CLUSTER ANALYSIS AND NEURAL NETWORK

1 Introduction

The cluster analysis represents a group of methods whose aim is to classify the investigated objects into clusters. The founders of cluster analysis were Tryon, Ward and James. There have been suggested many new algorithms recently. Some methods represent a modification of classical methods of cluster analysis; other ones use advanced methods such as neural networks, e.g. represented by Kohonen self-organizing maps, or genetic algorithms.

The aim of **cluster analysis** is to classify the objects into clusters, especially in such a way that two objects of the same cluster are more similar than the objects of other clusters. The objects can be of various characteristics. It is possible to cluster animals, plants, text documents, economic data etc.

2 Cluster analysis and Kohonen neural network

It is possible to use the neural network with so called unsupervised learning for cluster analysis that is based on evaluation of the difference (distance) of the weighted vector $\mathbf{w}$ of the neural network from the vector of input pattern $\mathbf{x}$ and search of neuron, whose weighted coefficient have the minimum distance of $\mathbf{w}$ from $\mathbf{x}$. This neuron, which won among the neurons of the network, has the right to adjust its weights and the weights of neurons in its surroundings and thus the response on submitted learning pattern to better value. After submitting of a further learning pattern it can win another neuron of the net that can adjust its weights and the weights of neurons in its surroundings and thus to increase better answer etc. The clusters are thus created in the net that in certain places of the network optimally respond to certain symptoms of submitted patterns, as well as unknown patterns. The "map" of patterns is created. This network was presented by Kohonen in 1982 and it is called Kohonen self organizing map. The schema of this network is presented in the Figure 1.

The network contains n distributing nodes that are fed by single components of the vector of input pattern $\mathbf{x}_p = \left(x_{1p}, x_{2p}, \ldots, x_{ip}, \ldots, x_{np}\right)^T$ and N efficient neurons distributed on imaginary surface (one-layer network), for example on the surface of specific configuration such as rectangular structure, where the efficient neurons are found in individual crossing lines of the rectangular structure (there are used other structures, such as with glory in efficient neuron using other structure, such as hexagon). The neurons have among themselves the bindings, so that it is possible to define the surroundings of each neuron. Further, it is possible to think over lateral activity of neuron on the neighbouring neurons according to the rule: the nearest surroundings are represented by exciting bindings and further surroundings by inhibitive bindings. The action of the i-th component x_i on the j-th neuron is done by weight coefficient w_{ji} (real value). The weight vector of the j-th neuron is $\mathbf{w}_j = \left(w_{j1}, w_{j2}, \ldots, w_{jn}\right)^T$.

The input vector (for the p-th input pattern) has the form
$\mathbf{x}_p = \left(x_{1p}, x_{2p}, \ldots, x_{np}\right)^T$.

