (o_1 and o_m) on the map. The remaining output vectors are initialized in the previous way or can be placed evenly on the line between the first and last output vectors.

The bast way to initialize the weights depends on the particular distribution of the imput vectors. *Figure 3.* shows the case when different initializations are used for the same input points. In the left picture the weights were initialized with method 3. while in the right picture method 2. was used.

The presentation of the input vectors

Only the input vectors for which there is no output vector that converged to them are submitted to the input layer of the net. The presentation of the input vectors should be carried out according to *figure 4.* In spite of the value 3 of the radius the neighbours of the best matching unit (which is the closest of the presented input) are its direct neighbours. The output neurons are connected by lines. If we consider six output vectors neighbours of the best matching unit then the converged outputs will be moved towards the presented input. Hence the approached inputs can be unapproached again. To avoid this problem the neighbours of the best matching unit should not be converged outputs and outputs beyond converged outputs.

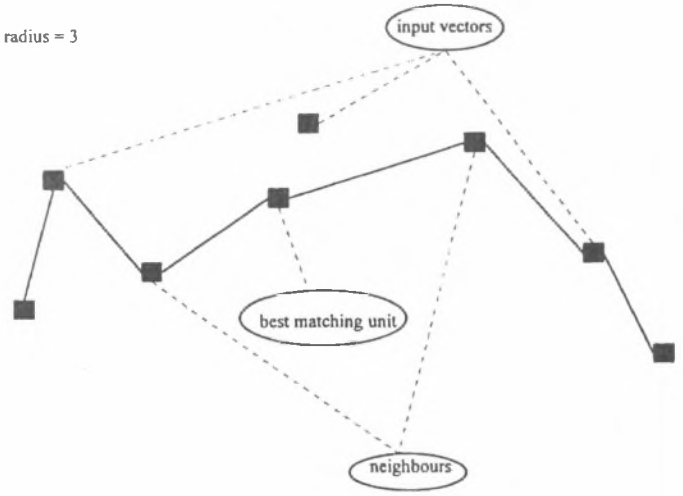

Figure 4

The radius and the gain term

The convergence and the rate of the convergence are influenced by these two factors. Both terms should be decreasing in time. To achieve convergence, the radius has to decrease to zero, otherwise the presented input would attract outputs converging to other inputs.

At the same time the gain term has to decrease in time, otherwise the presented input would attract the outputs which are near to other inputs but relatively far from the presented one (closer to another inputs) and are in the actual neighbourhood. It should be noted that if the gain term decreases rapidly compared to the radius (before the outputs are close enough to the inputs) then the gain term is too small to move the outputs to the presented input (i.e. position of the outputs is not changed virtually). Thus the gain term should decrease so that it remains large enough to have the outputs reach the inputs when the radius diminishes to zero.

In our program the Gaussian function is used for the gain term and for the radius in the following way:

$$gain(t) = \frac{1}{\sqrt{2 \cdot \pi}} \cdot e^{-\frac{t^2}{\frac{m}{(1{,}2 \cdot 10^8 \cdot e^{1300} - 9 \cdot 10^7)}}}$$

$$radius(t) = \left[\frac{m}{2} \cdot e^{-x_l} + 0.9999999 \right], \text{ where } x_l = \begin{cases} \sum_{i=1}^{t} \frac{1}{10} + \frac{t}{10^6}, \text{ if } radius(t-1) < \frac{m}{7} \\ \sum_{i=1}^{t} \frac{1}{10} + \frac{t}{10^4}, \text{ otherwise} \end{cases}$$

The number of output vectors

In the case of the original version of the Kohonen network the number of output vectors should be at least 4 times the number of input vectors. This number is based on experiments. It is obvious that we need at least as many output vectors as input vectors but this number is usually not enough. In the following figure the number of input and output vectors is the same, but one of the outputs is captured between two other output vectors on the left side, therefore only one output remains for two inputs on the right side.

