CSE-E4810 Machine Learning and Neural Networks (5 cr)

Lecture 10: Self-Organizing Maps and Learning Vector Quantization

Prof. Juha Karhunen

https://mycourses.aalto.fi/course/view.php?id=13086

Introduction

- This lecture deals with self-organizing neural networks.
- They are based on competitive learning, where the output neurons compete among themselves to determine a winner.
- If the competition process is controlled properly, the output neurons specialize to represent their own region of the input data vectors.
- The self-organizing map (SOM) uses unsupervised learning.
- The learning vector quantization (LVQ) refines the results provided by SOM using supervised learning in classification problems.
- SOM and LVQ are discussed in Du's and Swamy's book in Chapter 8, "Clustering I: Basic Clustering Models and Algorithms".
- However, SOM is more a nonlinear mapping and visualization method than a clustering method.

- The neural gas method discussed in Section 8.5 in Du's and Swamy's book is a modification of SOM which suits better to clustering.
- Chapter 8 in Du's and Swamy's book serves as a background material for this lecture.
- This lecture is based on the earlier used Ham's and Kostanic's as well as Haykin's books.
- And on examples and figures produced in our former laboratory.
- Our notations are somewhat different than in Du's and Swamy's book, but the correspondence between the notations is easy to understand.

The self-organizing map (SOM)

- The self-organizing map (SOM) was developed by Prof. Teuvo Kohonen in our former laboratory in early 1980's.
- It is one of the most widely used neural network methods with many real-world applications.
- In the SOM, there are no hidden layers.
- The input vectors are mapped nonlinearly onto the output layer.
- The neurons in the output layer are usually arranged into a two-dimensional rectangular grid.
- SOM aims at mapping the input vectors onto the output layer so that their topology is preserved as well as possible.
- Because the input vectors are often high-dimensional, some distortion is usually inevitable.

- But each neuron in the output layer specializes to present some region of the input data.
- Regions that are close to each other in the input space map onto neurons that close to each other in the final SOM map.
- SOM is especially useful in visualizing high-dimensional data.
- In SOM, there are no lateral connections between the neurons in the output layer.
- But the neurons close to each other interact via a neighborhood function in the learning algorithm.
- This interaction between the neurons in the output layer is critical in the training of SOM.

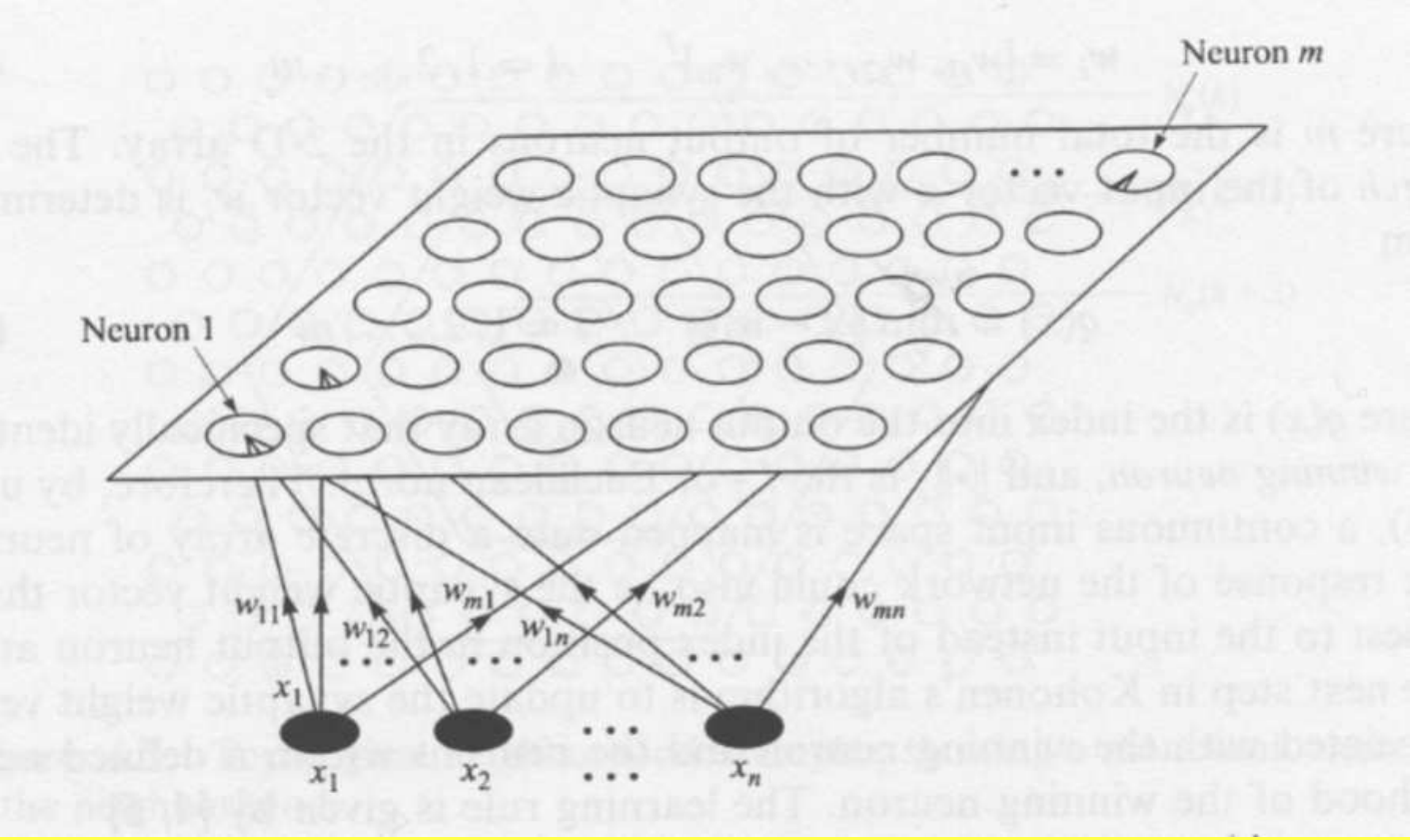


Figure 1: The conventional feature-mapping architecture of the self-organizing map.

Training algorithm for SOM

- Figure 1 presents the conventional feature-mapping architecture of the self-organizing map.
- The figure shows an input vector

$$\mathbf{x} = [x_1, x_2, \ldots, x_n]^T \tag{1}$$

- Its components x_i are depicted as black ellipses at the bottom of the figure.
- The output neurons are arranged into the two-dimensional rectangular array shown above the input vector in Figure 1.
- All the components of the input vector are connected to all the m output neurons.
- For clarity, only a few of the connections are shown.

- The synaptic weight vector of neuron i in the two-dimensional array of output neurons is

$$\mathbf{w}_i = [w_{i1}, w_{i2}, \ldots, w_{in}]^T, \qquad i = 1, 2, \ldots, m \tag{2}$$

- There m is the total number of output neurons.
- The neurons in the output layer are indexed according to some convention.
- Note that even though the output neurons are arranged into two-dimensional array, their weight vectors are n-dimensional.
- Having thus the same dimension as the input vectors $\mathbf{x}$.
- In the SOM learning algorithm, the Euclidean distances $\| \mathbf{x} - \mathbf{w}_i \|_2$ of the current input vector $\mathbf{x}$ to all the weight vectors $\mathbf{w}_i$, $i = 1, 2, \ldots, m$, are first computed.

- The winning neuron is defined as the output neuron whose weight vector $\mathbf{w}_q$ has the shortest Euclidean distance to $\mathbf{x}$.
- That is, the index of the winning neuron

$$q(\mathbf{x}) = \arg\min_i \| \mathbf{x} - \mathbf{w}_i \|_2, \qquad i = 1, 2, \ldots, m \tag{3}$$

- Of course, the winning neuron $q(\mathbf{x})$ depends on the current input vector $\mathbf{x}$.
- Next, the weight vectors of the winning neuron and the neurons in its pre-defined neighborhood N_q are updated.
- The SOM update rule is

$$\mathbf{w}_i(k+1) = \mathbf{w}_i(k) + \eta_{qi}(k)[\mathbf{x}(k) - \mathbf{w}_i(k)] \tag{4}$$

- The neurons outside the neighborhood N_q are not updated: $\eta_{qi}(k) = 0$ for them.

- While for neurons i belonging to the neighborhood N_q of the winning neuron q,

$$\eta_{qi}(k) = \mu(k) \tag{5}$$

- There the learning parameter $\mu(k)$, where $0 < \mu(k) < 1$, decreases with time (iteration number k) according to some strategy.
- The effect of the learning rule (4) is to pull the weights vectors of the winning neuron $\mathbf{w}_q$ and neurons in its neighborhood N_q towards the input vector $\mathbf{x}$.
- The neighborhood set $N_q(k)$ is also a function of iteration number (time) k.

