To appear in: Proc. of the IJCNN-91 Seattle

UNSUPERVISED CLUSTERING WITH GROWING CELL STRUCTURES

Bernd FRITZKE

Institut für Mathematische Maschinen und Datenverarbeitung
Lehrstuhl für Programmiersprachen
Universität Erlangen-Nürnberg
Martenstr. 3, D-8520 Erlangen, Germany

Abstract: A Neural Network model is presented which is able to detect clusters of similar patterns. The patterns are n-dimensional real number vectors according to an unknown probability distribution P(X). By evaluating sample vectors according to P(X) a two-dimensional cell structure is gradually built up which models the distribution. Through removal of cells corresponding to areas with low probability density the structure is then split into several disconnected substructures. Each of them identifies one cluster of similar patterns. Not only the number of clusters is determined but also an approximation of the probability distribution inside each clusters. The accuracy of the cluster description is increased linearly with the number of evaluated sample vectors.

1. Introduction

When dealing with large amounts of data it is often useful to perform a classification of the primary data into representative subsets or clusters. The main problem is to find clusters such that data items in one cluster have similar properties (with respect to an appropriate measure) but the properties of data items belonging to different clusters are not similar.

In numerous cases a data item (or pattern) can be expressed as an n-dimensional vector of real numbers. The network model presented here can then determine clusters of similar vectors in an unsupervised manner. This is done by evaluating sample vectors. The information gained by the model consists of the number of detected clusters and of a two-dimensional cell structure for each cluster. This cell structure contains an approximation of the pattern density in the cluster.

2. Growing Cell Structures

The networks introduced in the following are two-dimensional structures consisting of connected cells. The connection scheme is such, that always a number of triangles is formed with one cell in every corner. Every connection stands for a topological neighborhood relation.

These two-dimensional structures will be used to describe clusters of patterns in the n-dimensional vector space of real numbers $\Re^n$. For this reason every cell *c* has attached an n-dimensional **position vector** *pos(c)* indicating its virtual position in $\Re^n$. For convenience we will sometimes denote the position vector of a cell as its **position** and changes of this vector as **movements** of the cell.

Now let P(X) be an unknown probability distribution on $\Re^n$. Although P(X) is not known explicitly it shall be possible to generate arbitrarily many sample vectors according to P(X). By evaluating those sample vectors clusters can be detected as follows:

First the current structure **S** is distributed in $\Re^n$ such that the cell density roughly approximates the probability density of P(X). The cell density thereby is (in the same sense as above) the shorthand for "density of position vectors". This distribution is done by successively generating input vectors according to P(X). For every input vector x the nearest cell is determined. This is the cell **bmu** (best matching unit) satisfying the relation

$$\|\operatorname{pos}(bmu) - x\| \leq \|\operatorname{pos}(c) - x\| \quad \text{for all } (c \in S)$$

Thereby $\| . \|$ denotes the Euclidean Norm.

Then cell bmu and its direct topological neighbors are then moved towards this vector (see Fig. 1). The amount of movement is a fraction $\varepsilon_{\mathbf{bmu}}$ and $\varepsilon_{\mathbf{neighbor}}$ of the total distance, respectively. This step, consisting of input vector generation and cell movement, is repeated for a fixed number $n_{\mathbf{distribution}}$ of times. According to [1] the positions of the cells thereby tend to change towards an equilibrium state. In this state the density of cells approximates a continuous, monotonically increasing function of the probability density of P(X).

