Lecture 5:

Self–Organising Maps

Dr. Roman V Belavkin

Middlesex University

BIS4435

Euclidean Space and Euclidean Distance (Examples)

The Clustering Problem

SOM Architecture and Principles

Training Procedure

Contextual Maps

Applications of SOM

HISTORICAL BACKGROUND

1960s Vector quantisation problems studied by mathematicians (Glienn, 1964; Stratonowitch, 1966).

1973 von der Malsburg did the first computer simulation demonstrating self–organisation.

1976 Willshaw and von der Malsburg suggested the idea of SOM.

1980s Works by Kohonen further developed and studied computational algorithms for SOM.

EUCLIDEAN SPACE

- Points in Euclidean space have coordinates, represented by real numbers **R** (e.g. x, y and z or x_1, x_2 and x_3).

EUCLIDEAN SPACE

- Points in Euclidean space have coordinates, represented by real numbers $\mathbf{R}$ (e.g. x, y and z or x_1, x_2 and x_3).
- We denote m-dimensional space by $\mathbf{R}^m$.

EUCLIDEAN SPACE

- Points in Euclidean space have coordinates, represented by real numbers $\mathbf{R}$ (e.g. x, y and z or x_1, x_2 and x_3).
- We denote m-dimensional space by $\mathbf{R}^m$.
- Every point in $\mathbf{R}^m$ is defined by m coordinates:

$$\{x_1, \ldots, x_m\}$$

or by an m–dimensional vector

$$\mathbf{x} = (x_1, \ldots, x_m)$$

.

EUCLIDEAN SPACE

- Points in Euclidean space have coordinates, represented by real numbers $\mathbf{R}$ (e.g. x, y and z or x_1, x_2 and x_3).
- We denote m-dimensional space by $\mathbf{R}^m$.
- Every point in $\mathbf{R}^m$ is defined by m coordinates:

$$\{x_1, \ldots, x_m\}$$

or by an m–dimensional vector

$$\mathbf{x} = (x_1, \ldots, x_m)$$

.

- How can we compute the distance between different points in such a space?

EXAMPLES

Example

In $\mathbf{R}^1$ (one–dimensional space or a line) points are represented by just one number, such as $\mathbf{a} = (2)$ or $\mathbf{b} = (-1)$.

EXAMPLES

Example

In $\mathbf{R}^1$ (one–dimensional space or a line) points are represented by just one number, such as $\mathbf{a} = (2)$ or $\mathbf{b} = (-1)$.

Example

In $\mathbf{R}^3$ (three–dimensional space) points are represented by three coordinates x, y and z (or x_1, x_2 and x_3), such as $\mathbf{a} = (2, -1, 3)$.

EUCLIDEAN DISTANCE

The distance $\rho(\mathbf{a}, \mathbf{b})$ between two points

$$\mathbf{a} = (a_1, \ldots, a_m) \quad \text{and} \quad \mathbf{b} = (b_1, \ldots, b_m)$$

in Euclidean space $\mathbf{R}^m$ is calculated as:

$$\begin{aligned} \|\mathbf{a} - \mathbf{b}\| &= \sqrt{\sum_{i=1}^{m} (a_i - b_i)^2} \\ &= \sqrt{(a_1 - b_1)^2 + \cdots + (a_m - b_m)^2} \end{aligned}$$

EUCLIDEAN DISTANCE

The distance $\rho(\mathbf{a}, \mathbf{b})$ between two points

$$\mathbf{a} = (a_1, \ldots, a_m) \quad \text{and} \quad \mathbf{b} = (b_1, \ldots, b_m)$$

in Euclidean space $\mathbf{R}^m$ is calculated as:

$$\begin{aligned} \|\mathbf{a} - \mathbf{b}\| &= \sqrt{\sum_{i=1}^{m} (a_i - b_i)^2} \\ &= \sqrt{(a_1 - b_1)^2 + \cdots + (a_m - b_m)^2} \end{aligned}$$

Remark

Note that Euclidean distance is

1. *Positive:* $\rho(\mathbf{a}, \mathbf{b}) \geq 0$, *and* $\rho(\mathbf{a}, \mathbf{b}) = 0$ *iff* $\mathbf{a} = \mathbf{b}$
2. *Symmetric:* $\rho(\mathbf{a}, \mathbf{b}) = \rho(\mathbf{b}, \mathbf{a})$.
3. *Triangle inequality:* $\rho(\mathbf{a}, \mathbf{c}) \leq \rho(\mathbf{a}, \mathbf{b}) + \rho(\mathbf{b}, \mathbf{c})$

Such functions are called metrics, and sets with metrics are called metric spaces.