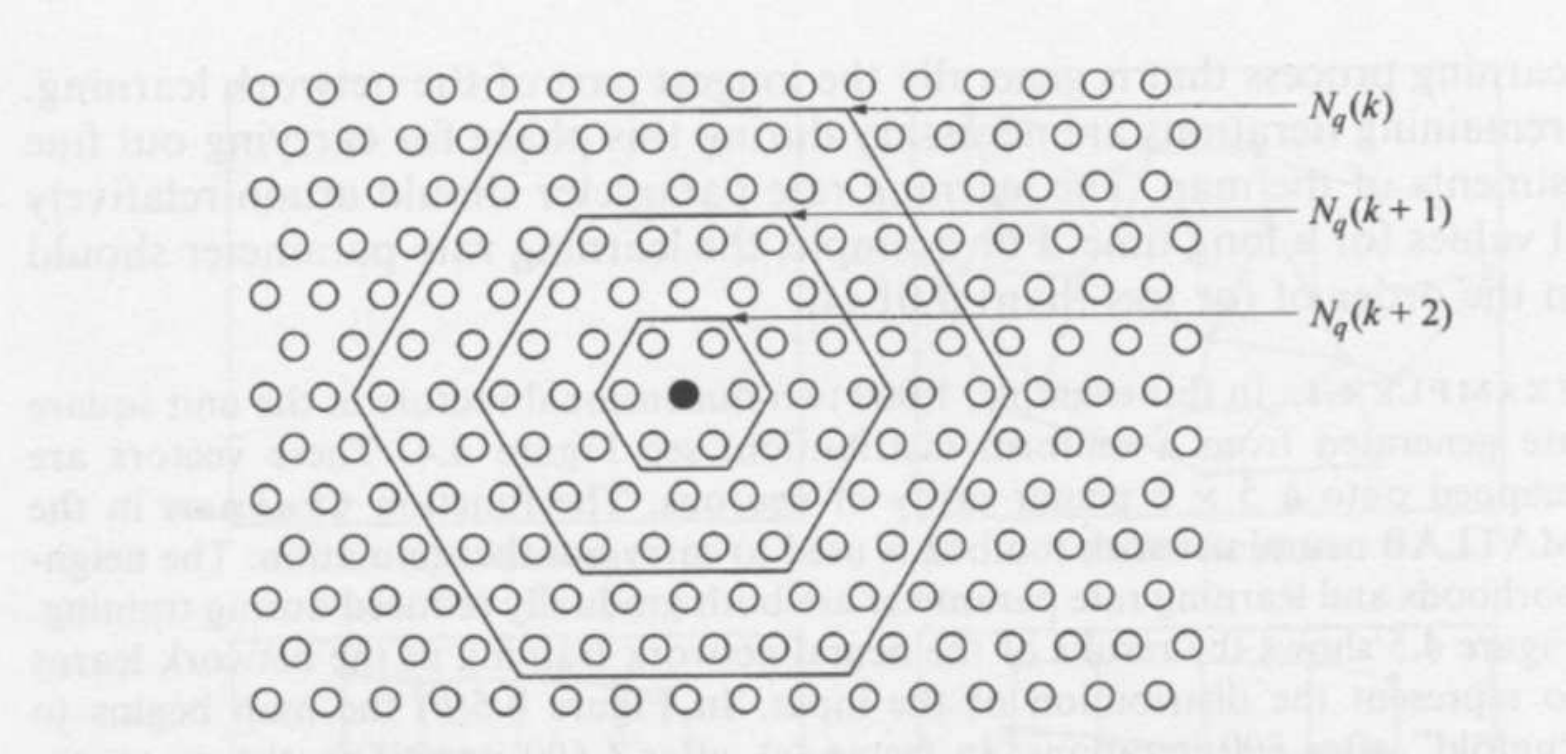


Figure 2: Topological neighborhood examples, showing a monotonic decrease in the neighborhood.

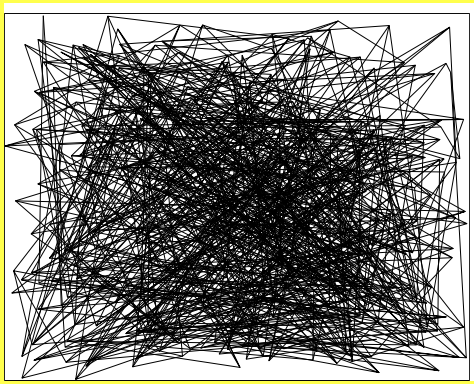
- The neighborhood $N_q(k)$ should be relatively wide in beginning of training and then shrink slowly with time.
- This is illustrated in Figure 2.
- However, the width of the neighborhood is usually kept constant during one epoch at least.
- Recall that an epoch in learning means using all the training samples once.
- The simple neighborhood function $\eta_{qi}(k)$ in (5) is constant $\mu(k)$ for the winning neuron and all the neurons belonging to its neighborhood $N_q(k)$ at each iteration.
- In practice, it is usually better to update the winning neuron more than its neighbors.
- This is achieved by using a Gaussian (bell-shaped) neighborhood function.

- Denote by $\mathbf{r}_q$ and $\mathbf{r}_i$ the coordinate vectors of the winning neuron q and neuron i in its neighborhood in the two-dimensional array of output neurons.
- The Gaussian neighboorhood function is defined typically by

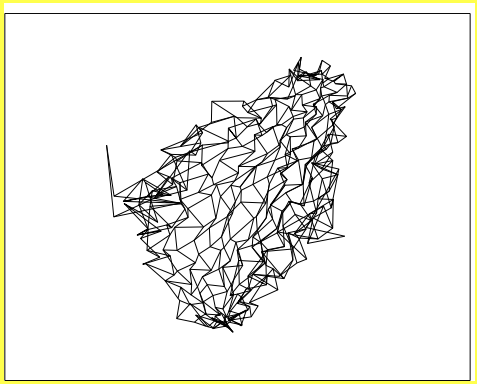
$$\eta_{qi}(k) = \eta_0(k) \exp(- \parallel \mathbf{r}_i - \mathbf{r}_q \parallel^2 / \sigma^2(k)) \tag{6}$$

- There both the height $\eta_0(k)$ and the width parameter $\sigma(k)$ of the Gaussian function (6) decrease with time (iteration number) k.
- Using the function (6), the neuron i in the neighborhood N_q is updated the less the farther away it is from the winning neuron q.
- The resulting SOM learning algorithm is summarized as Algorithm 8.1 in Du's and Swamy's book.
- The weight vectors are often initialized to some random values.

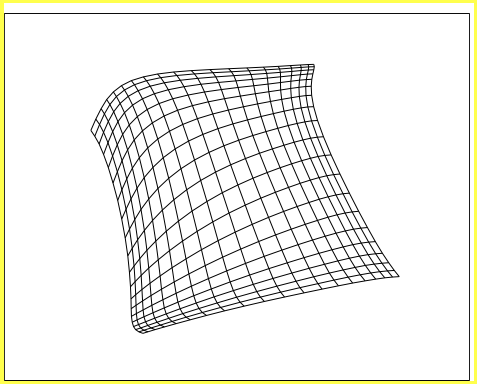
- The SOM algorithm is stopped after some total specified number of iterations or when the weight vectors do not change markedly.
- During the SOM learning process there are two separate but related phases.
- Namely the ordering phase and the convergence phase.
- During the initial ordering phase , the topological ordering of the weight vectors is carried out.
- The learning rate parameter $\mu(k)$ should first be set close to unity.
- And then gradually decreased but not allowed to go below 0.1.
- In the second convergence phase, the self-organizing map is fine tuned.
- The learning rate parameter $\mu(k)$ should then have values on the order of 0.01 for a long time.



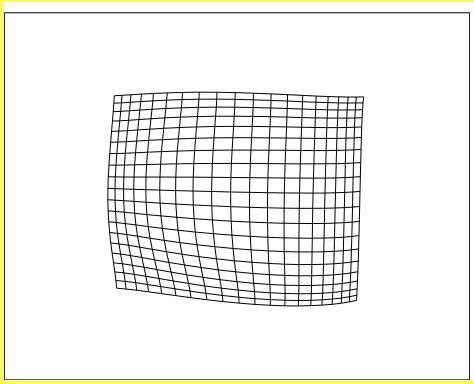
0



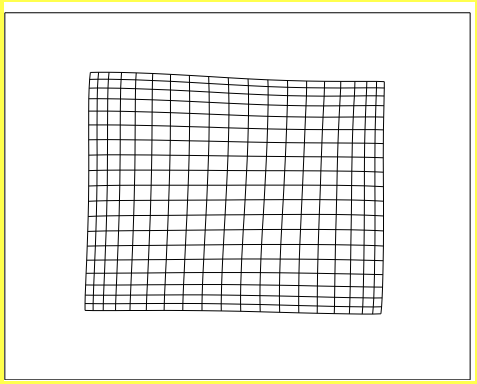
20



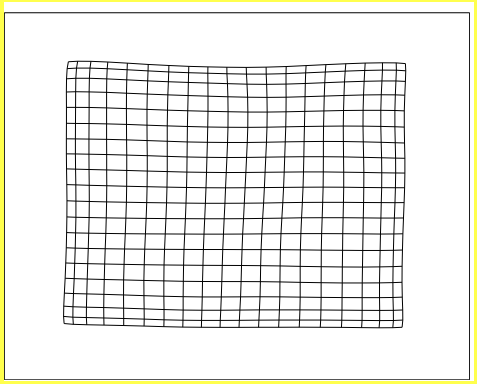
100



1000



5000



10000

Learning of SOM during 10,000 epochs for uniformly distributed data.

- The previous six subfigures show an example of the evolution of SOM for uniformly distributed data.
- The numbers below the subfigures give the number of epochs used in learning.
- During the learning process, the neighborhoods and learning rate parameter are both gradually reduced as described before.
- It can be seen that the ordering phase have been completed after 100 epochs.
- But the fine tuning of SOM in the convergence phase takes many more epochs.
- Other examples on evolution of SOM maps are given in Du's and Swamy's book: example 8.2 and 8.3 (traveling salesman problem).

