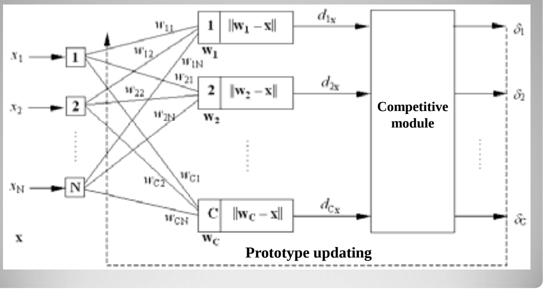
CI & A

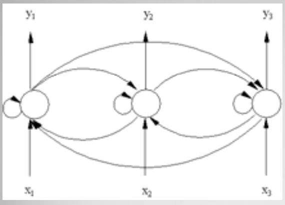
ANNS FOR CLASSIFICATION AND RECOGNITION

COMPETITIVE NETWORKS

Basic architecture

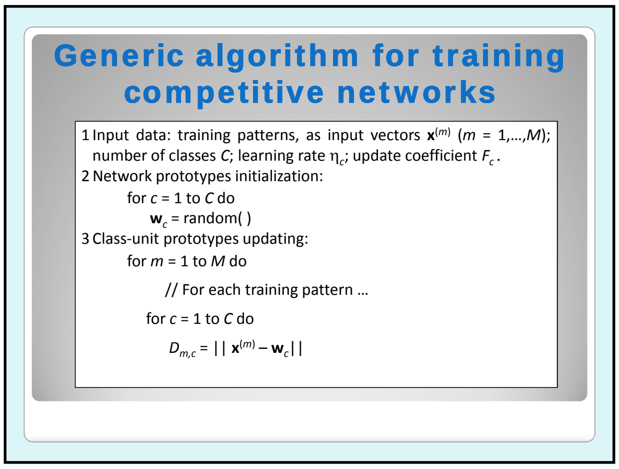


Competitive module – MaxNet / MinNet nets



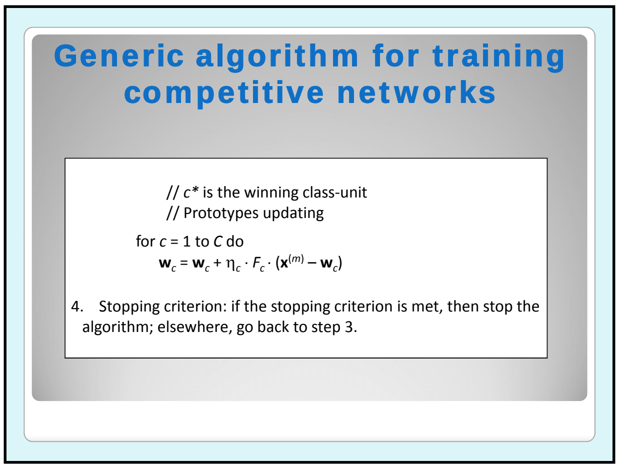
$$y_j^{(t)} = max\left(0, x_j^{(t-1)}\right)$$

$$x_j^{(t)} = y_j^{(t-1)} - \alpha \cdot \sum_{\substack{k=1 \\ k \neq j}}^{N} y_k^{(t-1)}$$



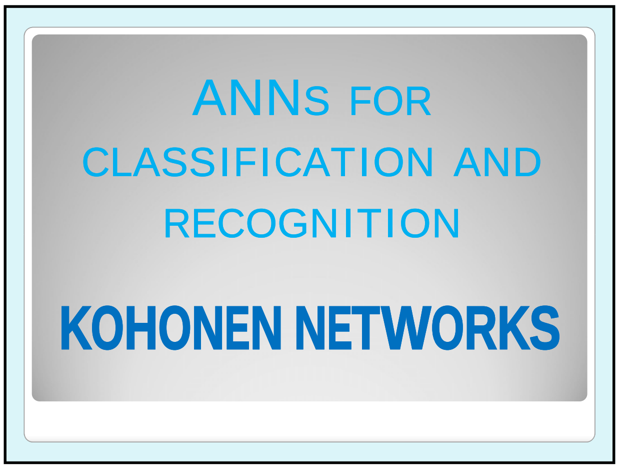
Generic algorithm for training competitive networks

```
1 Input data: training patterns, as input vectors $\mathbf{x}^{(m)}$ ($m$ = 1,…,$M$);
  number of classes $C$; learning rate $\eta_c$; update coefficient $F_c$ .
2 Network prototypes initialization:
      for $c$ = 1 to $C$ do
         $\mathbf{w}_c$ = random( )
3 Class-unit prototypes updating:
      for $m$ = 1 to $M$ do
           // For each training pattern …
        for $c$ = 1 to $C$ do
           $D_{m,c} = || \mathbf{x}^{(m)} - \mathbf{w}_c ||$
```