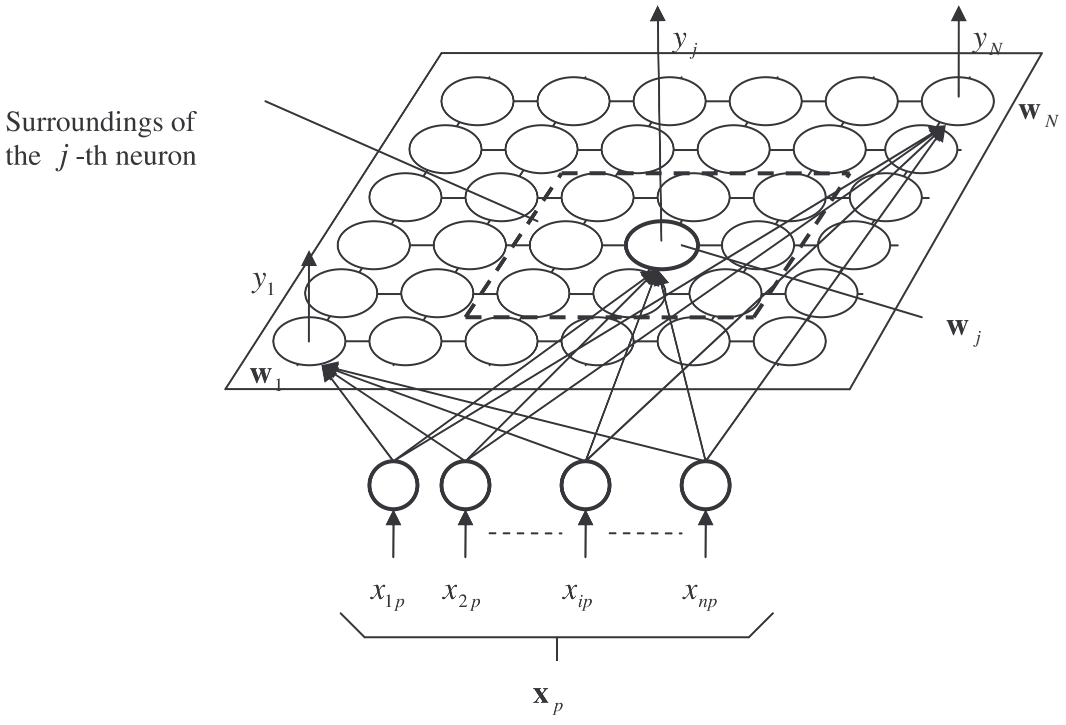

Figure 1: Sample schema of the Kohonen network

It is calculated for each neuron its distance D_j (of the vector $\mathbf{w}_j$ from the vector $\mathbf{x}_p$) as an Euclidean distance

$$D_j(\mathbf{w}_j, \mathbf{x}_p) = |\mathbf{w}_j - \mathbf{x}_p|$$

or as a spherical distance

$$D_j(\mathbf{w}_j, \mathbf{x}_p) = 1 - \mathbf{w}_j * \mathbf{x}_p .$$

For the evaluation of D_j is used the formula

$$D_j = \sum_{i=1}^{n} (x_{ip} - w_{ji})^2 \text{ , where } j = 1, \ldots, N .$$

The competition of neuron consists in the fact that the neuron is found, that has the weighted coefficients with the smallest distance from values of relevant components of the vector $\mathbf{x}_p$. This neuron is considered to be a winner and it obtains the right of the biggest response $y_{jC} = 1$ and adjusts its weights according to the formula

$$\mathbf{w}_j^{new} = \mathbf{w}_j^{old} + \alpha(\mathbf{x}_p - \mathbf{w}_j^{old})y_j$$

where α is so called learning constant for which applies $0 < \alpha < 1$.

For the winning neuron it applies $y_j = y_{jc} = 1$, then

$$\mathbf{w}_j^{new} = \mathbf{w}_j^{old}(1 - \alpha) + \alpha \mathbf{x}_p$$

and for the loosing neurons it applies $y_j = 0$ and the weight coefficients are not changed with the exception of neurons in a defined surroundings of the winning neuron. Their weights are corrected in a specific way (for example by the same way like for winning neuron).

The surroundings NE_c of the winning neuron is a set of neurons inside the bounded area (for example of a square or (because the circle is an ideal area) the hexagon is used that approximate the circle) around the winning neuron. See Figure 2 or Figure 3, respectively.

The radius of surroundings R_C is given by the number of neurons in the surroundings on one side from the winning neuron (for example the radius of square surroundings can be $R_C = 2$ and for hexagon $R_C = 1$). The index C represents the winning neuron.

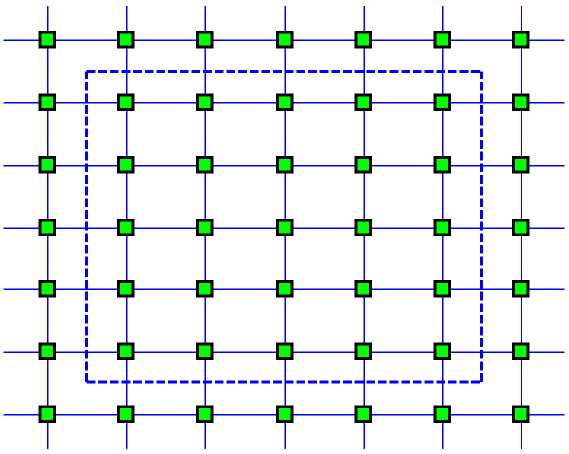
Figure 2: Square surroundings of the winning neuron, $R_C = 2$