An example with artificial data

- In this example, we illustrate further the properties of SOM and its use in visualization and clustering.
- We introduce the important concepts of component planes and U-matrix which are not discussed in any textbooks.
- The data vectors $\mathbf{x} = [x_1, x_2]^T$ in this example are two-dimensional and uniformly distributed.
- They are marked by red circles in Figure 3.
- There Var1 means the first variable x_1, and Var2 the second one x_2.
- The black crosses in Figure 3 denote the weight vectors of SOM after learning.
- The corresponding hexagonal SOM grid is shown in Figure 4.

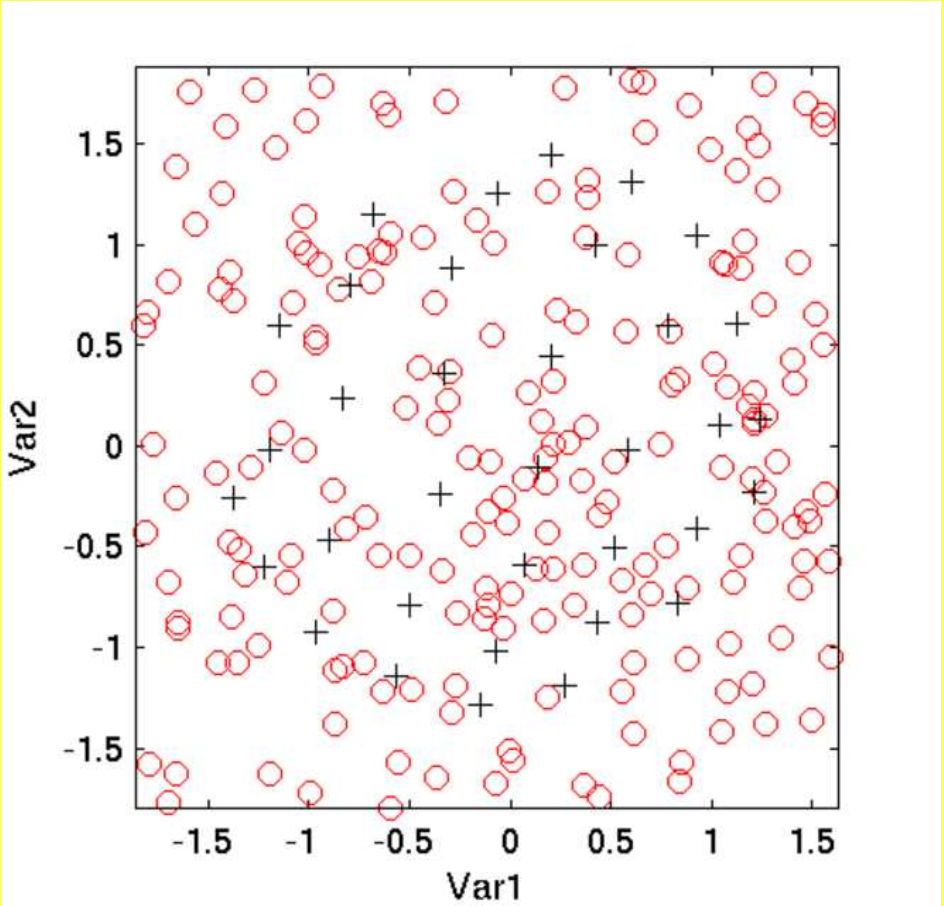


Figure 3: The data vectors (red circles) and weight vectors of SOM (black crosses).

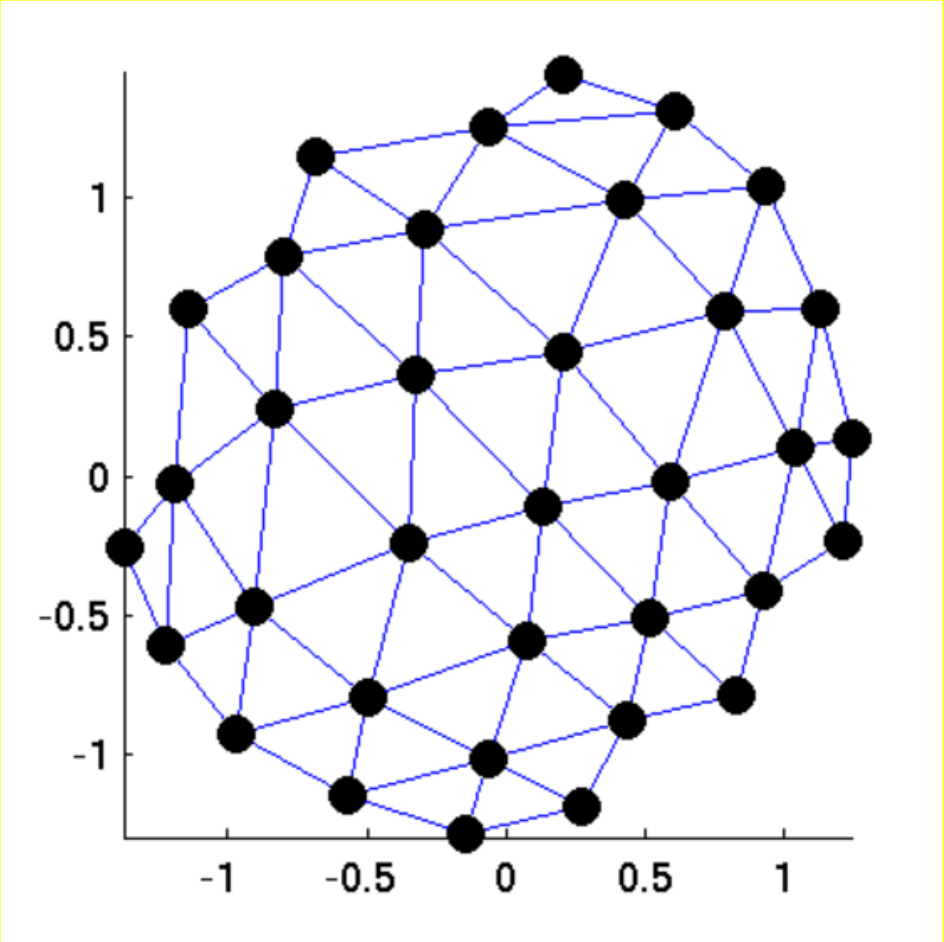


Figure 4: SOM grid.

- A component plane presents the values of a selected component of the weight vectors of the neurons in the SOM grid.
- Each component corresponds to some variable or parameter of interest.
- The component plane describes the variation of that component more accurately than the entire SOM map.
- Figure 5 shows the two component planes for the uniformly distributed data of Figure 3.
- The U-matrix (Unified distance matrix) visualizes the distances between the weight vectors in SOM.
- The U-matrix contains the distances of each weight vector to the adjacent weight vectors in the SOM map.

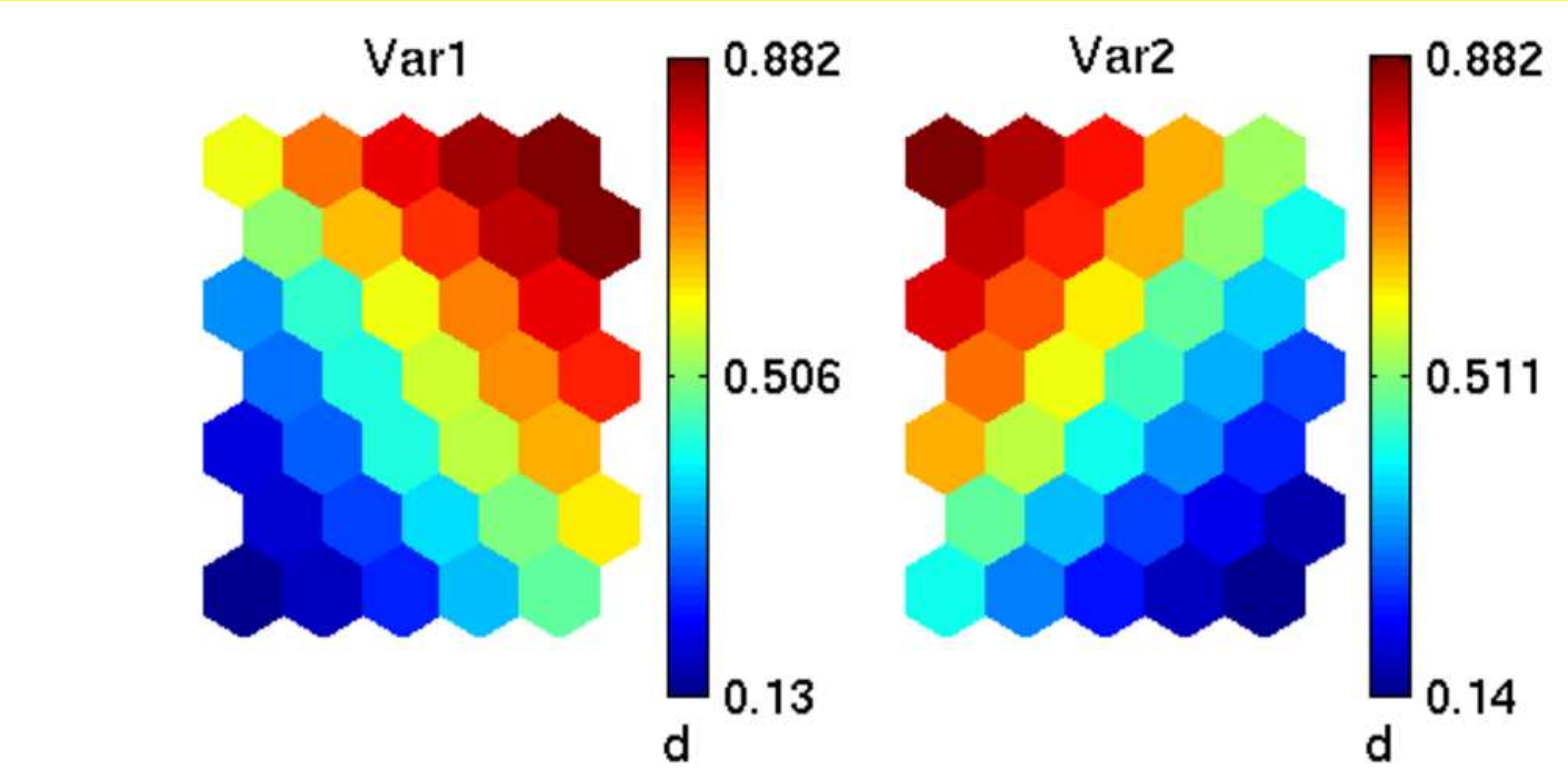


Figure 5: Component planes for the uniformly distributed data.