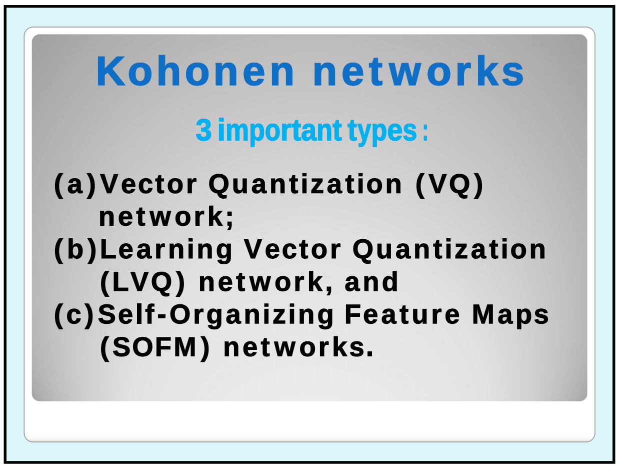
Generic algorithm for training competitive networks

```
           // $c^*$ is the winning class-unit
           // Prototypes updating
        for $c$ = 1 to $C$ do
           $\mathbf{w}_c = \mathbf{w}_c + \eta_c \cdot F_c \cdot (\mathbf{x}^{(m)} - \mathbf{w}_c)$
4. Stopping criterion: if the stopping criterion is met, then stop the
   algorithm; elsewhere, go back to step 3.
```



ANNs for classification and recognition

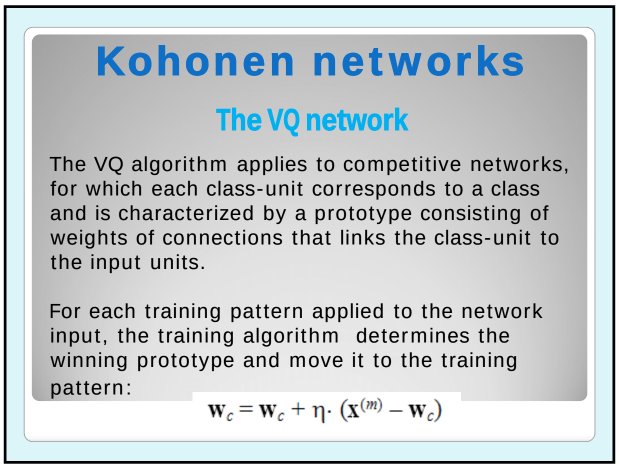
KOHONEN NETWORKS



Kohonen networks

3 important types :

(a) Vector Quantization (VQ) network;
(b) Learning Vector Quantization (LVQ) network, and
(c) Self-Organizing Feature Maps (SOFM) networks.



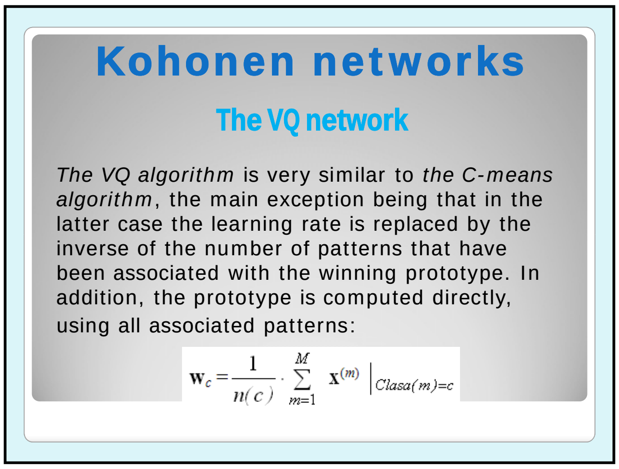
Kohonen networks

The VQ network

The VQ algorithm applies to competitive networks, for which each class-unit corresponds to a class and is characterized by a prototype consisting of weights of connections that links the class-unit to the input units.

For each training pattern applied to the network input, the training algorithm determines the winning prototype and move it to the training pattern:

$$\mathbf{w}_c = \mathbf{w}_c + \eta \cdot (\mathbf{x}^{(m)} - \mathbf{w}_c)$$

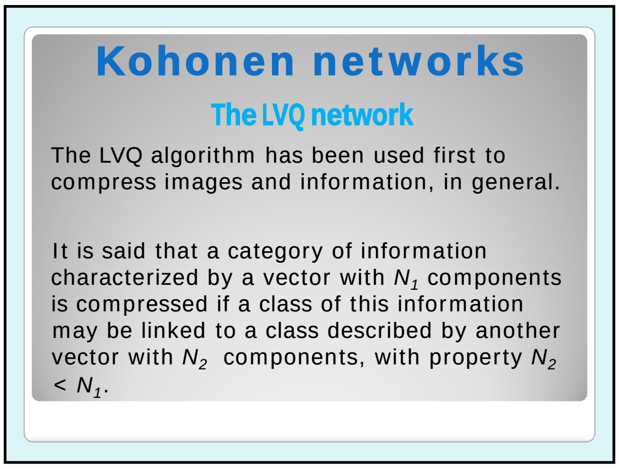


Kohonen networks

The VQ network

The VQ algorithm is very similar to *the C-means algorithm*, the main exception being that in the latter case the learning rate is replaced by the inverse of the number of patterns that have been associated with the winning prototype. In addition, the prototype is computed directly, using all associated patterns:

$$\mathbf{w}_c = \frac{1}{n(c)} \cdot \sum_{m=1}^{M} \mathbf{x}^{(m)} \Big|_{Clasa(m)=c}$$

