Modified Kohonen Neural Network for Surface Reconstruction

by MIKLÓS HOFFMANN

Abstract: Kohonen neural networks, also known as self-organizing maps are applied frequently for the handling of the problems of unordered data structures. In this paper a modified algorithm is presented which is more suitable for our surface reconstruction problem, because with the help of the new, continuously diminishing neighborhood the surface will be smoother without decreasing the effectiveness of the approximation.

Introduction

Different types of neural networks have been studied for many years as useful models in artificial intelligence. Special net models were developed for special tasks and these models can be classified in term of the input values and the learning method. As a good overview of the different types of artificial neural networks see e.g. [1]. The Kohonen neural network, developed by T. Kohonen [2], is an unsupervised neural network with continuous-valued input pattern. The unsupervised learning means that no expected-output data is used as a teacher to signal the network to modify himself. The main feature of Kohonen net is the self-organizing ability, which is mainly used for handling unordered data structures.

Neural networks have two essential properties: the topology of the units and the learning algorithm. The Kohonen neural network is a two-layered net. The first layer, called input layer, contains n input neurons, which are totally interconnected to the m neurons of the second, competitive layer. According to the problem – curve or surface reconstruction – this second layer can be one or two dimensional. Weights $w_{ij}, (i = 1...n, j = 1...m)$ are associated with the connections from the i^{th} input neuron to the j^{th} output neuron and are adjusted during the training procedure. Only one single neuron can be active at a time and this neuron represents the cluster which the actual input data set belongs to. Before the training process the associated weights are initialized as small random values. During the training an input data vector $x_1, x_2, ..., x_n$ is presented to the input layer of the network. This layer contains n neurons and the net computes the Euclidean distance d_j between the input vector x and each output node at time t using

$$d_j = \sqrt{\sum_{i=1}^{n}(x_i(t) - w_{ij}(t))^2}.$$

This research was supported by the Hungarian National Foundation for Scientific Research grant No. F019395

Now the output neuron c with the minimum distance is selected and will be the active neuron. Only the weights associated to the connections of this neuron and its neighbors will be adjusted. In a two dimensional output layer the neighborhood $N_c(t)$ is a rectangular area around the active neuron c and it decreases in size over time. Weights for neuron c and all neurons in the neighborhood $N_c(t)$ are updated by the following rule:

$$w_{ij}(t+1) = w_{ij}(t) + \eta(t)(x_i(t) - w_{ij}(t)),$$

where $\eta(t)$ is a time-dependent learning rate also decreases in time and can be chosen as follows:

$$\eta(t) = \frac{1}{\sqrt{2\pi} \cdot exp(\frac{-t^2}{2})}.$$

A well-known example of the behavior of this algorithm can be seen in [3]. Here the coordinates of the points of a two dimensional uniformly distributed area form the input of the neural network, i.e. the input layer has two neurons, while the output layer has 10×10 neurons forming a rectangular grid. Since every output neuron has two connections to the input neurons, the weights of these connections can be considered as two dimensional coordinates of the output neurons. Initially these weights are defined as small random values around the centroid of the input area, while the topology of the connections of the output neurons is defined as a regular rectangular grid. Now presenting the randomly selected input points the grid of the output neurons gradually expands while its point density approximates the distribution of the area. The most important feature of the Kohonen net is that during the training session the topology of the output layer is preserved, so the initially defined grid remains the same.

Surface reconstruction

Our purpose was to reconstruct surfaces from their known points. For the standard surface modelling methods used in CAGD (see e.g. [4]) ordered data structure is required, while the known points form scattered data structure. To solve this problem first the Kohonen network is applied to order the scattered data similar to the example mentioned above, while the standard surface modelling algorithms can be used after this process. These algorithms need rectangular point grid as input, so the Kohonen net has to produce this type of grid. In this context the Kohonen net can be considered as a preprocessing algorithm for producing the appropriate input for surface modelling.

Now the applied neural network has two layers: an input layer with three neurons, since the input points are from E^3 and have three coordinates. The output layer has the rectangular grid topology defined above, while the number of the output neurons depends on the number of the input points. For the detailed discussion of this problem see [5].

The main difference between our problem and the example given above is the number of input points. In the surface reconstruction problem we have finite number of points, actually their number can be very limited, while the randomly selected points of an area can be considered as infinitely many points. Because of this difference some problems arise applying the original Kohonen algorithm, so now a modified algorithm is presented, which is more effective in surface reconstruction.

Continuously decreasing neighborhoods

As we have seen in the example, the Kohonen net starts from a little area of the output space and expands during the training session. For each input point the best matching (i.e. closest) neuron is selected and this neuron and its neighbors have been dragging by the input point towards itself. The next input point is generally far from the previous one, so another part of the output layer will be dragging and finally the whole grid will expand.

The neighborhood decreases in time and normally this neighborhood is a $k \times k$ rectangular area around the active neuron, while after some steps this area decreases to a $(k-1) \times (k-1)$ area etc. To achieve the convergence of the whole net the neighborhood has to decrease to one single neuron. In the rectangular topology of the output layer the radius of the $k \times k$ neighborhood is defined as $(k-1)$. During the training session the radius has to decrease to zero. Unfortunately the process of determining the neighborhood is likely to be application dependent (see [3]). In our case the following term has been used:

$$radius(t) = INT\left(\frac{n}{2 \cdot exp(-\sum_{i=1}^{t} \frac{1}{10} + \frac{t}{10^4})}\right),$$

where n is the maximum possible radius in the rectangular grid.