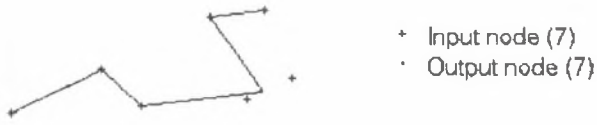

Figure 5

In the case of *dynamic neural network* the above state canont cause problem because in this situation sooner or later an extra neuron will be inserted. Initially the number of the output neurons is the same as the number of the input vectors. At the convergence of the net the number of the output vectors much less than in the above case.

Run results

The above mentioned formulas were found after several runs.These formulas are purely experimental. The table below shows the run results of different versions of the selforganizing net. The first column contains the number of input vectors. Three cases are shown here: the original case where the original Kohonen net is used, the dynamic Kohonen net and the dynamic Kohonen net with the input presentation we mentioned above. The columns of iteration show the average number of iterations that were needed for stable convergence. The columns of number of outputs represent the number of output neurons of the net at the convergence. We accepted the state of the net to be a stable converging state if the radius was small enough and every input vector had at least one output vector in its neighbourhood of radius 0.1. In the original case there were two cases from the 30 runs when the net did not converge, so the probability of the convergence of the net is about 0.98. In the dynamic cases the net always converged. We run the program 10 times for each line in the table.

	The original case		The dynamic version		The dynamic version with special input presentation	
Input vectors	Iterations	Number of outputs	Iterations	Number of outputs	Iterations	Number of outputs
10	1344	40	1112	19	1025	17
20	2332	80	1787	30	1447	28
30	2595	120	1978	47	1789	45

The following figures show the ordering of the input vectors and the approxinating B-spline. There are 20 input vectors.

FUTURE WORK

We plan to generalize the method to three dimensional input points using the Kohonen net. In this case the output map is two dimensional (a grid) and the input vectors and the weights are three dimensional. When the net converges, the grid approximates the input points and an interpolating or approximating surface can be fitted to the input points.

REFERENCES

[1] BOEHM W., FARIN G. and KAHMANN J.: A survey of curve and surface methods in CAGD, CAGD 1 (1984), 1 - 60.

[2] KOHONEN T.: Self-organization and associative memory, Springer Verlag, 1984.

[3] ALDER M., TOGNERI R., LAI E. and ATTIKIOUZEL Y.: Kohonen's algorithm for the numerical parametrisation of manifolds, Pattern Recognition Letters 11 (1990), 313 - 319.

[4] FAUX L.D. and PRATT M.J.: Computational Geometry for Design and Manufacture, Wiley & Sons, NY, 1979.

[5] FRITZKE B.: Unsupervised clustering with growing cell structures, In: Proc. of the IJCNN-91, 1991,Seattle.

[6] FRITZKE B.: Let it grow - self-organizing feature maps with problem dependent cell structure, In: Proc. of the ICANN-91, Elsevier, 1991.

Streszczenie

W pracy zaprezentowano nowe podejście do problemu uporządkowania danych rozproszonych, których modelem geometrycznym jest zbiór punktów płaszczyzny o nieustalonej kolejności łączenia. W celu określenia odpowiedniego uporządkowania posłużono się modelem samoorganizującej się sieci komórek nerwowych Kohonena. Aby zmniejszyć liczbę powtórzeń zastosowano dynamiczną wersję tej sieci. Zmienność liczby elementów sieci w tej wersji pozwala na doskonalsze jej dopasowanie do poszczególnych faz procesu samoorganizacji. Przedstawiono zmodyfikowaną metodę określenia wag początkowych i sposób prezentacji sygnałów wejściowych oraz tworzenia sąsiedztwa. Przytoczono odnośne wzory obliczeniowe. Załączono rezultaty symulacji komputerowej.

Po zakończeniu procesu porządkowania stosuje się standardowe metody aproksymacji i interpolacji. W przedstawionym w pracy przykładzie dokonano aproksymacji za pomocą krzywej typu B-spline.