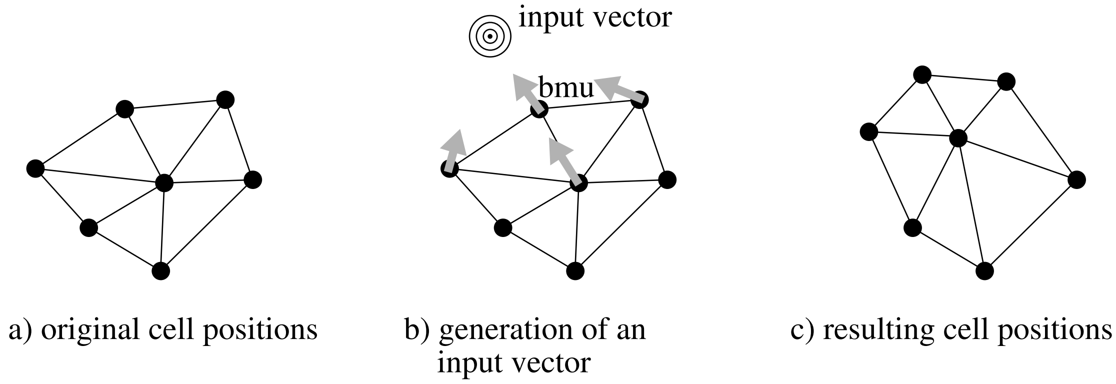

Fig. 1 One step during the redistribution: the bmu and its direct topological neighbors are moved towards the position of the input vector.

During redistribution information can be compiled on where to insert the next cell: Each time when a cell is bmu, we add the distance between this cell and the input vector to an "error" variable attached to the cell. We are then able to determine the amount of error caused by every cell over the time. A high error value indicates, that near the position of the corresponding cell in $\mathfrak{R}^n$ the density of cells is comparatively low. Therefore after the redistribution phase we search for the cell *bs* ("black sheep") with maximum error value. We then determine the farthest direct neighbor of *bs*, for example a cell *f*, and split the edge which connects both.A new cell c_{new} is then inserted halfway between *bs* and *f* in $\mathfrak{R}^n$. Its position vector is initialized as $\operatorname{pos}(c_{new}) = 0.5\,(\operatorname{pos}(bs) + \operatorname{pos}(f))$.

The cell c_{new} is connected with the surrounding cells such, that again a plane structure of triangles is built. Two different cases can occur depending on whether the split edge lies inside the structure S (Fig. 2a) or on the rim of S (Fig. 2b).

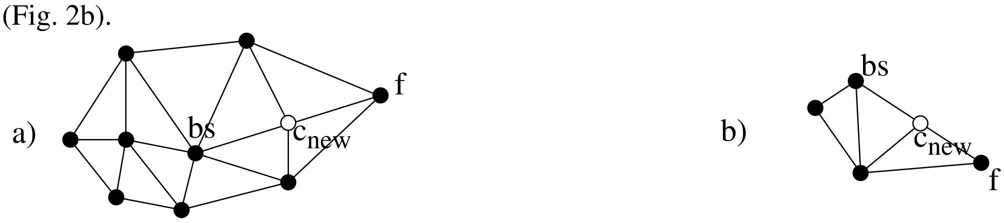

Fig. 2 Insertion of a new cell between *bs* and *f*
a) Either *bs* or *f* (or both) lies inside the structure S. The edge which connects them, is then part of two triangles. This edge is removed and c_{new} is connected to *bs*, *f*, and the opposite corners of both triangles. Thus c_{new} gets four direct neighbors.
b) Both *bs* and *f* lie on the rim of S. In this case the edge connecting them is part of only one triangle. Again this edge is removed. c_{new} is then connected to *bs*, *f*, and the opposite corner of the triangle. So c_{new} gets three direct neighbours.

The error variable of c_{new} is initialized as $err(c_{new}) = \frac{1}{d+1}\sum_{i=1}^{d} err(neighbor_i)$.

Thereby d is the number of direct neighbors of c_{new} (three or four as stated above) and $neighbor_i$ is the i-th of these cells. It is desirable to keep the total error of the involved cells constant. Therefore the error values of the direct neighbors of c_{new} are divided by $(d+1)/d$.

Since the insertion of a new cell diminishes the error of the direct neighbor cells (among them the current *bs*) they will less likely become the “black sheep” the next time. In the long run this will lead to a cell density which approximates the probability density of P(X).

The initial structure we use is the simplest two-dimensional structure, a triangle. Fig. 3 show some steps of the resulting growth process. In this case the probability distribution is two-dimensional. It is uniform in