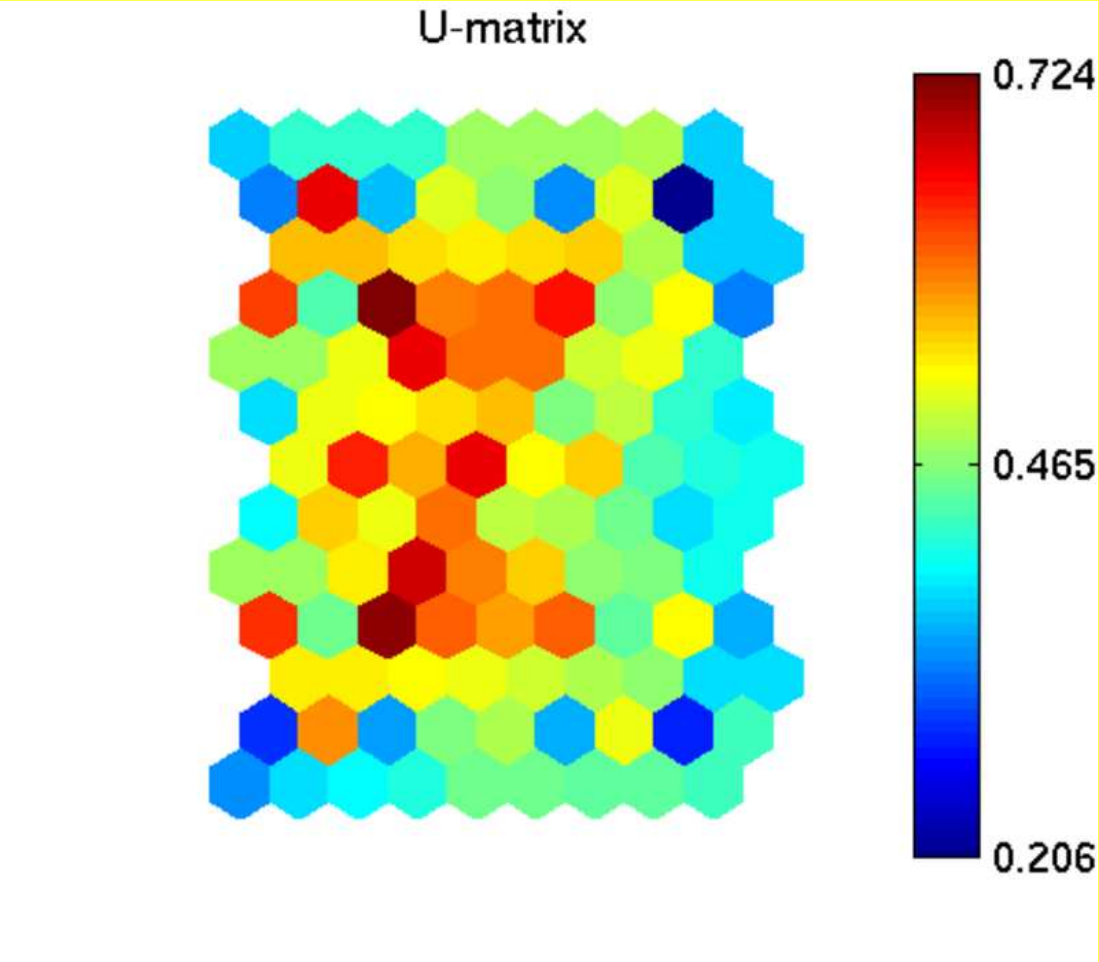


Figure 6: U-matrix for the uniformly distributed data.

- Furthermore, in the U-map the neuron itself gets a value which is the average of its distances to the adjacent neurons.
- Thus the U-matrix is nearly twice as large as the two-dimensional SOM map in both directions.
- Figure 6 shows U-matrix for the uniformly distributed data of Figure 3.
- The distances between the weight vectors of neurons are visualized using a color scale.
- In Figure 6, red color shows large distances and blue color small distances.
- The distances collected into the U-matrix can be used for clustering.
- The clusters are separated by red areas.

- Figure 7 shows the result of K-means clustering for the U-matrix of Figure 6: 10 clusters marked with different colors are found.
- The SOM grid colored using these clusters is shown in Figure 8
- These visualizations has been done using Matlab and Somtoolbox.
- Somtoolbox is a Matlab toolbox developed in our laboratory containing the extra features shown in this example and more, see `http://www.cis.hut.fi/somtoolbox/`.
- Check the subpage 'Gallery' there for an example on real-world Iris data showing clearer structure and even more options.
- A new version of the SOM toolbox moved to GitHub is available at http://research.ics.aalto.fi/software/somtoolbox/.

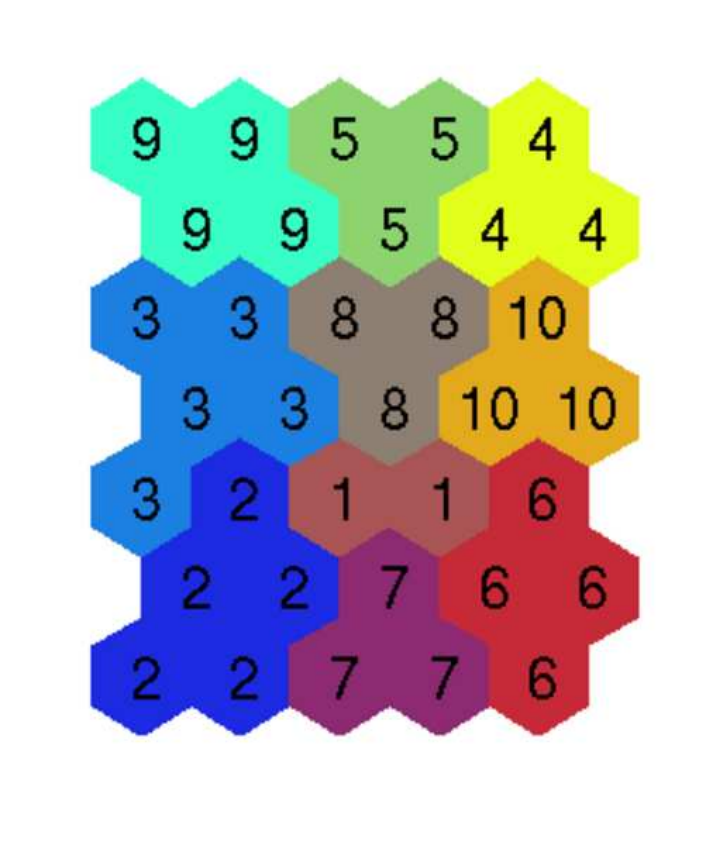


Figure 7: K-means clusters of the SOM map for uniformly distributed data.