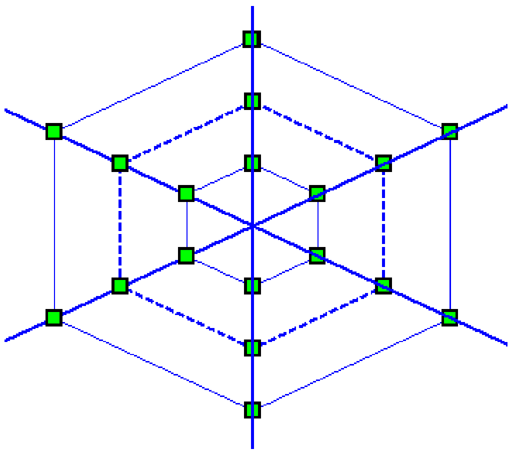
Figure 3: Hexagon surroundings of winning neuron, $R_C = 1$

Probably some other neuron wins when the next learning pattern is set to the input. The process of adjusting of output level of signal, the values of the weight coefficients and surroundings of the new winning neuron is analogue to the first example. The weights of individual neurons are adjusted by consecutive feed of all learning patterns in such a way that certain types of input patterns initiate the neurons in certain places of the network. The neurons are clustered on certain patterns of signals in certain places of network, it is called the map of patterns. When the whole set of learning patterns is used for learning, the network is learned to respond correctly on searched patterns and to distinguish them. For a finer adjustment of the weights it is used the principle of narrowing of the surroundings as well as the downsizing of the value of learning constant. The network is learned in case when further pattern do not change the weight coefficients.

The algorithm of learning of the neural network is as follows:

1) Set up of the number of input variables n according to the structure of input pattern and determination of a number N of efficient neurons according to condition: $L \le N/(4\log N)$ where L is the number of patterns to be recognized by net.

2) Set up of initial values that are selected as random values in the range $\langle -0.5; 0.5 \rangle$.

3) Set up of the shape and radius of initial environs of winning neuron $NE(t_0)$, at first it is greater, major, then it is decreased step by step to be $R_C(t) = 1$.

4) The use of first learning pattern $\mathbf{x}_1$ and foundation of the winning neuron is done in such a way, that the Euclidean distance of its weighted vector $\mathbf{w}_j$ from $\mathbf{x}_1$ is the smallest. The calculation is done according to the formula

$$D_j = \sum_{i=1}^{n} \left(x_i(t) - w_{ji}(t)\right)^2 \rightarrow \min \text{ for } j = 1, \ldots, N$$

The ideal case is the situation when $\min_j D_j \rightarrow 0$.

5) The adaptation of weighted coefficients is done according to the formula

$$w_{ji}(t+1) = w_{ji}(t) + \alpha(t)\left(x_i(t) - w_{ji}(t)\right), \text{ for } j \in NE_C(t)$$

and

$$w_{ji}(t+1) = w_{ji}(t), \text{ for } j \notin NE_C(t).$$

The teaching is done so long, until the weighted coefficients do not change if random pattern from the collection of learning patterns is used. For further details see **Chyba! Nenalezen zdroj odkazů.**, **Chyba! Nenalezen zdroj odkazů.**, **Chyba! Nenalezen zdroj odkazů.** and **Chyba! Nenalezen zdroj odkazů.**.

3 Case studies

The above discussed theoretical results will be applied in this chapter for solving practical problems. We present here two case studies devoted to searching of optimal location of distribution centers and to clustering of realties, respectively. We use software Matlab 7.1 and its Neural Network toolbox to prepare software applications that can be used to solve these types of problems.

3.1 Optimal locations of distribution centers

The input data are represented by coordinates $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_k$ that characterize the objects. It is possible to define any number of clusters. The aim is to minimize the sum of squares of distances between the objects (customers) and centroids (distribution centres). The coordinates of centroids $\mathbf{x}_{c1}, \mathbf{x}_{c2}, ..., \mathbf{x}_{ck}$ are changed. The calculation assigns the objects to their centroids. The whole learning process is repeated until the specified number of learning epochs is reached. The process of learning ensures that the defined coordinates $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_k$ of objects and assigned coordinates $\mathbf{x}_{c1}, \mathbf{x}_{c2}, ..., \mathbf{x}_{ck}$ of clusters have small distances if adequate value of Kohonen parameter is used (in our case 0.1).

Input data for a three-dimensional task are represented by 14 objects with x_1, x_2 and x_3 coordinates (see Table 1). The input data are in an MS Excel format file *ShNN3.xls*.