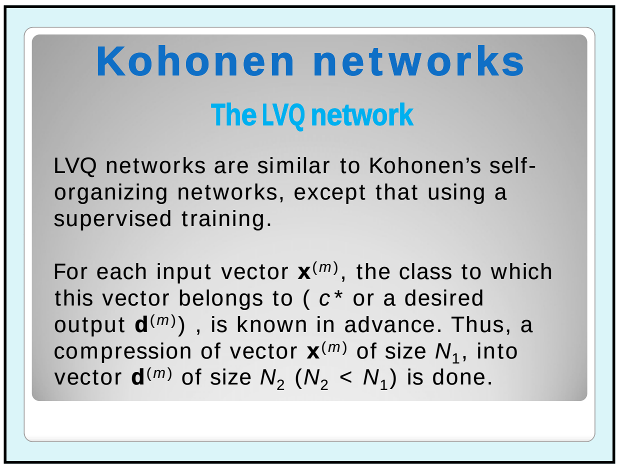


Kohonen networks

The LVQ network

The LVQ algorithm has been used first to compress images and information, in general.

It is said that a category of information characterized by a vector with N_1 components is compressed if a class of this information may be linked to a class described by another vector with N_2 components, with property $N_2 < N_1$.

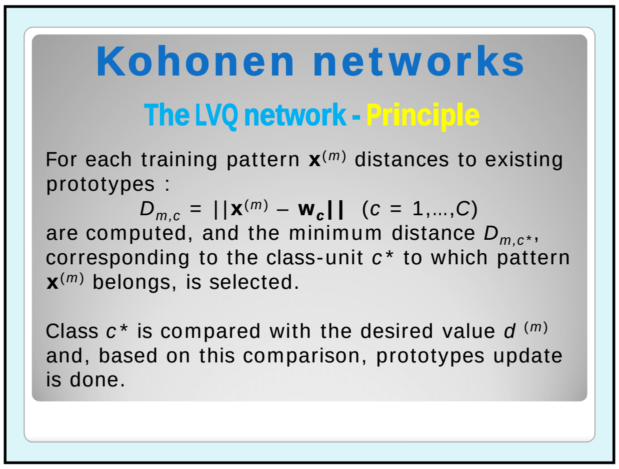


Kohonen networks

The LVQ network

LVQ networks are similar to Kohonen's self-organizing networks, except that using a supervised training.

For each input vector $\mathbf{x}^{(m)}$, the class to which this vector belongs to ($c*$ or a desired output $\mathbf{d}^{(m)}$) , is known in advance. Thus, a compression of vector $\mathbf{x}^{(m)}$ of size N_1, into vector $\mathbf{d}^{(m)}$ of size N_2 ($N_2 < N_1$) is done.



Kohonen networks

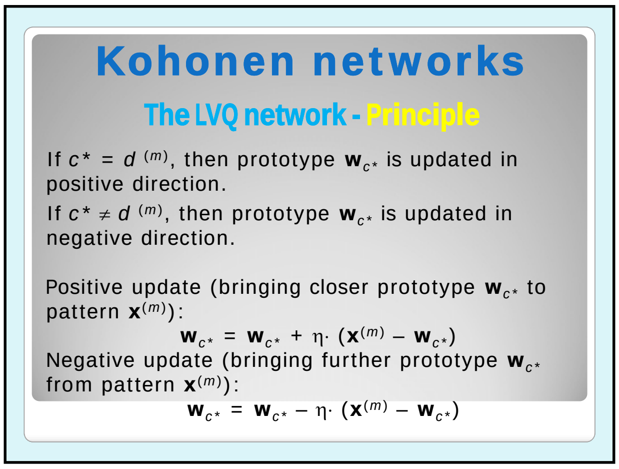
The LVQ network - Principle

For each training pattern $\mathbf{x}^{(m)}$ distances to existing prototypes :

$$D_{m,c} = ||\mathbf{x}^{(m)} - \mathbf{w}_c|| \quad (c = 1,\dots,C)$$

are computed, and the minimum distance $D_{m,c*}$, corresponding to the class-unit $c*$ to which pattern $\mathbf{x}^{(m)}$ belongs, is selected.

Class $c*$ is compared with the desired value $d^{(m)}$ and, based on this comparison, prototypes update is done.



Kohonen networks

The LVQ network - Principle

If $c^* = d^{(m)}$, then prototype $\mathbf{w}_{c^*}$ is updated in positive direction.

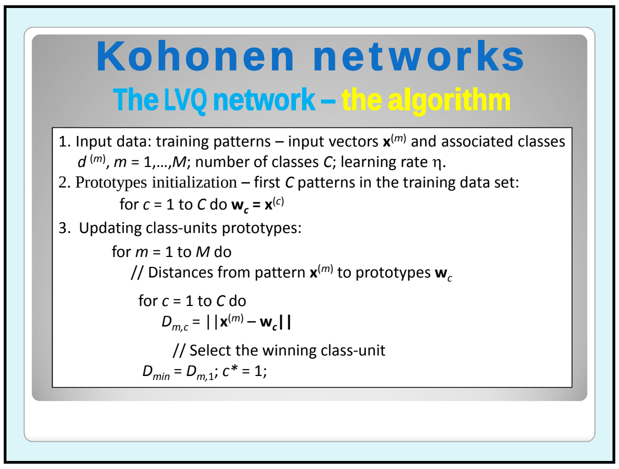
If $c^* \neq d^{(m)}$, then prototype $\mathbf{w}_{c^*}$ is updated in negative direction.