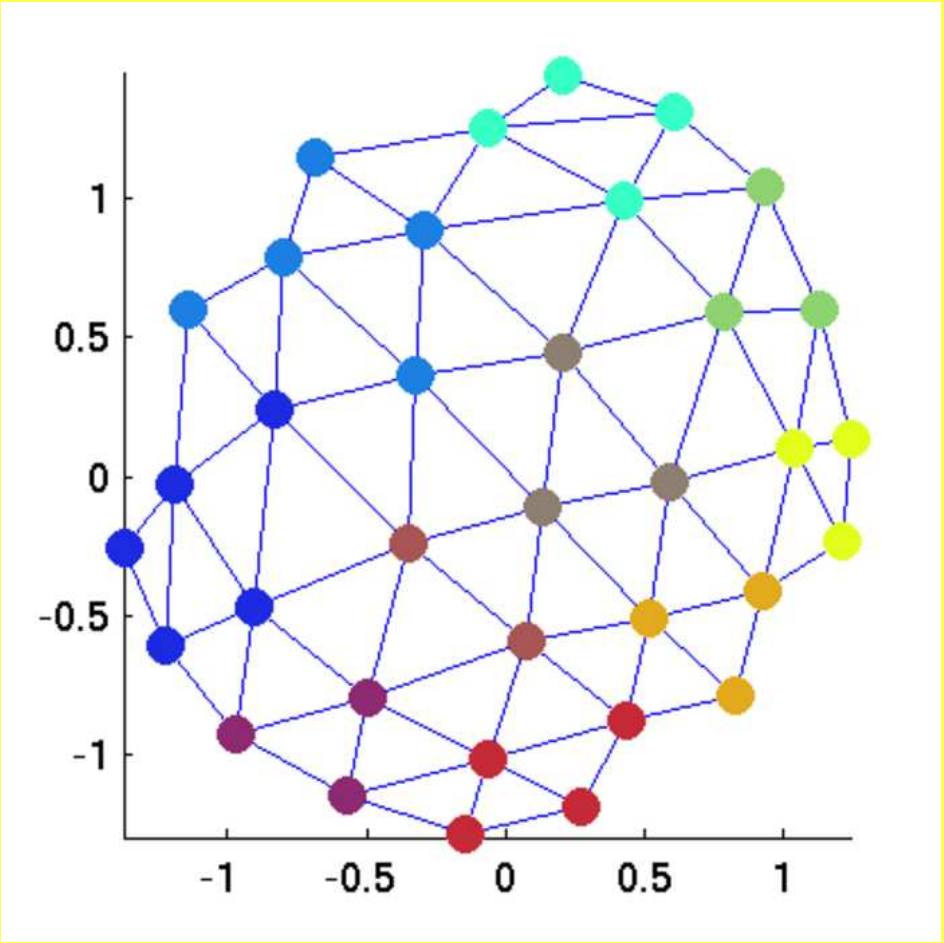


Figure 8: SOM grid colored with results of K-means clustering.

Real-world mobile network data

- In our laboratory, SOM has been used for analyzing and illustrating the properties of many real-world data sets.
- In this example, we show some results for mobile network data.
- The data was collected from three base stations of a mobile network.
- For each base station, four variables were measured:
  - The number of users;
  - The interference ratio of the signal of the base station compared with signals coming from other base stations and elsewhere;
  - The avarage noise in uplink transmission from the mobile phone to the base station;
  - And the average total power in downlink transmission from the base station to the mobile phone.

- The data vectors were 12-dimensional, obtained by concatenating the four-dimensional data vectors of three base stations into a single vector.
- Figure 9 shows the component planes of the weight vectors of SOM neurons.
- Each of them represents the SOM map for one variable.
- Figure 10 shows the U-matrix for this mobile data.
- In the figure, one can distinguish clear clusters separated by light areas corresponding to large distances.
- For example in the upper left part of the SOM map.
- These clusters corrospond to operational states of mobile cells.

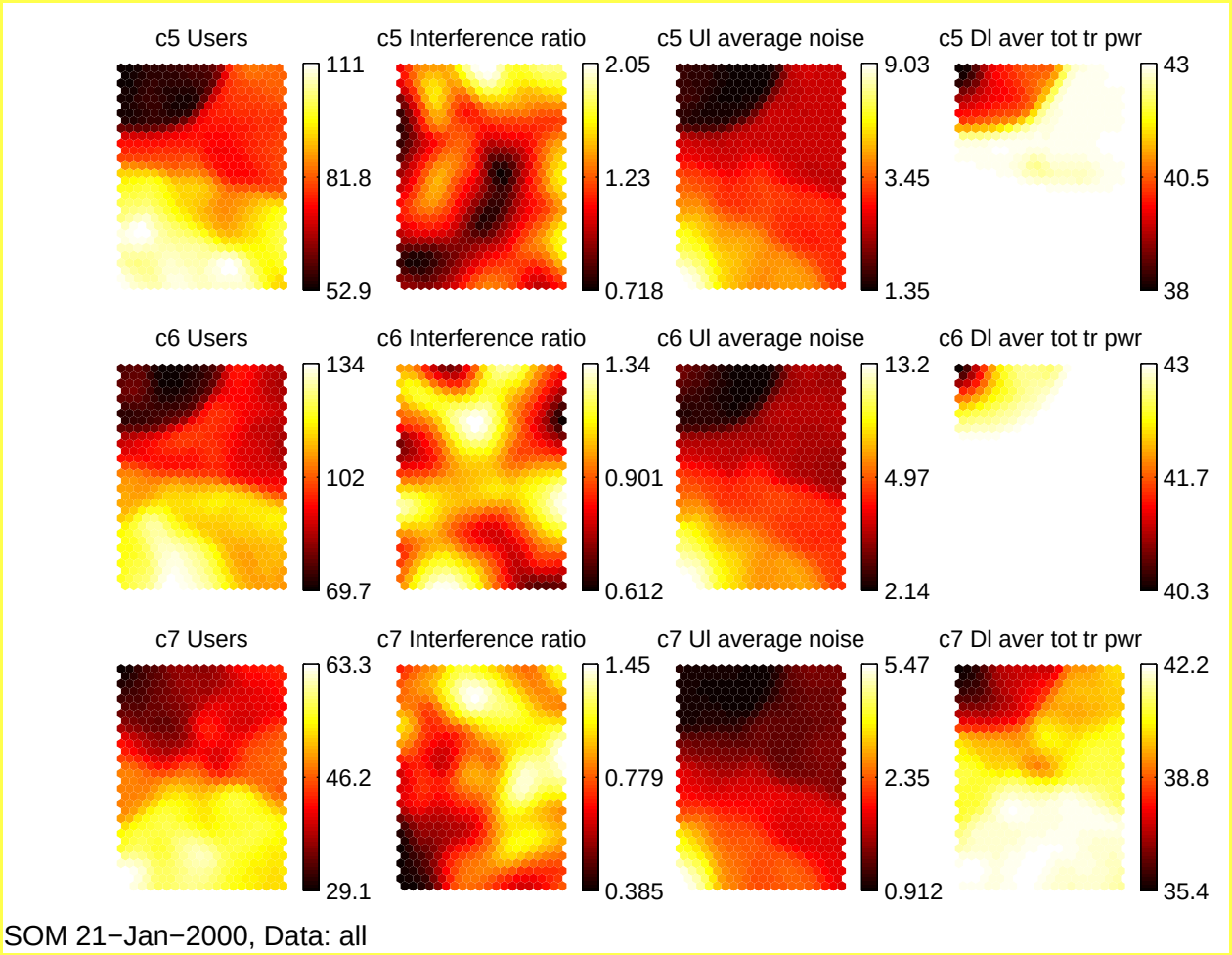


Figure 9: 12 SOM component planes for the mobile network data.

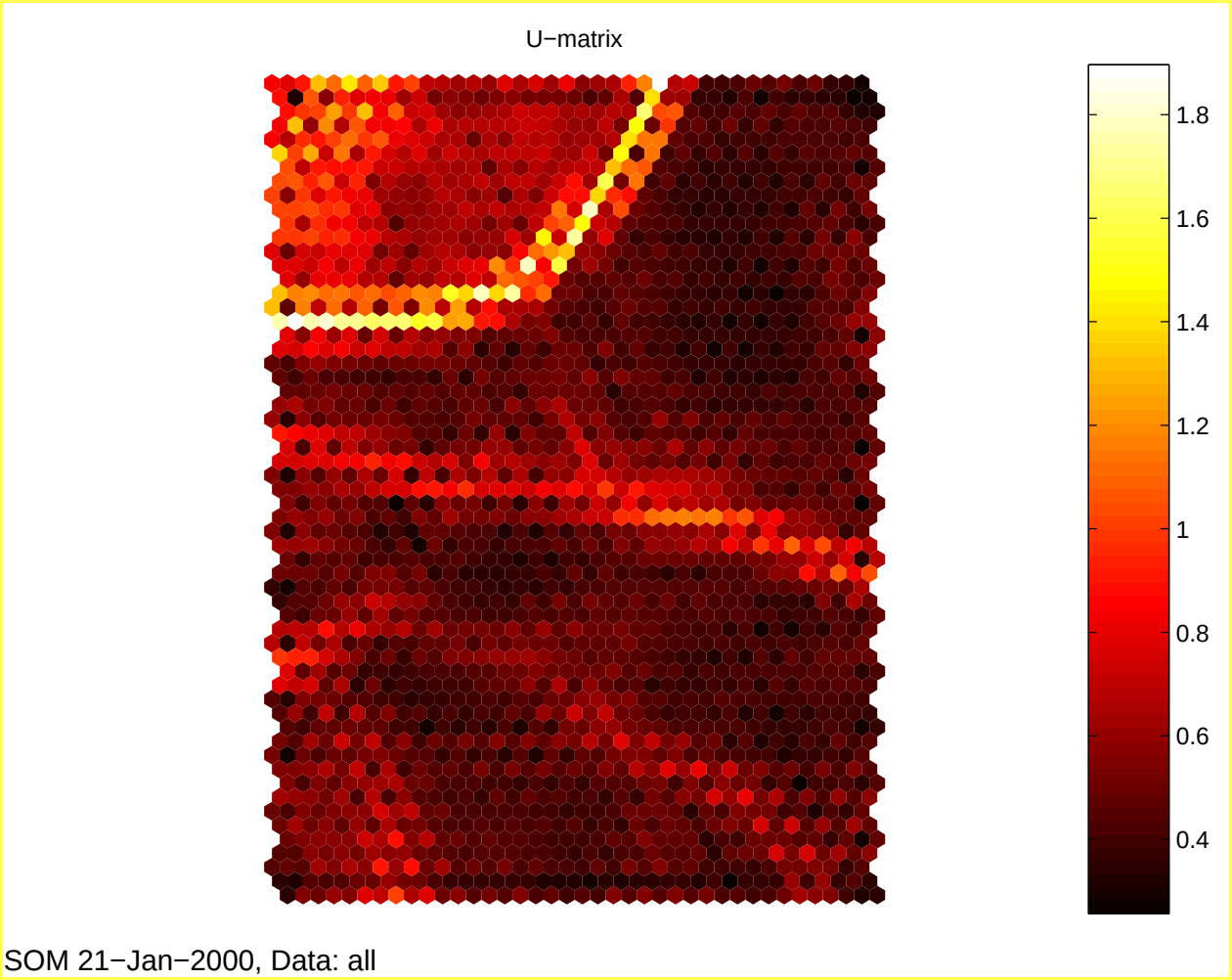


Figure 10: U-matrix for the mobile network data.