In the original algorithm the weights of the neurons inside the neighborhood are modified with the same factor while the weights of the neurons outside the neighborhood remain the same as before. As we see, the radius is a natural number which remains the same during several training steps, while suddenly decreases by one. These sudden changes may cause problems in the structure of the output grid, however its topology remains unchanged. This effect can be detected especially when the number of the input points is relatively small.

To avoid this problem a new, continuously decreasing neighborhood will be defined. The purpose of this new type of neighborhood is to decrease the radius slowly, without sudden changes at the border of the neighborhoods. There are different possibilities to define the new radius but the most evident way is to ignore the INT function from the above definition:

$$radius(t) = \frac{n}{2 \cdot exp(-\sum_{i=1}^{t} \frac{1}{10} + \frac{t}{10^4})}.$$

Generally this new radius is not a natural number, but a positive real number and changes after each training step. Originally the neighborhood was defined by the radius as a $(radius + 1) \times (radius + 1)$ rectangular area around the active neuron. For the new radius the following holds:

$$oldradius(t) <= newradius(t) < oldradius(t) + 1,$$

hence the new neighborhood can be defined as the old neighborhood plus a border around it with the width of one neuron. The weights of the neurons inside the old neighborhood will change in the same way as before, but the changes of the weights of the neurons at the border will be multiplied by the fractional part of the new radius.

The modified algorithm

The new algorithm of the training steps of the Kohonen neural network is the following:

Step 1. Initialize the weights $w_{ij}(i = 1...n, j = 1...m)$ as small random values around the centroid of the input space.

Step 2. Present a new input vector.

Step 3. Compute the Euclidean distance of each output neuron to the input as:

$$d_j = \sqrt{\sum_{i=1}^{n}(x_i(t) - w_{ij}(t))^2}.$$

Step 4. Select the active neuron c for which $d_c = min(d_j)$.

Step 5. Compute the neighborhood of the neuron c as:

$$radius(t) = \frac{n}{2 \cdot exp(-\sum_{i=1}^{t} \frac{1}{10} + \frac{t}{10^4})}.$$

Step 6. Update weights of the neuron c and its neighbors by the following equations (the function η(t) is defined above):

$$w_{ij}(t+1) = w_{ij}(t) + \eta(t)(x_i(t) - w_{ij}(t))$$

if the neuron is inside the neighborhood with $radius = INT(radius(t))$, and

$$w_{ij}(t+1) = w_{ij}(t) + FRACT(radius(t)) \cdot (\eta(t)(x_i(t) - w_{ij}(t)))$$

if the neuron is on the border of the neighborhood ($FRACT(x) = x - INT(x)$).

Step 6. Repeat steps by going to **Step 2.** until convergence.

Advantages of the new algorithm

If two rectangular grids with the same topology are produced for the same input points, then one of the grid (and finally the surface determined by this grid) is said to be smoother than the other one if the maximum of the distances between the points of this grid is smaller than in the case of the other grid.

Let a set of input points be given and consider the two grids G_{old} and G_{new} produced by the original and the modified algorithms, respectively. In a certain moment the changing of the positions of the neurons are the same in the two grids, except the border of the neighborhood. All the neurons inside the neighborhood move towards the closest input point, while all the neurons outside the neighborhood keep their positions. The only exceptions are the neurons on the border of the neighborhood in the grid G_{new} produced by the modified algorithm. These neurons also move towards the input point but only with a fractional part of the movement of the neurons inside the neighborhoods. Since in the grid G_{new} the border of the neighborhood will be between the neurons of the neighborhood and the rest of the grid, which evidently yields that the maximum distance in the grid G_{new}

will be equal to or smaller than the maximum distance of the other grid G_{old}. This means that the grid produced by the modified algorithm will be smoother in a sense defined above.

Conclusions

A modification of the well-known Kohonen neural network algorithm has been presented. This modified algorithm gives a better solution for our surface reconstruction problem, since the sudden changes caused by the old neighborhood structure have been ignored. This new, continuously decreasing neighborhood helps to drag the neurons toward the input points in a better way, and finally the output layer will form a smoother rectangular grid over the scattered input data. After this preprocessing steps the standard free-form algorithm can be used for the surface reconstruction.

References

[1] J.A. Freeman, D.M. Skapura: *Neural Networks: Algorithms, Applications and Programming Techniques*, Addison-Wesley, New York, 1991.
[2] T. Kohonen: *Self-Organization and Associative Memory*, Springer Verlag, Berlin, 1984.
[3] R.P. Lippmann: *An Introduction to Computing with Neural Nets*, IEEE ASSP Magazine, April, 1987, pp.18-21.
[4] F. Yamaguchi: *Curves and Surfaces in Computer Aided Geometric Design*, Springer-Verlag, Berlin, 1988.
[5] M. Hoffmann, L. Várady and T. Molnár: *Approximation of Scattered Data by Dynamic Neural Networks*, Journal of Silesian Inst. of Technology, 1996, pp.15-25.

Miklós Hoffmann
Institute of Mathematics and Informatics
Lajos Kossuth University
P.O.Box 12.
H-4010 Debrecen
Hungary