Positive update (bringing closer prototype $\mathbf{w}_{c^*}$ to pattern $\mathbf{x}^{(m)}$):

$$\mathbf{w}_{c^*} = \mathbf{w}_{c^*} + \eta \cdot (\mathbf{x}^{(m)} - \mathbf{w}_{c^*})$$

Negative update (bringing further prototype $\mathbf{w}_{c^*}$ from pattern $\mathbf{x}^{(m)}$):

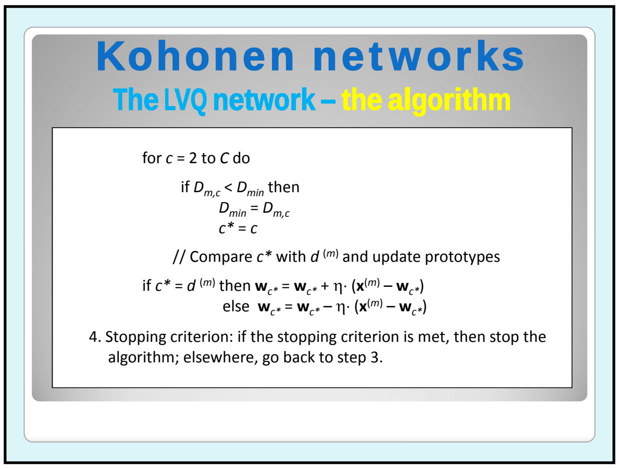
$$\mathbf{w}_{c^*} = \mathbf{w}_{c^*} - \eta \cdot (\mathbf{x}^{(m)} - \mathbf{w}_{c^*})$$



Kohonen networks

The LVQ network – the algorithm

```
1. Input data: training patterns – input vectors $\mathbf{x}^{(m)}$ and associated classes
   $d^{(m)}$, $m = 1,\dots,M$; number of classes $C$; learning rate $\eta$.
2. Prototypes initialization – first $C$ patterns in the training data set:
       for $c = 1$ to $C$ do $\mathbf{w}_c = \mathbf{x}^{(c)}$
3. Updating class-units prototypes:
       for $m = 1$ to $M$ do
          // Distances from pattern $\mathbf{x}^{(m)}$ to prototypes $\mathbf{w}_c$
          for $c = 1$ to $C$ do
              $D_{m,c} = ||\mathbf{x}^{(m)} - \mathbf{w}_c||$
              // Select the winning class-unit
          $D_{min} = D_{m,1}$; $c^* = 1$;
```



Kohonen networks

The LVQ network – the algorithm

```
          for $c = 2$ to $C$ do
              if $D_{m,c} < D_{min}$ then
                  $D_{min} = D_{m,c}$
                  $c^* = c$
              // Compare $c^*$ with $d^{(m)}$ and update prototypes
          if $c^* = d^{(m)}$ then $\mathbf{w}_{c^*} = \mathbf{w}_{c^*} + \eta \cdot (\mathbf{x}^{(m)} - \mathbf{w}_{c^*})$
                else $\mathbf{w}_{c^*} = \mathbf{w}_{c^*} - \eta \cdot (\mathbf{x}^{(m)} - \mathbf{w}_{c^*})$
    4. Stopping criterion: if the stopping criterion is met, then stop the
       algorithm; elsewhere, go back to step 3.
```

Kohonen networks

The SOFM network – biological inspiration

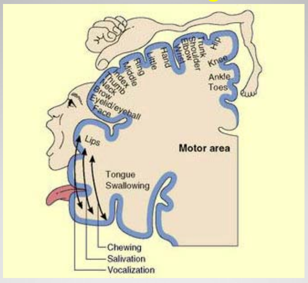
Mapping of sensory segments in the human brain:

Spatial relationships between stimuli corresponds to spatial relationships between neurons.

A *distorted* projection / image of the human body appears on the cortex.

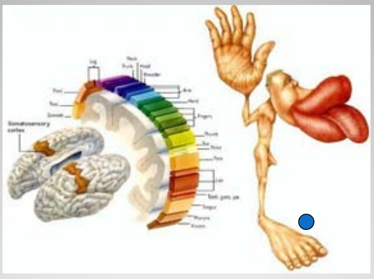
Kohonen networks

The SOFM network – biological inspiration



Kohonen networks

The SOFM network – biological inspiration



Kohonen networks

The SOFM network

SOFM networks were designed for ***unsupervised learning***, sometimes called ***self-organization.***

The training data set contains only input data, and the SOFM network tries to learn the data structure.

Self - organization

… ability of a system to discover and learn the structure of input data, even in the absence of any information on this structure.

Kohonen networks

The SOFM network

Kohonen network (SOFM) is an extension of the standard competitive learning to the ***feature maps*** concept.

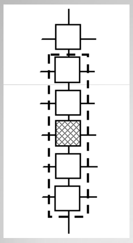
A feature map can be obtained if a certain topological organization is added to units from a competitive learning network.

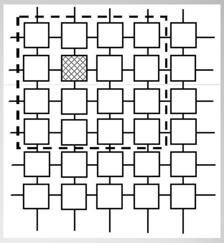
Units are placed in the nodes of a ***support-grid***, usually 2-dimensional, but sometimes 1-dimensional or even 3- or multi-dimensional.