Vector Quantization

- Vector quantization (VQ) is a technique used for compression of speech and image data.
- The basic idea is to represent the available N data (input) vectors $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$ using a smaller set of reproduction or reconstruction vectors.
- Each reconstruction vector $\hat{\mathbf{x}}_i$ represents its own region of the data.
- The set $\hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2, \ldots, \hat{\mathbf{x}}_M$ of all the M reconstruction vectors should provide a good approximation of the input data.
- In vector quantization, the data (input) space is divided into M distinct regions.
- This can be done for example by applying K-means clustering or SOM to the input data $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$.

- The number M of clusters (neurons in SOM) is predetermined and sufficiently large for achieving adequate representation accuracy.
- For each region (cluster) $\mathcal{R}_i$, the associated reconstruction vector $\hat{\mathbf{x}}_i$ is usually selected to be the mean vector of the data vectors falling into that region.
- For a new data vector $\mathbf{x}$, its region $\mathcal{R}_q$ is determined at first.
- Usually this takes place by computing the squared Euclidean distances $\| \mathbf{x} - \hat{\mathbf{x}}_i \|^2$ between the data vector $\mathbf{x}$ and all the reconstruction vectors $\hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2, \ldots, \hat{\mathbf{x}}_M$
- The reconstruction vector $\hat{\mathbf{x}}_q$ to which $\mathbf{x}$ has the minimum distance is then selected to represent that data vector $\mathbf{x}$.
- Using an encoded version of this reconstruction vector, considerable savings in storage or transmission bandwidth can be realized.

- The collection of all the M reconstruction vectors $\hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2, \ldots, \hat{\mathbf{x}}_M$ is called the codebook of the vector quantizer.
- Its members are called code words.
- A vector quantizer with minimum encoding distortion is called a Voronoi or nearest-neighbor quantizer.
- Voronoi cells are partition cells provided by the nearest-neighbor rule based on the Euclidean distance.
- An example of four Voronoi cells, separated by dashed lines, is presented in Figure 11.
- The reconstruction (Voronoi) vectors $\hat{\mathbf{x}}_i$ corresponding to these regions (cells) are shown by circles.

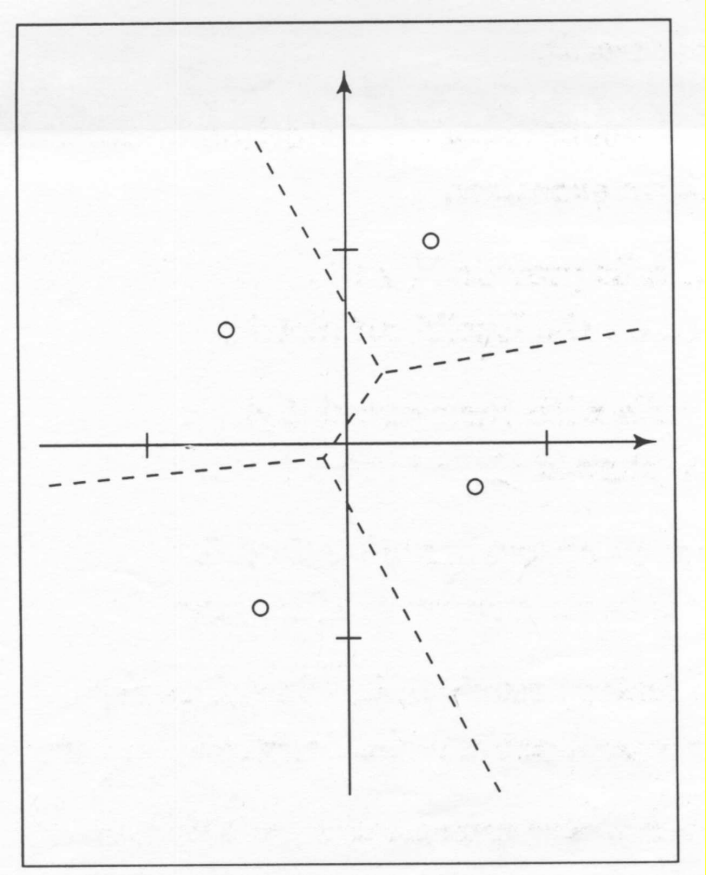


Figure 11: An example of 4 Voronoi cells, separated by dashed lines, with their associated reconstruction (Voronoi) vectors (circles).

- The standard squared Euclidean distance between $\mathbf{x}$ and $\hat{\mathbf{x}}$ is defined by

$$d(\mathbf{x}, \hat{\mathbf{x}}) = \| \mathbf{x} - \hat{\mathbf{x}} \|^2 = (\mathbf{x} - \hat{\mathbf{x}})^T (\mathbf{x} - \hat{\mathbf{x}}) \tag{7}$$

- Instead of it, one can use the Mahalanobis distance

$$d_M(\mathbf{x}, \hat{\mathbf{x}}) = (\mathbf{x} - \hat{\mathbf{x}})^T \mathbf{C}_x^{-1} (\mathbf{x} - \hat{\mathbf{x}}) \tag{8}$$

- There the weighting matrix is the inverse of the covariance matrix $\mathbf{C}_x$ of the input vectors:

$$\mathbf{C}_x = \mathrm{E}\{(\mathbf{x} - \mathbf{m})(\mathbf{x} - \mathbf{m})^T\} \tag{9}$$

  where $\mathbf{m} = \mathrm{E}\{\mathbf{x}\}$ is the mean vector of $\mathbf{x}$.

- The Mahalanobis distance normalizes the distance with respect to the variances (and cross-covariances) of the components of $\mathbf{x}$.

Learning Vector Quantization (LVQ)

- Learning vector quantization (LVQ) is a supervised learning method based on vector quantization.
- It is a supervised version of self-organizing map (SOM) for classification purposes resembling closely SOM.
- Using class information, it moves the Voronoi vectors slightly for improving the decision regions of the classifier.
- Instead of Voronoi vectors one can use any vectors that represent the decision regions of classes.
- There are usually several such representation or reconstruction vectors per each class.
- We denote now these reconstruction vectors $\hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2, \ldots, \hat{\mathbf{x}}_M$ by $\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_M$.

- Take an input vector $\mathbf{x}$ at random from the data space.
- If the class labels of $\mathbf{x}$ and the winning weight vector $\mathbf{w}$ closest to $\mathbf{x}$ agree, $\mathbf{w}$ is moved in LVQ in the direction of $\mathbf{x}$.
- If the class labels of $\mathbf{x}$ and $\mathbf{w}$ are different, $\mathbf{w}$ is moved away from the input vector $\mathbf{x}$.
- Assumption: there are many more input (data) vectors $\mathbf{x}_1, \ldots, \mathbf{x}_N$ than Voronoi vectors $\mathbf{w}_1, \ldots, \mathbf{w}_M$ ($N \gg M$).
- More formally, the Learning Vector Quantization (LVQ) algorithm is as follows:
- Suppose that the weight vector $\mathbf{w}_q(k)$ at iteration k is closest to the input vector $\mathbf{x}_i$:

$$\mathbf{w}_q(k) = \arg\min_j \| \mathbf{x}_i - \mathbf{w}_j(k) \|_2, \qquad j = 1, 2, \ldots, M \tag{10}$$

- Let $\mathcal{C}_{\mathbf{w}_q}$ denote the class of $\mathbf{w}_q(k)$ and $\mathcal{C}_{\mathbf{x}_i}$ the class of $\mathbf{x}_i$.
- The weight vector $\mathbf{w}_q(k)$ is adjusted as follows:
  - If the classes are the same: $\mathcal{C}_{\mathbf{w}_q} = \mathcal{C}_{\mathbf{x}_i}$, then

$$\mathbf{w}_q(k+1) = \mathbf{w}_q(k) + \mu(k)[\mathbf{x}_i - \mathbf{w}_q(k)] \tag{11}$$

  - If the classes are different: $\mathcal{C}_{\mathbf{w}_q} \neq \mathcal{C}_{\mathbf{x}_i}$, then

$$\mathbf{w}_q(k+1) = \mathbf{w}_q(k) - \mu(k)[\mathbf{x}_i - \mathbf{w}_q(k)] \tag{12}$$

- The other (non-closest) Voronoi vectors are not changed.
- The learning parameter $\mu(k)$ usually decreases monotonically with the number of iterations k.
- For example, $\mu(k)$ may decrease linearly from its initial value 0.1.
- The weight vectors typically converge after several epochs.
- The weight vectors are initialized suitably, typically using K-means