Table 1: COORDINATES OF OBJECTS (SHNN3.XLS)

Object	*Coordinates of objects*		
Number	x_1	x_2	x_3
1	0,00	0,16	0,16
2	0,34	0,00	0,00
3	0,39	0,26	0,26
4	0,35	0,49	0,49
5	0,50	0,36	0,36
6	0,46	0,48	0,48
7	0,51	0,83	0,83
8	0,52	0,99	0,52
9	0,66	0,36	0,66
10	0,81	0,61	0,81
11	0,64	0,95	0,64
12	0,85	1,00	0,85
13	0,93	0,98	0,93
14	1,00	0,56	1,00

The program is started by command *NNNN1* in MATLAB (for source code see File 1). The user may choose an arbitrary number of distribution centers and number of learning epochs (e.g. 3 clusters and 40 learning epochs).

```
clear all;
P=(xlsread('ShNN3','Locations'))';
num=input('Set the number of distribution centers:');
epochs=input('Set the number of epochs:');
net=newc([0 1; 0 1; 0 1],num,0.1);
net.trainParam.epochs = 1;
for k=1:epochs
    net = train(net,P);
    w = net.IW{1};
    stem3(w(:,1),w(:,2),w(:,3),'sr','MarkerFaceColor','b','MarkerSize',10)
    grid on;
    hold on
    Q=P';
    for i=1:size(Q,1)
        for j=1:(size(w,1))
            distances(j)=sqrt((Q(i,1)-w(j,1))^2+(Q(i,2)-w(j,2))^2+(Q(i,3)-
            w(j,3))^2);
        end
        [min_distance(i),assignment(i)]=min(distances);
    end
    for i=1:size(Q,1)
      stem3(Q(i,1),Q(i,2),Q(i,3),'sr','MarkerFaceColor',[assignment(i)/num,
      assignment(i)/num,assignment(i)/num],'MarkerSize',10)
      xlabel('x');ylabel('y');zlabel('z');
    end
    figure(gcf)
    hold off
end
total_distance=sum(min_distance)
Q
w
assignment
```

File 1: NNNN1.m

During the calculation the dynamical three-dimensional graph is presented (see Figure 4).