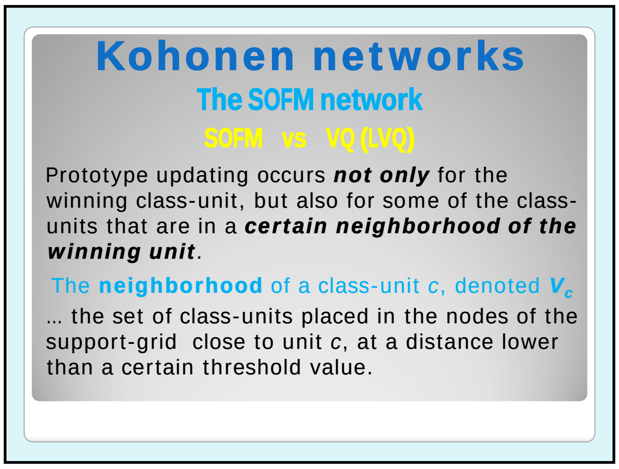
Kohonen networks

The SOFM network

Support – grids and Class - units







Kohonen networks

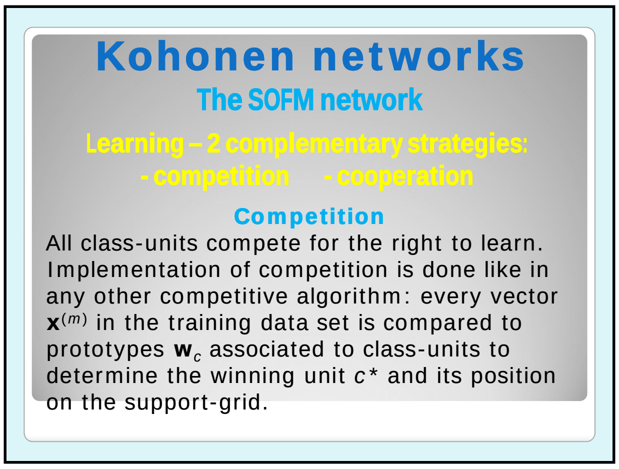
The SOFM network

SOFM vs VQ (LVQ)

Prototype updating occurs ***not only*** for the winning class-unit, but also for some of the class-units that are in a ***certain neighborhood of the winning unit***.

The **neighborhood** of a class-unit *c*, denoted $\boldsymbol{V_c}$

… the set of class-units placed in the nodes of the support-grid close to unit *c*, at a distance lower than a certain threshold value.



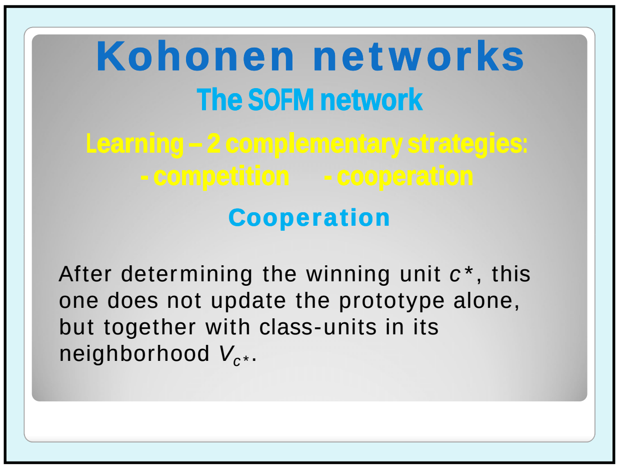
Kohonen networks

The SOFM network

Learning – 2 complementary strategies:
- competition - cooperation

Competition

All class-units compete for the right to learn. Implementation of competition is done like in any other competitive algorithm: every vector $\mathbf{x}^{(m)}$ in the training data set is compared to prototypes $\mathbf{w}_c$ associated to class-units to determine the winning unit c^* and its position on the support-grid.



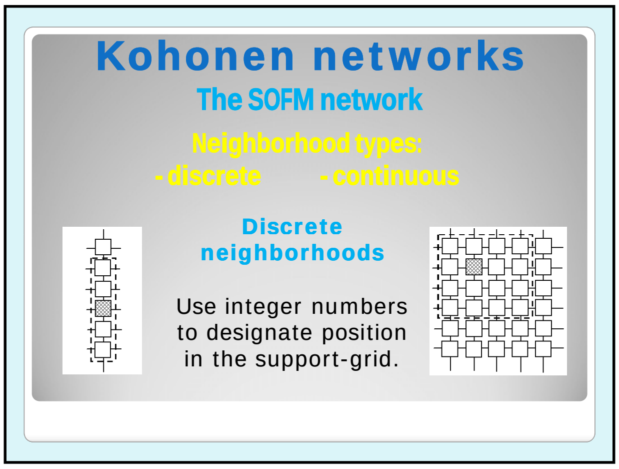
Kohonen networks

The SOFM network

Learning – 2 complementary strategies:
- competition - cooperation

Cooperation

After determining the winning unit c^*, this one does not update the prototype alone, but together with class-units in its neighborhood V_{c^*}.