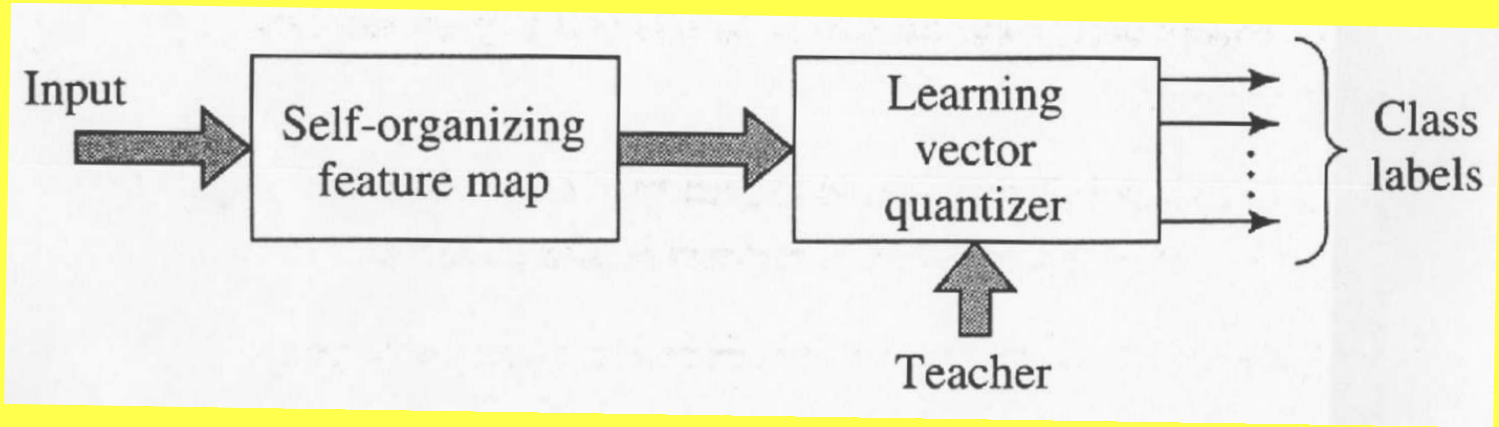
clustering or SOM.

- The LVQ algorithm is stopped by checking whether the chosen stopping condition is fulfilled.
- The stopping condition can be:
  - The total number of desired training epochs.
  - Adequate convergence of the weight vectors.
  - The learning parameter $\mu(k)$ becomes sufficiently small.
- The above algorithm is the basic version of LVQ, called LVQ1.
- It is given as algorithm 8.2 in Du's and Swamy's book.
- LVQ2 and LVQ3 are refined versions of learning vector quantization which are not discussed in our course.
- Example 8.4 in Du's and Swamy's book presents a simple classification example using LVQ for artificially generated data.

- In practice, the SOM and LVQ methods perform well.
- But they are very difficult to analyze rigorously mathematically.
- Self-organizing map has been occasionally criticized because it is rather heuristic.
- There is no probabilistic latent variable model behind it.
- A method called GTM (generative topographic mapping) was developed by Bishop et al. to mimick SOM.
- GTM is based on a probabilistic latent variable model, and uses the maximum likelihood method for optimizing the model.
- However, its performance is not as good as for SOM.

Combining SOM and LVQ

- SOM provides an approximate method for computing the weight vectors needed in LVQ in an unsupervised manner.
- Computation of the SOM feature map can be viewed as the first stage of adaptively solving a pattern classification problem.
- The second stage is learning vector quantization, which fine tunes the SOM feature map; see the figure below.



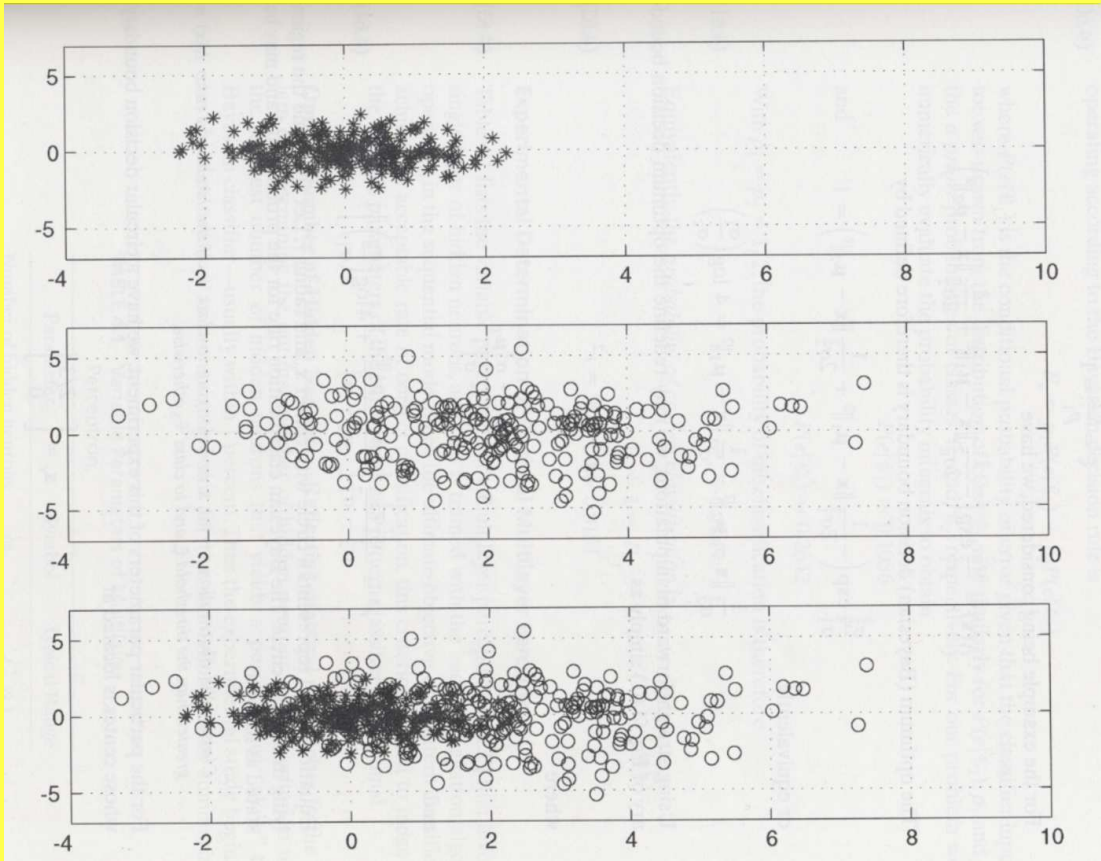


Figure 12: Data vectors of class $\mathcal{C}_1$ (top), class $\mathcal{C}_2$ (middle), and both the classes (bottom).

Computer experiment

- Consider again the simple pattern classification problem first introduced when discussing MLP networks.
- There are two two-dimensional overlapping Gaussian distributed classes.
- Data vectors from class $\mathcal{C}_1$, class $\mathcal{C}_2$, and both the classes are depicted in Figure 12.
- Note the different scales of horizontal and vertical axes.
- The optimal Bayes classifier for this classification problem achieves the accuracy of 81.51%.
- Figure 13 shows the two-dimensional SOM map of 5×5 neurons after training with 500 data vectors.
- The weight vectors have been labeled according to their response to

the test data drawn from the distributions of the input vectors.

- The decision boundary given by SOM alone is presented in Figure 14.
- Figure 15 shows the modified feature map after supervised fine tuning using the LVQ algorithm using 500 training vectors.
- Finally, Figure 16 shows the decision boundary given by combined use of SOM and LVQ.
- The performance of SOM and LVQ varied only a little in 10 independent trials of this experiment.
- Each of them used 30,000 pattern vectors as test data.
- The average performance of SOM was 79.6%.
- For the combined LVQ and SOM method it was 80.5%.

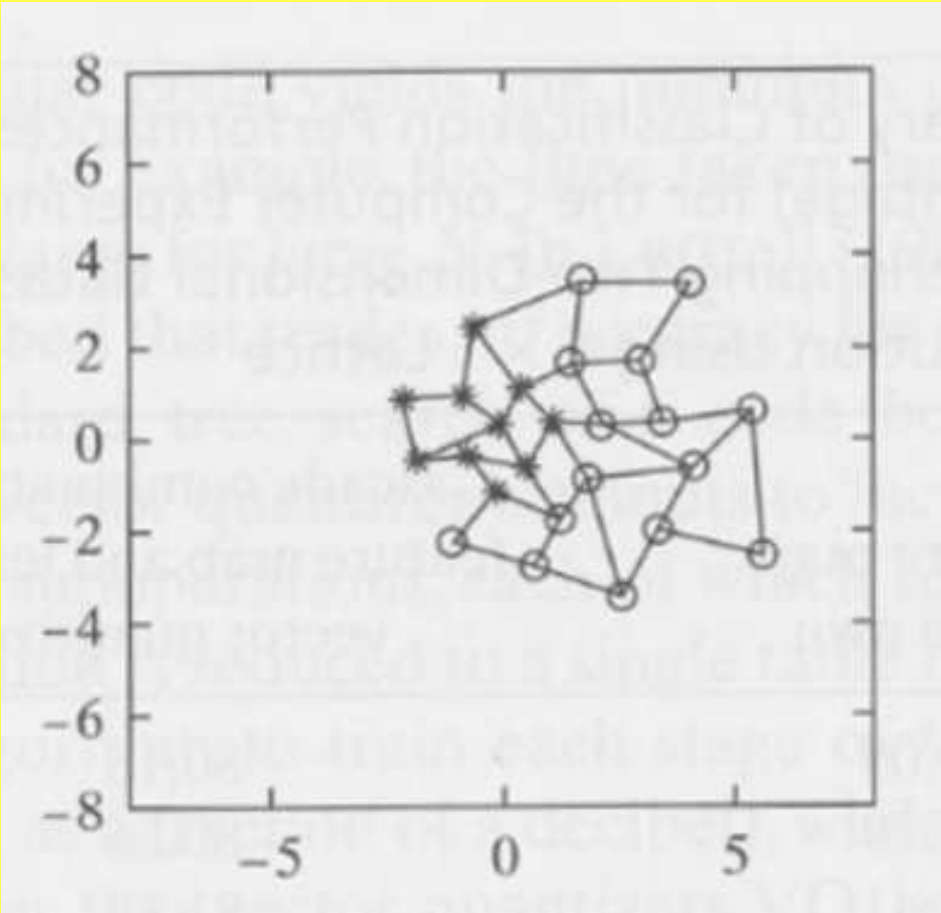


Figure 13: Labeled two-dimensional 5×5 SOM map after training.