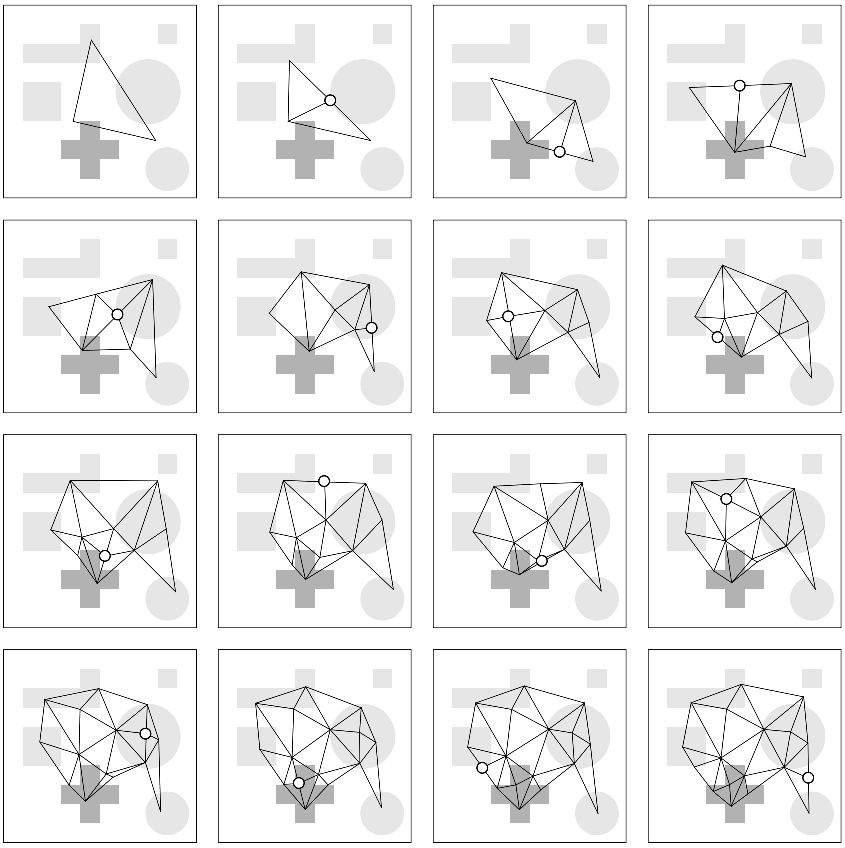

Fig. 3 Cell structure growing according to a structured probability distribution.The cell most recently inserted is marked by a white circle.
(simulation parameters: $n_{\mathrm{distribution}}$ = 25, $\varepsilon_{\mathrm{bmu}}$ = 0.1, $\varepsilon_{\mathrm{neighbor}}$ = 0.005, p_{sure}= 0.9995)

the shaded areas and zero elsewhere. Only in the cross-shaped area the probability density is twice as high as in the other shaded areas. The lines in Fig. 3 denote the connections among the cells. The cells are not shown with the exception of the one most recently inserted. After each $n_{\mathrm{distribution}}$ input vectors a new cell is inserted. This leads to a linear growth of the structure with respect to the number of input vectors.

Since the structure is only extended each cell is directly or indirectly connected to every other. To detect clusters (if they exist) it is necessary to introduce the removal of cells. Clusters of similar vectors are separated by regions with low probability density. Consequently an obvious criterium for a cell to be removed would be, that it has a position in an area of the $\Re^n$, where the probability density is zero. However, since P(X) is not given explicitly, we can only decide this by evaluating sample vectors according to P(X). The general idea is that the longer a cell was not bmu, the more likely it lies in an area of $\Re^n$ where the probability density of P(X) is zero.

Let *n* now be the current number of cells in the structure. Then assume, that P(X) is a uniform distribution in a sub area B of $\Re^n$ and that the cells are uniformly distributed over B. Let c be an arbitrary cell with a position somewhere in B. The event "c is bmu for the current input vector" has the probability $p_{\text{c\_bmu}} = \frac{1}{n}$.

This event does not occur with a probability $p_{\overline{\text{c\_bmu}}} = 1 - p_{\text{c\_bmu}} = 1 - \frac{1}{n}$.