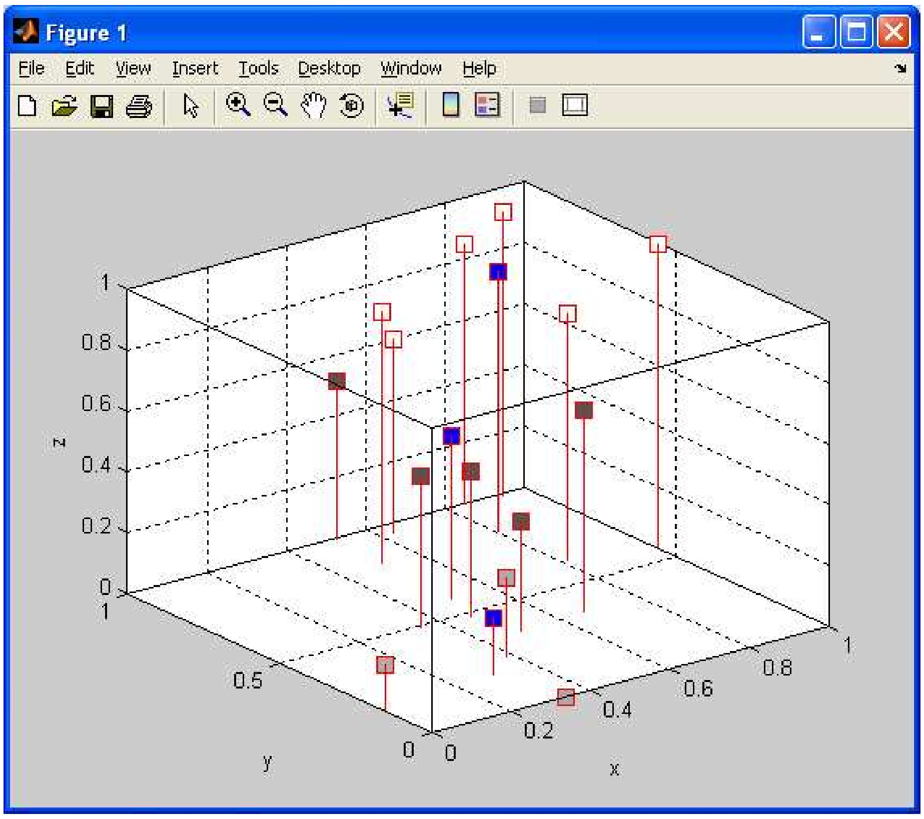

Figure 4: Three-dimensional graph – three clusters

When the calculation is terminated the input parameters and results of calculation are displayed on the screen. The results are presented by the total distance of the objects from corresponding centroids, coordinates of centroids (weights **w**) and assignment of objects to clusters:

```
total_distance =

   3.5773

w =

   0.4947    0.5792    0.5366
```

```
    0.2982    0.1888    0.1888
    0.8056    0.8335    0.8502
assignment =
     2     2     2     1     1     1     3     1     1     3     3     3     3     3
```

3.2 Clustering of realties

The input data of this problem are represented by the values of individual criteria $\mathbf{x}_1$, $\mathbf{x}_2,\dots,\mathbf{x}_k$ that characterize the objects (realties) from various points of view. The aim is to divide the realties into specified number of groups (clusters) such that the realties in the same group have similar characteristics that are different enough from the properties of the realties in other clusters. Sample input data are given in the Table 2 for 48 realties and 10 criteria describing e.g. the locality, area, equipment and price of the particular realty, etc. The input data are loaded from an MS Excel format file *ShNNem.xls*.

TABLE 2: CHARACTERISTICS (CRITERIA) OF INDIVIDUAL OBJECTS (REALTIES) FOR CLUSTERING

Object	Locality	Type	Pool	Rooms	Child. room	Furniture	Storey	Area	Equipment	Price
1	21	3	1	5	1	0	0	242	1	600
2	12	11	0	4	1	1	0	1043	2	1650
3	9	3	1	2	1	2	7	113	2	550
4	10	11	0	5	1	0	2	929	2	1900
...	...	...	...	...	...	...	...	...	...	...
48	27	4	0	3	1	1	1	162	2	445