EXAMPLES

Example

Let $\mathbf{a} = (2)$ and $\mathbf{b} = (-1)$ in $\mathbf{R}^1$. Then

EXAMPLES

Example

Let $\mathbf{a} = (2)$ and $\mathbf{b} = (-1)$ in $\mathbf{R}^1$. Then

$$\|\mathbf{a} - \mathbf{b}\| = \sqrt{(2+1)^2} = 3$$

EXAMPLES

Example

Let $\mathbf{a} = (2)$ and $\mathbf{b} = (-1)$ in $\mathbf{R}^1$. Then

$$\|\mathbf{a} - \mathbf{b}\| = \sqrt{(2+1)^2} = 3$$

Example

Let $\mathbf{a} = (2, -3)$ and $\mathbf{b} = (4, 1)$ in $\mathbf{R}^2$.
Then

EXAMPLES

Example

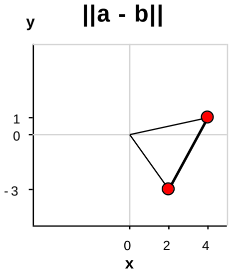
Let $\mathbf{a} = (2)$ and $\mathbf{b} = (-1)$ in $\mathbf{R}^1$. Then

$$\|\mathbf{a} - \mathbf{b}\| = \sqrt{(2+1)^2} = 3$$

Example

Let $\mathbf{a} = (2, -3)$ and $\mathbf{b} = (4, 1)$ in $\mathbf{R}^2$. Then

$$\begin{aligned} \|\mathbf{a} - \mathbf{b}\| &= \sqrt{(2-4)^2 + (-3-1)^2} \\ &= \sqrt{20} \approx 4.47 \end{aligned}$$



MULTIDIMENSIONAL DATA IN BUSINESS

- A bank gathered information about its customers:

Case:	Age	Gender	Income ($K p.m.)	Expenses ($K p.m.)	Home owner	Credit score
1	21	0	2	1	0	3
2	18	1	1	2	0	1
3	50	1	6	2	1	5
4	23	0	3	1	1	4
5	40	1	3	2	0	2

MULTIDIMENSIONAL DATA IN BUSINESS

- A bank gathered information about its customers:

Case:	Age	Gender	Income ($K p.m.)	Expenses ($K p.m.)	Home owner	Credit score
1	21	0	2	1	0	3
2	18	1	1	2	0	1
3	50	1	6	2	1	5
4	23	0	3	1	1	4
5	40	1	3	2	0	2

- We may consider each variable (age, gender, income, etc) as a coordinate x_i and each case as a point in an m–dimensional space.

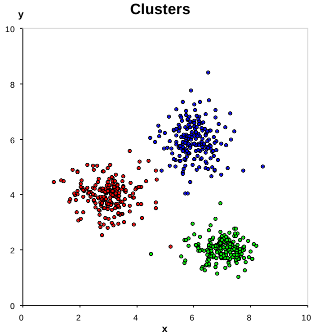
MULTIDIMENSIONAL DATA IN BUSINESS

- A bank gathered information about its customers:

Case:	Age	Gender	Income ($K p.m.)	Expenses ($K p.m.)	Home owner	Credit score
1	21	0	2	1	0	3
2	18	1	1	2	0	1
3	50	1	6	2	1	5
4	23	0	3	1	1	4
5	40	1	3	2	0	2

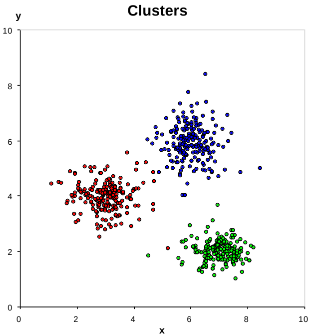
- We may consider each variable (age, gender, income, etc) as a coordinate x_i and each case as a point in an m–dimensional space.
- How far should be similar cases from each other?

CLUSTERS



- **Clusters** are groups of points close to each other.
- One of the main goals of **multivariate analysis** is to find clusters of points.

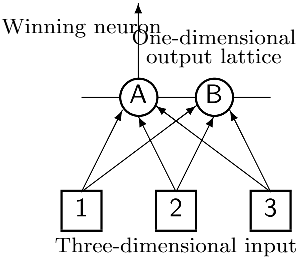
CLUSTERS

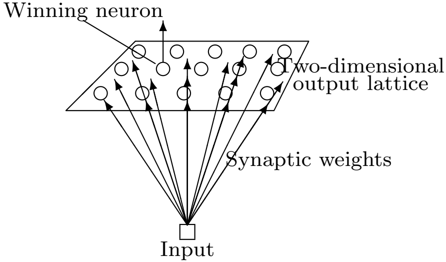


- **Clusters** are groups of points close to each other.
- One of the main goals of **multivariate analysis** is to find clusters of points.
- 'Similar' customers would have small Euclidean distance between them and would belong to the same group (cluster).