We want to remove cells, which were not bmu for a "sufficiently" large number k_{remove} of successive input vectors. Considering that after every $\boldsymbol{n}_{\textbf{distribution}}$ input vectors a new cell is created, the probability that a cell lying in B has not been bmu for k_{remove} input vectors is

$$p_1 = \left(\frac{n-q}{n}\right)^{n_{distribution}+1}$$

with q defined as $q = \dfrac{k_{remove}}{n_{distribution}}$

Then the probability that each of the currently n cells lying in B was at least one time bmu during the last k_{remove} input vectors is

$$p_2 = (1-p_1)^n$$

To make it improbable that a cell in B is removed we have to choose k_{remove} sufficiently large. If we want to be certain to a degree p_{sure} that no cell in B is removed, then p_2 must exceed p_{sure}. The following relation must be satisfied:

$$\left(1 - \left(\frac{n-q}{n}\right)^{n_{distribution}+1}\right)^n > p_{sure}$$

With the above definition of q this finally leads to

$$k_{remove} > n_{distribution}\text{""}n\left(1 - \left(1 - p_{sure}^{\frac{1}{n}}\right)^{\frac{1}{n_{distribution}+1}}\right)$$

The removal of cells can now easily be realized by numbering the input vectors and storing with every cell the number of the input vectors for which it was bmu most recently. Then k_{remove} is periodically computed for the current number n of cells. If the difference between the current input vector number and the number stored with some cell exceeds k_{remove} this cell and all adjacent edges are removed. In some cases it is necessary to remove further nodes and edges to get again a structure of triangles. An example of the resulting structures is shown in Fig. 4. It is the continuation of the simulation in Fig. 3.

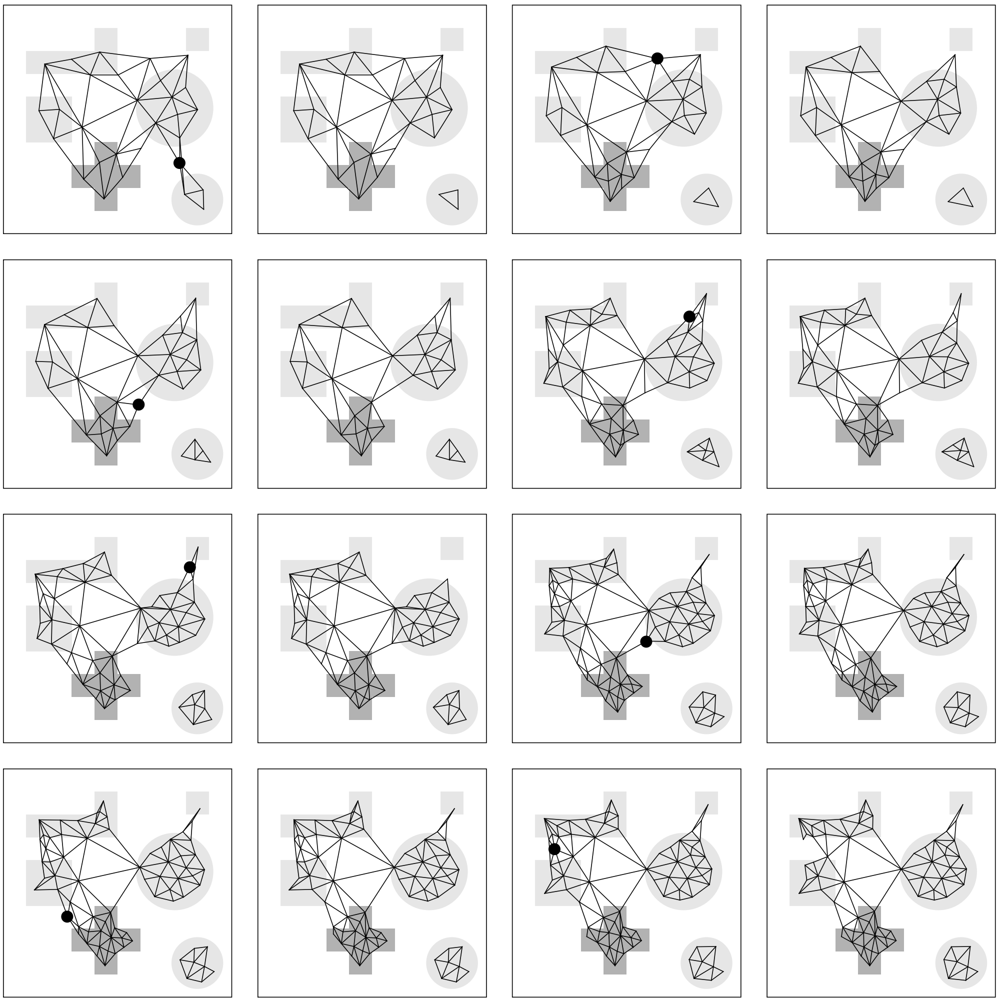

Fig. 4 Sequence of cell structures obtained with insertion and removal of cells. The first 8 removals in the simulation are shown.The cell to be removed next is shown by a black circle. (simulation parameters: $n_{\text{distribution}}$ = 25, ε_{bmu} = 0.1, $\varepsilon_{\text{neighbor}}$ = 0.005, p_{sure}= 0.9995)