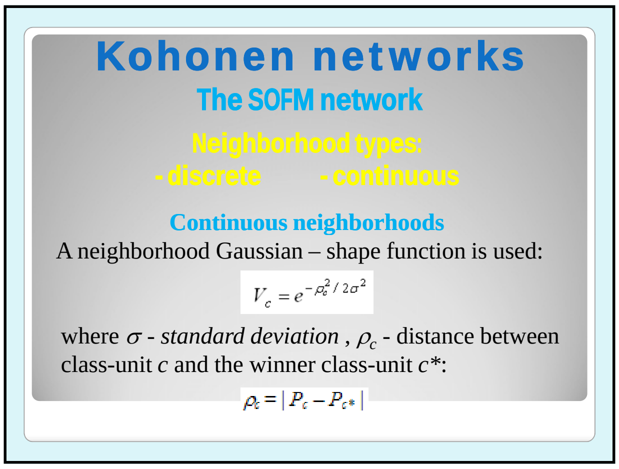
Kohonen networks

The SOFM network

Neighborhood types:
- discrete - continuous

Discrete neighborhoods

Use integer numbers to designate position in the support-grid.



Kohonen networks

The SOFM network

Neighborhood types:
- discrete - continuous

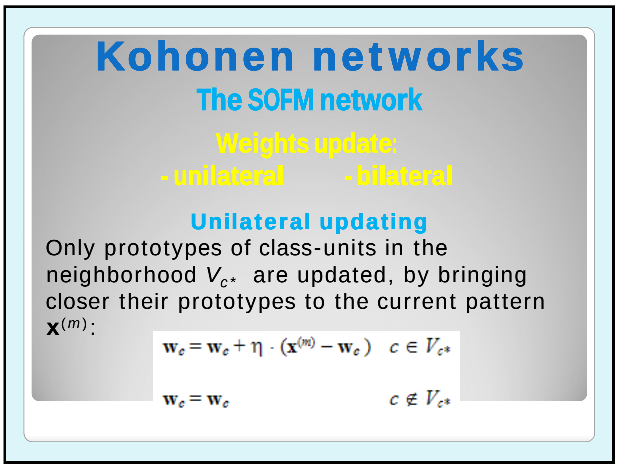
Continuous neighborhoods

A neighborhood Gaussian – shape function is used:

$$V_c = e^{-\rho_c^2/2\sigma^2}$$

where σ - *standard deviation* , ρ_c - distance between class-unit c and the winner class-unit c*:

$$\rho_c = |P_c - P_{c*}|$$



Kohonen networks

The SOFM network

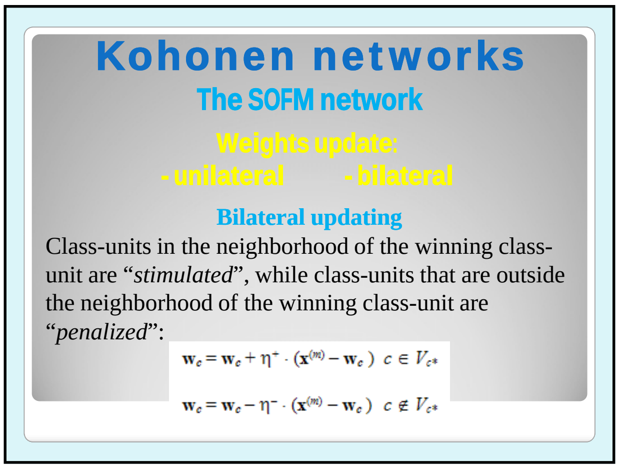
Weights update:
- unilateral - bilateral

Unilateral updating

Only prototypes of class-units in the neighborhood V_{c*} are updated, by bringing closer their prototypes to the current pattern $\mathbf{x}^{(m)}$:

$$\mathbf{w}_c = \mathbf{w}_c + \eta \cdot (\mathbf{x}^{(m)} - \mathbf{w}_c) \quad c \in V_{c*}$$

$$\mathbf{w}_c = \mathbf{w}_c \quad c \notin V_{c*}$$



Kohonen networks

The SOFM network

Weights update:

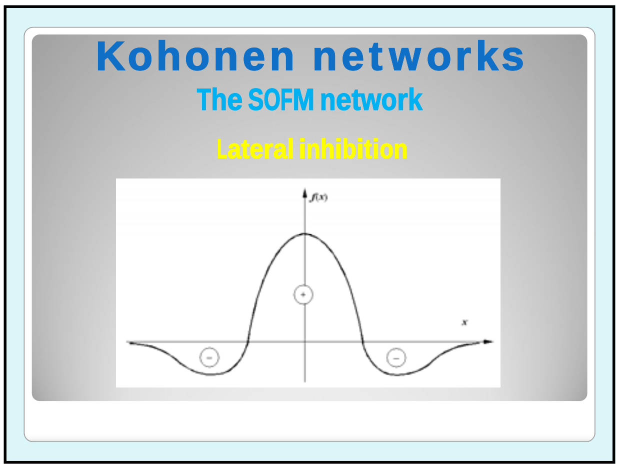
- unilateral
- bilateral

Bilateral updating

Class-units in the neighborhood of the winning class-unit are "*stimulated*", while class-units that are outside the neighborhood of the winning class-unit are "*penalized*":

$$\mathbf{w}_c = \mathbf{w}_c + \eta^+ \cdot (\mathbf{x}^{(m)} - \mathbf{w}_c) \quad c \in V_{c*}$$