SOM ARCHITECTURE

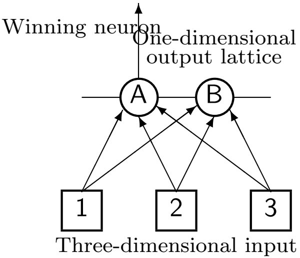
- SOM uses a single layer network of neurons **competing** with each other, so that only one neuron can fire at a time.

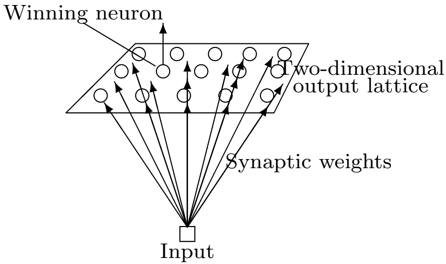




SOM ARCHITECTURE

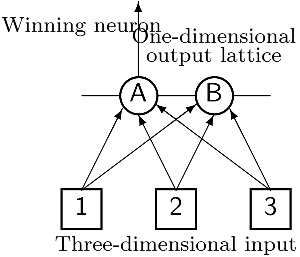
- SOM uses a single layer network of neurons **competing** with each other, so that only one neuron can fire at a time.
- The algorithm consists of three phases: **competition**, **cooperation** and **adaptation**.

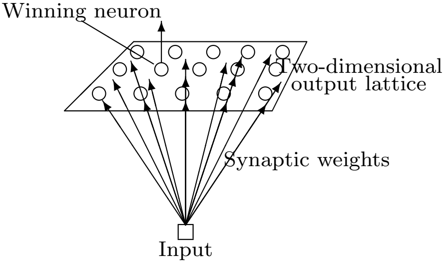




INPUT TOPOLOGY

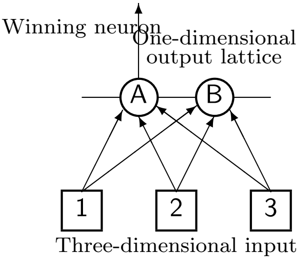
- The input layer has m nodes, so the input pattern is point $\mathbf{x} = (x_1, \ldots, x_m)$ in m–dimensional input space $\mathbf{R}^m$.

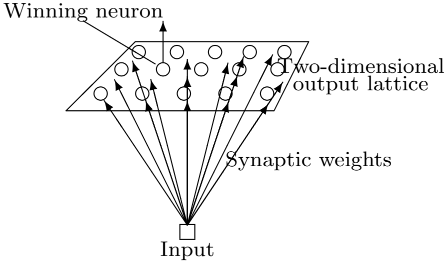




INPUT TOPOLOGY

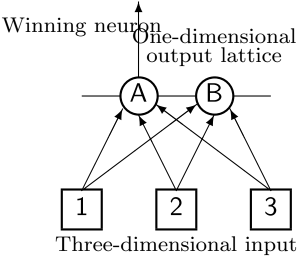
- The input layer has m nodes, so the input pattern is point $\mathbf{x} = (x_1, \ldots, x_m)$ in m–dimensional input space $\mathbf{R}^m$.
- Each neuron j has m weights, so the weights represent point $\mathbf{w}_j = (w_{1j}, \ldots, w_{mj})$ in the same input space $\mathbf{R}^m$.

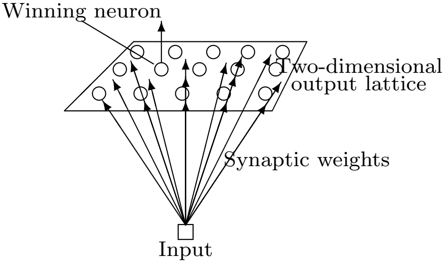




INPUT TOPOLOGY

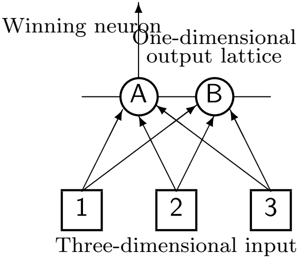
- The input layer has m nodes, so the input pattern is point $\mathbf{x} = (x_1, \ldots, x_m)$ in m–dimensional input space $\mathbf{R}^m$.
- Each neuron j has m weights, so the weights represent point $\mathbf{w}_j = (w_{1j}, \ldots, w_{mj})$ in the same input space $\mathbf{R}^m$.
- 'Closeness' (neighbourhood) in the m-dimensional input space $\mathbf{R}^m$ is computed by the Euclidean distance $\rho(\mathbf{x}, \mathbf{w})$.