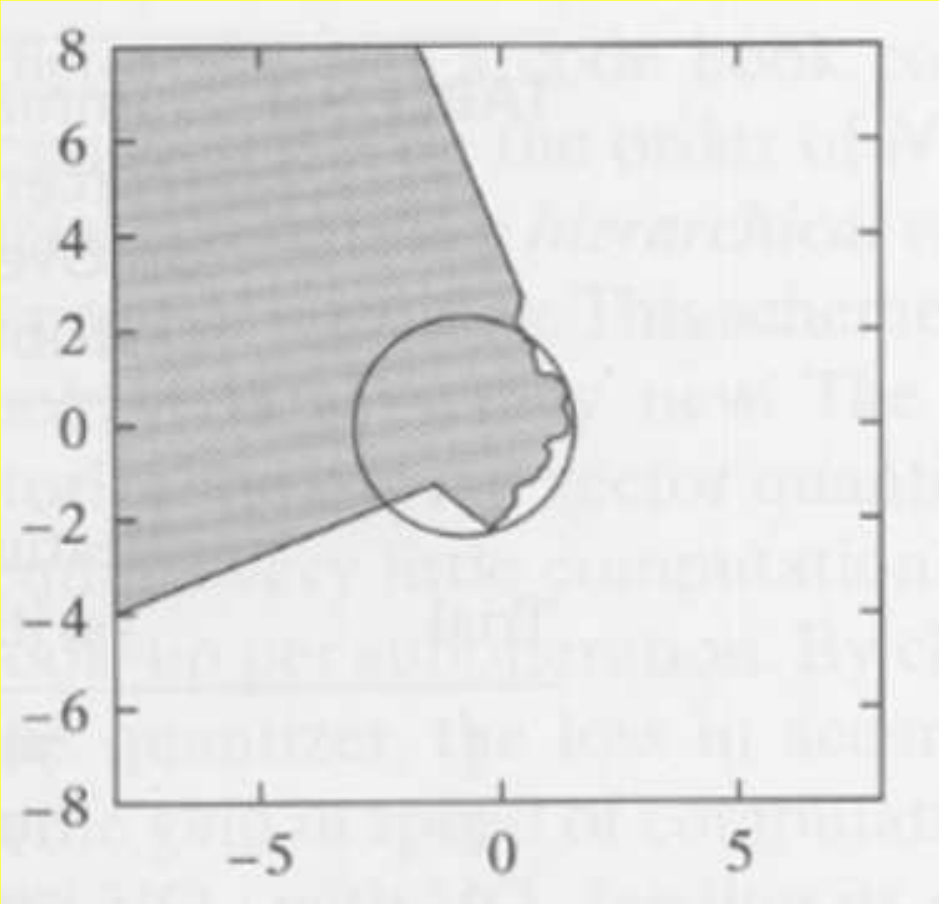


Figure 14: Decision boundary given by SOM only.

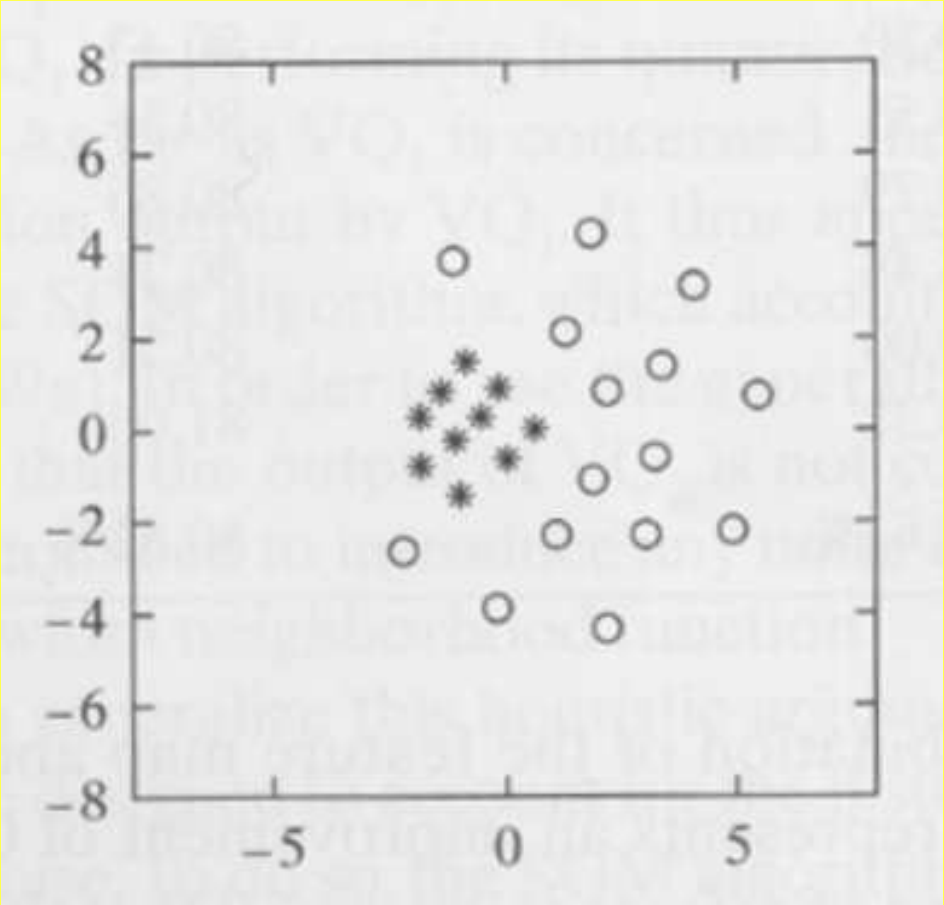


Figure 15: Labeled map after LVQ fine tuning.

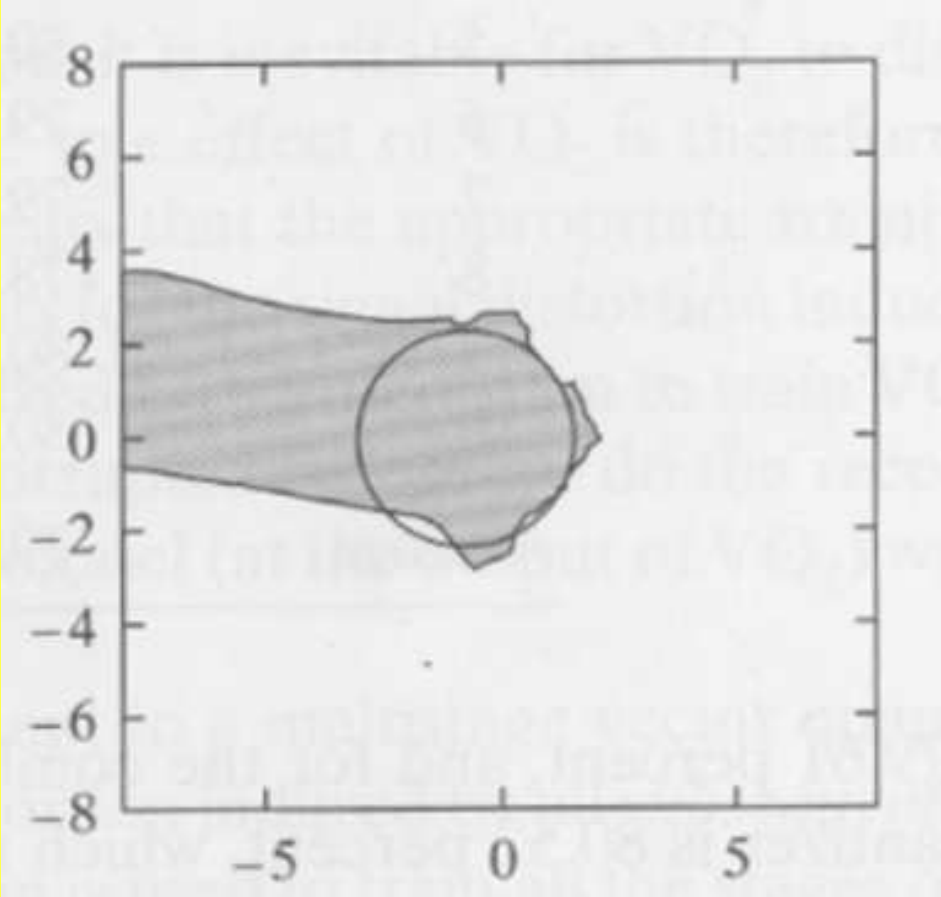


Figure 16: Decision boundary given by combined use of SOM and LVQ.