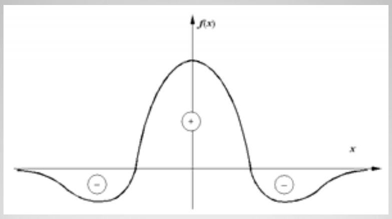
$$\mathbf{w}_c = \mathbf{w}_c - \eta^- \cdot (\mathbf{x}^{(m)} - \mathbf{w}_c) \quad c \notin V_{c*}$$

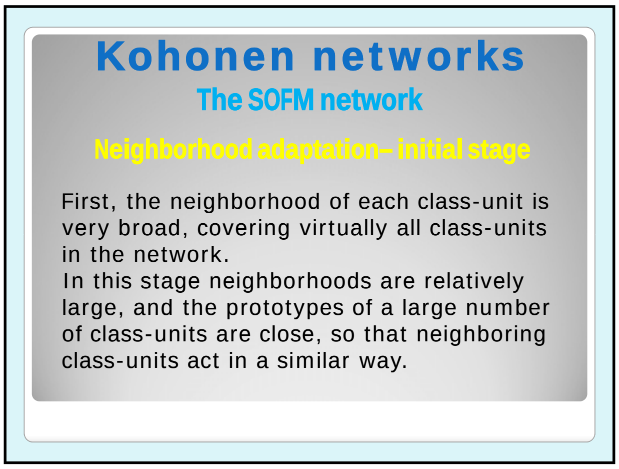


Kohonen networks

The SOFM network

Lateral inhibition



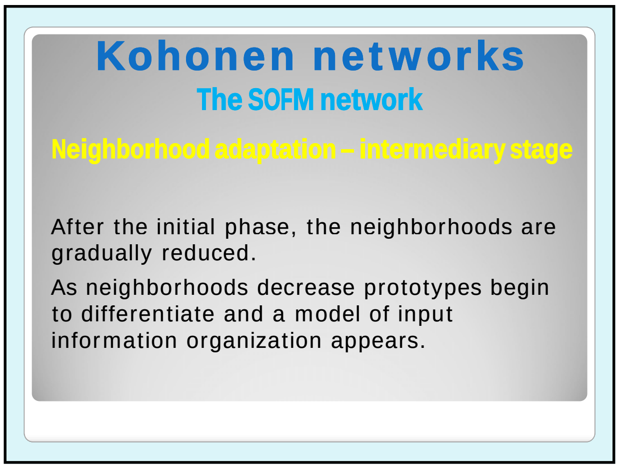


Kohonen networks

The SOFM network

Neighborhood adaptation– initial stage

First, the neighborhood of each class-unit is very broad, covering virtually all class-units in the network.
In this stage neighborhoods are relatively large, and the prototypes of a large number of class-units are close, so that neighboring class-units act in a similar way.



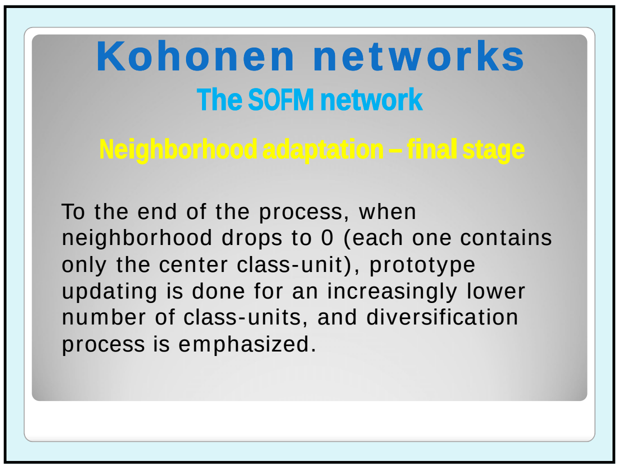
Kohonen networks

The SOFM network

Neighborhood adaptation – intermediary stage

After the initial phase, the neighborhoods are gradually reduced.

As neighborhoods decrease prototypes begin to differentiate and a model of input information organization appears.

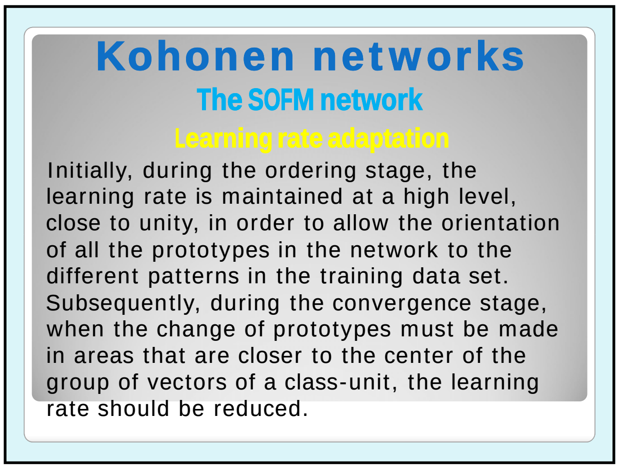


Kohonen networks

The SOFM network

Neighborhood adaptation – final stage

To the end of the process, when neighborhood drops to 0 (each one contains only the center class-unit), prototype updating is done for an increasingly lower number of class-units, and diversification process is emphasized.