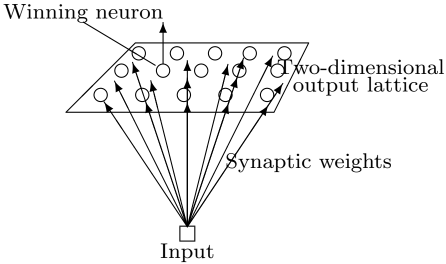




OUTPUT TOPOLOGY

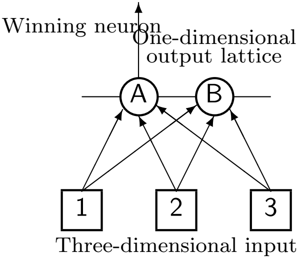
- The outputs of the nodes of an SOM are arranged in a lattice, which is usually a 1 or 2-dimensional space (i.e. a line $\mathbf{R}^1$ or a plane $\mathbf{R}^2$).

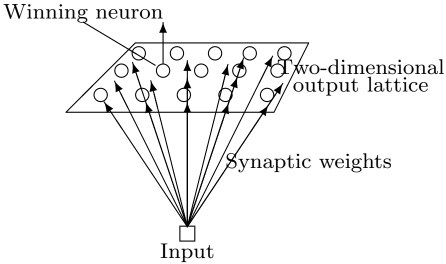




OUTPUT TOPOLOGY

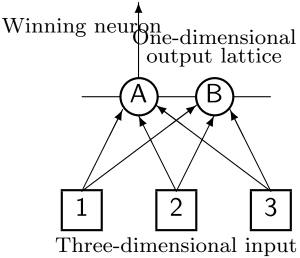
- The outputs of the nodes of an SOM are arranged in a lattice, which is usually a 1 or 2-dimensional space (i.e. a line $\mathbf{R}^1$ or a plane $\mathbf{R}^2$).
- The lattice represents the output space, where different points correspond to the outputs of different nodes i and j.

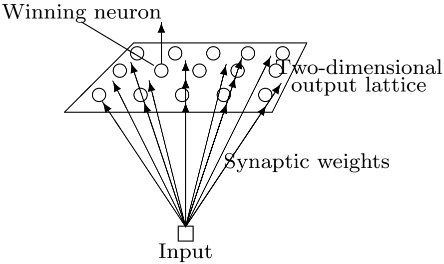




OUTPUT TOPOLOGY

- The outputs of the nodes of an SOM are arranged in a lattice, which is usually a 1 or 2-dimensional space (i.e. a line $\mathbf{R}^1$ or a plane $\mathbf{R}^2$).
- The lattice represents the output space, where different points correspond to the outputs of different nodes i and j.
- 'Closeness' (neighbourhood) in this 1 or 2-dimensional output space is computed by abother distance function $d(ij)$.





TOPOLOGICAL MAPPING

- SOM $f : \mathbf{R}^m \rightarrow \mathbf{R}^n$ maps m-dimensional input space $\mathbf{R}^m$ onto the n-dimensonal output space $\mathbf{R}^n$ (usually $m \gg n$).

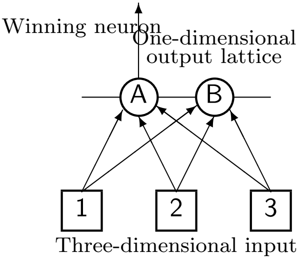
TOPOLOGICAL MAPPING

- SOM $f : \mathbf{R}^m \rightarrow \mathbf{R}^n$ maps m-dimensional input space $\mathbf{R}^m$ onto the n-dimensonal output space $\mathbf{R}^n$ (usually $m \gg n$).
- SOM is designed to preserve topology so that 'closeness' in the input space corresponds to 'closeness' in the output space (i.e. in the lattice), and vice verse.

TOPOLOGICAL MAPPING

- SOM $f : \mathbf{R}^m \rightarrow \mathbf{R}^n$ maps m-dimensional input space $\mathbf{R}^m$ onto the n-dimensonal output space $\mathbf{R}^n$ (usually $m \gg n$).
- SOM is designed to preserve topology so that 'closeness' in the input space corresponds to 'closeness' in the output space (i.e. in the lattice), and vice verse.