The source code of the application prepared with use of the software Matlab and its Neural Network toolbox is shown in File 2. The program is started by command *NNNN1Nem* in MATLAB We use normed values of the criteria such that each criterion has the same weight. It is possible to choose an arbitrary number of clusters, number of learning epochs and an arbitrary value of the Kohonen parameter (e.g. 3 clusters, 40 epochs and Kohonen parameter equal to 0.01).

```
clear all;
tic
P=(xlsread('ShNNem','Criteria'))';
for i=1:size(P,1)
    max(P(i,:));
    P(i,:)=P(i,:)/max(P(i,:));
end
scrsz = get(0,'ScreenSize');
num=input('Enter the number of clusters:');
epochs=input('Enter the number of epochs:');
kohonen=input('Enter the value of the Kohonen parameter:');
palette=([1 0 0; 0 1 0; 0 0 1; 1 1 1; 0 0 0; 0 1 1; 1 1 0; 1 0 1; 0.5 0.5
0.5; 1 0 0.5; 0.5 0.5 0; 0 1 0.5; 0.5 0 0.5; 0 0.5 1; 0 0.5 0.5; 0.5 0 1;
0.5 1 0; 1 0.5 0]);
number=floor(num/18);
if num>17
    for i=0:(number-1)
        for j=1:18
            for k=1:3
                colours(i*18+j,k)=palette(j,k);
            end
        end
    end
end
if (num-18*number)~=0
    for i=1:(num-18*number)
        for k=1:3
            colours(number*18+i,k)=palette(i,k);
        end
    end
end
net=newc([min(P(1,:)) max(P(1,:)); min(P(2,:)) max(P(2,:));min(P(3,:))
max(P(3,:)); ...
min(P(4,:)) max(P(4,:)); min(P(5,:)) max(P(5,:));min(P(6,:))
max(P(6,:));min(P(7,:)) ...
max(P(7,:));min(P(8,:)) max(P(8,:));min(P(9,:)) max(P(9,:));min(P(10,:))
max(P(10,:))],num,kohonen);
net.trainParam.epochs = 1;
for k=1:epochs
net = train(net,P);
w = net.IW{1};
subplot(2,2,1);
for j=1:size(w,1)

stem3(w(j,1),w(j,8),w(j,10),'ob','MarkerFaceColor',[colours(j,1),colours(j,
2),colours(j,3)],'MarkerSize',15,'LineWidth',2);
    hold on
end
title('Cluster analysis for criteria 1, 8 and 10 (x, y and z axes,
respectively)');
set(gcf,'Name','Neural Network: cluster analysis','Position',[scrsz(3)/17
scrsz(4)/15 scrsz(3)/15*14 scrsz(4)/15*12]);
grid on;
Q=P';
count=zeros(1,num);
for i=1:size(Q,1)
    for j=1:(size(w,1))
        for l=1:size(Q,2)
            distance(l)=(Q(i,l)-w(j,l))^2;
        end
        distances(j)=sqrt(sum(distance));
    end
    [min_distance(i),assignment(i)]=min(distances);
    count(assignment(i))=count(assignment(i))+1;
end
for i=1:size(Q,1)
stem3(Q(i,1),Q(i,8),Q(i,10),'sr','MarkerFaceColor',[colours(assignment(i),1
),colours(assignment(i),2),colours(assignment(i),3)],'MarkerSize',10);
xlabel('Criterium 1');ylabel('Criterium 8');zlabel('Criterium 10');
end
figure(gcf)
hold off
subplot(2,2,2);
for i=1:size(w,1)
plot(w(i,:),'Color',[colours(i,1),colours(i,2),colours(i,3)],'LineWidth',
2);
    hold on;
end
title('Values of criteria for centroids of individual clusters');
xlabel('Criterium');ylabel('Value');
figure(gcf)
hold off
subplot(2,2,3)
count=count/size(Q,1);
for i=1:num
stem(i,count(i),'LineWidth',2,'MarkerFaceColor',[colours(i,1),colours(i,2),
colours(i,3)],'MarkerSize',20);
    hold on
end
title('Cluster weights');
xlabel('Cluster');ylabel('Cluster Weight');
figure(gcf)
hold off
```

```
total_distance(k)=sum(min_distance);
subplot(2,2,4);
plot(total_distance,'LineWidth',2);
xlabel('epoch');
ylabel('total distance');
xlim([1 epochs]);
title('Value of the total distance of individual objects from corresponding
centroids');
figure(gcf)
hold off
end
toc
time=toc
total_distance=sum(min_distance)
w
assignment
```

File 2: NNNN1Nem.m

During the calculation a dynamical graph with the most important features of the results is presented (see Figure 5). The first subplot displays all the objects (realties) and corresponding centroids with respect to chosen (most significant) criteria 1, 8 and 10 (locality, area and price of the particular realty, respectively). The second subplot shows the values of the individual criteria for different centroids. The third subplot represents the weights of the clusters (i.e. the number of objects assigned to each centroid). The last subplot indicates the evolution of the total distance of individual objects from corresponding centroids.