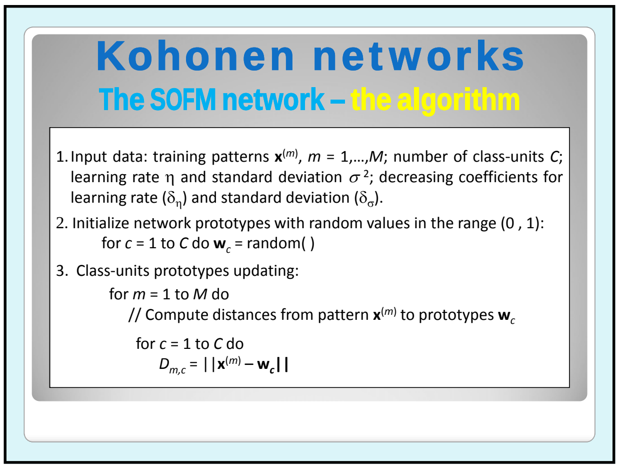


Kohonen networks

The SOFM network

Learning rate adaptation

Initially, during the ordering stage, the learning rate is maintained at a high level, close to unity, in order to allow the orientation of all the prototypes in the network to the different patterns in the training data set. Subsequently, during the convergence stage, when the change of prototypes must be made in areas that are closer to the center of the group of vectors of a class-unit, the learning rate should be reduced.



Kohonen networks

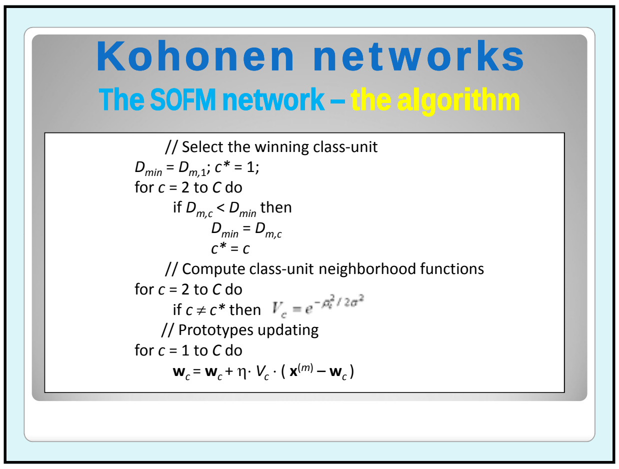
The SOFM network – the algorithm

1. Input data: training patterns $\mathbf{x}^{(m)}$, $m = 1,\ldots,M$; number of class-units C; learning rate η and standard deviation σ^2; decreasing coefficients for learning rate (δ_η) and standard deviation (δ_σ).
2. Initialize network prototypes with random values in the range (0 , 1):

```
        for $c = 1$ to $C$ do $\mathbf{w}_c$ = random( )
```

3. Class-units prototypes updating:

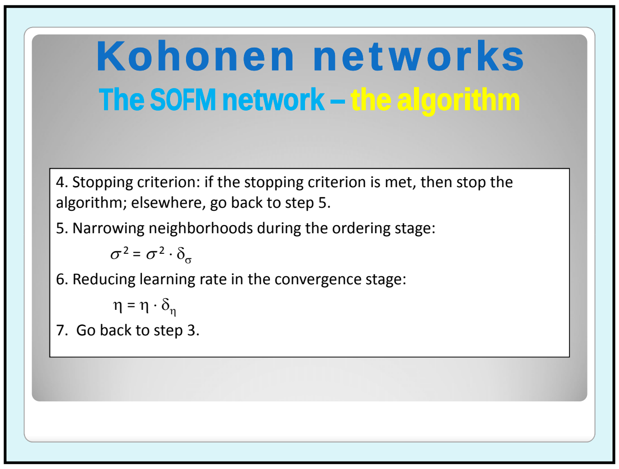
```
        for $m = 1$ to $M$ do
            // Compute distances from pattern $\mathbf{x}^{(m)}$ to prototypes $\mathbf{w}_c$
            for $c = 1$ to $C$ do
                $D_{m,c} = ||\mathbf{x}^{(m)} - \mathbf{w}_c||$
```



Kohonen networks

The SOFM network – the algorithm

```
            // Select the winning class-unit
        $D_{min} = D_{m,1}$; $c^* = 1$;
        for $c = 2$ to $C$ do
            if $D_{m,c} < D_{min}$ then
                $D_{min} = D_{m,c}$
                $c^* = c$
            // Compute class-unit neighborhood functions
        for $c = 2$ to $C$ do
            if $c \neq c^*$ then $V_c = e^{-\rho_c^2/2\sigma^2}$
            // Prototypes updating
        for $c = 1$ to $C$ do
            $\mathbf{w}_c = \mathbf{w}_c + \eta \cdot V_c \cdot (\mathbf{x}^{(m)} - \mathbf{w}_c)$
```



Kohonen networks

The SOFM network – the algorithm

4. Stopping criterion: if the stopping criterion is met, then stop the algorithm; elsewhere, go back to step 5.
5. Narrowing neighborhoods during the ordering stage:

$$\sigma^2 = \sigma^2 \cdot \delta_\sigma$$

6. Reducing learning rate in the convergence stage:

$$\eta = \eta \cdot \delta_\eta$$

7. Go back to step 3.