The simulation was finished, when the structure consisted of 400 cells (see Fig. 5). In the final result all clusters of the example distribution were found and are rather accurately described by the cell positions. The accuracy can still be increased by continuing the process and thereby letting the structure grow further.

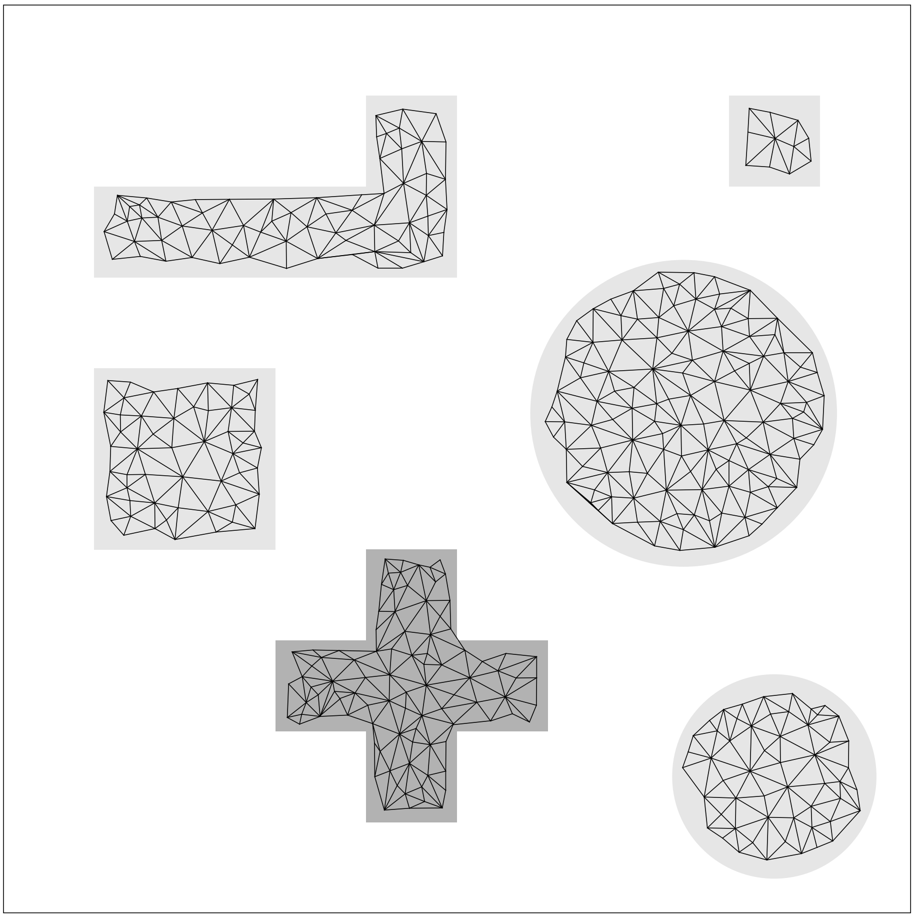

Fig. 5 Simulation result after the structure has reached a size of 400 cells

3. Conclusion And Further Research

We presented a network model which is able to detect clusters of similar patterns according to an unknown probability distribution. The clustering is done unsupervised guided only by sample vectors according to the distribution. Since our model is a generalization of Kohonen's Self-organizing Feature Maps [2], all applications of Feature Maps including robot control and speech recognition might benefit from it.

Currently ongoing studies explore the possibility of constructing three and higher-dimensional cell structures with the same principles. Further investigations are being made to compare our approach directly with the model of Kohonen for various applications.

References

[1] Teuvo Kohonen (1984), *Self-organization and associative memory*, Springer Series in Information Sciences 8, Heidelberg

[2] Bernd Fritzke (1991), *Let it grow - Self-organizing feature maps with problem dependent cell structure*, (submitted to ICANN-91 Helsinki)