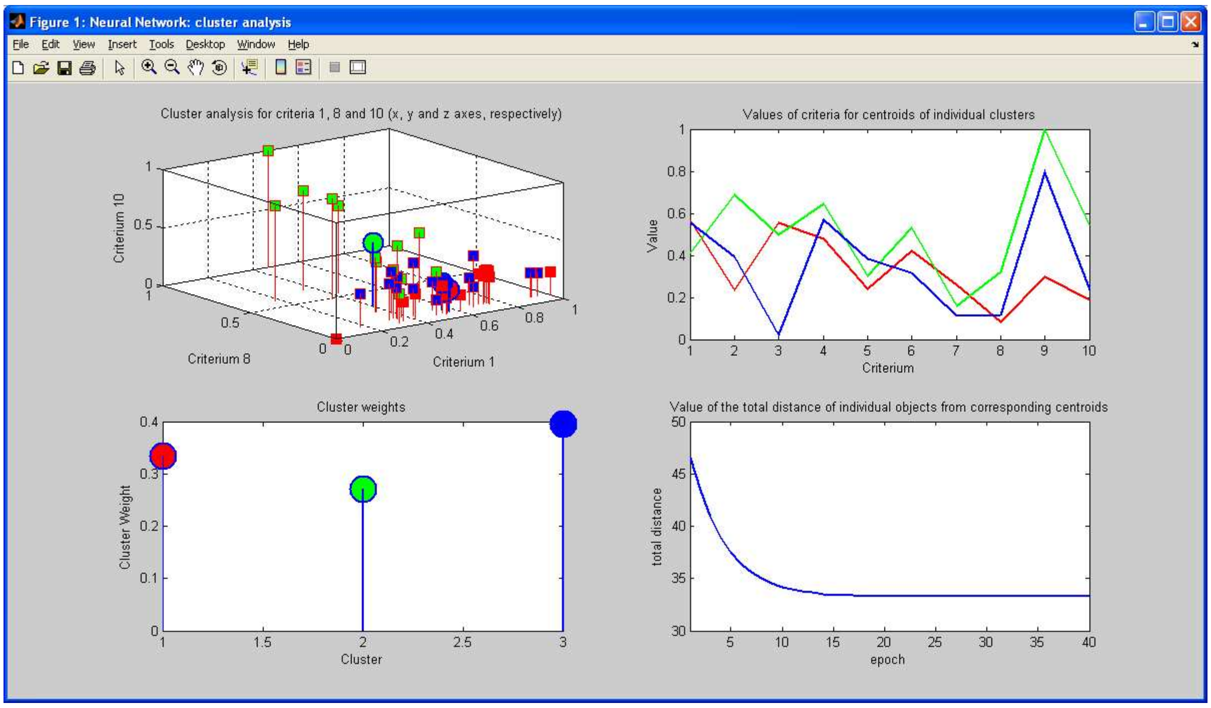
Figure 1: Neural Network: cluster analysis
File Edit View Insert Tools Desktop Window Help
Cluster analysis for criteria 1, 8 and 10 (x, y and z axes, respectively)
Criterium 10
Criterium 8
Criterium 1
Values of criteria for centroids of individual clusters
Value
Criterium
Cluster weights
Cluster Weight
Cluster
Value of the total distance of individual objects from corresponding centroids
total distance
epoch

Figure 5: Cluster analysis of realties (with use of Kohonen neural network)

Finally, the following results are being displayed (the time of computation, total distance of objects from centroids, characteristics (criteria) of different centroids and the assignment of individual realties (objects) to the corresponding centroids – clusters):

```
time =

  49.1422

total_distance =

  33.3494

w =
 Columns 1 through 9
  0.5628    0.2369    0.5529    0.4797    0.2395    0.4228    0.2590    0.0822    0.2958
  0.4098    0.6864    0.4985    0.6434    0.3012    0.5322    0.1600    0.3214    0.9978
  0.5564    0.3907    0.0210    0.5676    0.3848    0.3188    0.1096    0.1143    0.7944
 Column 10
  0.1878
  0.5394
  0.2353
assignment =
 Columns 1 through 16
   1   2   2   2   2   2   3   1   3   3   3   2   1   1   3   3
 Columns 17 through 32
   3   1   1   2   3   2   1   2   3   1   3   3   1   3   2   3
 Columns 33 through 48
   3   2   1   2   3   1   3   2   3   1   1   3   1   1   1   3
```

The sample calculation was prepared with use of three clusters, 40 learning epochs and the value of the Kohonen parameter equal to 0.01 in order to obtain neat results. However, from the obtained results it follows that the neural networks are suitable mainly for solving the cluster analysis problems with higher number of clusters (and a corresponding higher value of the Kohonen parameter). For this number of clusters the genetic algorithms yield better results.

4 Conclusion

We described the use of neural networks for cluster analysis. We dealt specifically with the Kohonen self organizing maps. The cluster analysis can be used in various branches. One of them is economy and business. We can mention for example the search of best location of a market, bank or a facility. We discussed two specific applications prepared with use of the software Matlab and its Neural Network toolbox which concerned optimal location of distribution centers and clustering of realties.

There is a significant dependence of the quality of the results on the chosen value of the Kohonen parameter. For low values of this parameter there are better convergence properties. However, the resulting value is sometimes worse than for computation with higher value of the Kohonen parameter when the computation is not so stabile. Further, the properties of the results are strongly influenced by the specified number of clusters. To obtain better results for higher number of clusters it is better to use higher value of the Kohonen parameter. Finally, if we study the results for higher number of clusters obtained by use of neural networks, they are better than results obtained with use of genetic algorithms **Chyba! Nenalezen zdroj odkazů.**. However, for low number of clusters it is better to use genetic algorithms.