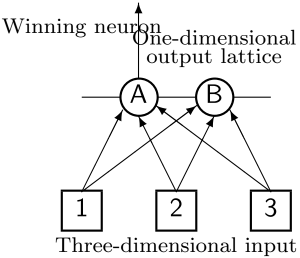
$f : \mathbf{R}^3 \rightarrow \mathbf{R}^1$



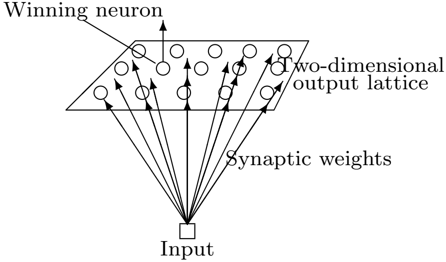
TOPOLOGICAL MAPPING

- SOM $f : \mathbf{R}^m \rightarrow \mathbf{R}^n$ maps m-dimensional input space $\mathbf{R}^m$ onto the n-dimensonal output space $\mathbf{R}^n$ (usually $m \gg n$).
- SOM is designed to preserve topology so that 'closeness' in the input space corresponds to 'closeness' in the output space (i.e. in the lattice), and vice verse.

$f : \mathbf{R}^3 \rightarrow \mathbf{R}^1$



$f : \mathbf{R}^1 \rightarrow \mathbf{R}^2$



COMPETITION

- Input pattern $\mathbf{x} = (x_1, \ldots, x_m)$ is compared with each weight pattern $\mathbf{w}_j = (w_{1j}, \ldots, w_{mj})$.

COMPETITION

- Input pattern $\mathbf{x} = (x_1, \ldots, x_m)$ is compared with each weight pattern $\mathbf{w}_j = (w_{1j}, \ldots, w_{mj})$.
- The **winner** is the node whose weight $\mathbf{w}_j$ is the closest to the input $\mathbf{x}$ in terms of Euclidean distance:

$$\begin{aligned}
\|\mathbf{x} - \mathbf{w}_1\| &= \sqrt{(x_1 - w_{11})^2 + \cdots + (x_m - w_{m1})^2} \\
&\vdots \\
\|\mathbf{x} - \mathbf{w}_n\| &= \sqrt{(x_1 - w_{1n})^2 + \cdots + (x_m - w_{mn})^2}
\end{aligned}$$

COMPETITION

- Input pattern $\mathbf{x} = (x_1, \ldots, x_m)$ is compared with each weight pattern $\mathbf{w}_j = (w_{1j}, \ldots, w_{mj})$.
- The **winner** is the node whose weight $\mathbf{w}_j$ is the closest to the input $\mathbf{x}$ in terms of Euclidean distance:

$$\begin{aligned}
\|\mathbf{x} - \mathbf{w}_1\| &= \sqrt{(x_1 - w_{11})^2 + \cdots + (x_m - w_{m1})^2} \\
&\vdots \\
\|\mathbf{x} - \mathbf{w}_n\| &= \sqrt{(x_1 - w_{1n})^2 + \cdots + (x_m - w_{mn})^2}
\end{aligned}$$

- Thus, nodes 'compete' in the sense which of the nodes $\mathbf{w}_j$ is more 'similar' to a give input pattern $\mathbf{x}$.

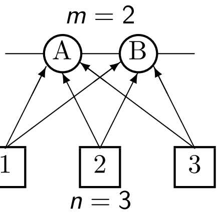
Example

Consider SOM with three inputs and two output nodes (A and B). Let

$$\mathbf{w}_A = (2, -1, 3)\,, \quad \mathbf{w}_B = (-2, 0, 1)$$

Find which node is the winner for the input

$$\mathbf{x} = (1, -2, 2)$$



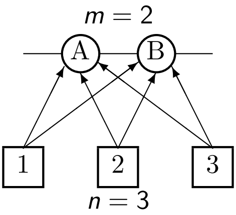
Example

Consider SOM with three inputs and two output nodes (A and B). Let

$$\mathbf{w}_A = (2, -1, 3)\,, \quad \mathbf{w}_B = (-2, 0, 1)$$

Find which node is the winner for the input

$$\mathbf{x} = (1, -2, 2)$$



$$\begin{aligned}
\|\mathbf{x} - \mathbf{w}_A\| &= \sqrt{(1-2)^2 + (-2+1)^2 + (2-3)^2} = \sqrt{3} \\
\|\mathbf{x} - \mathbf{w}_B\| &= \sqrt{(1+2)^2 + (-2-0)^2 + (2-1)^2} = \sqrt{14}
\end{aligned}$$

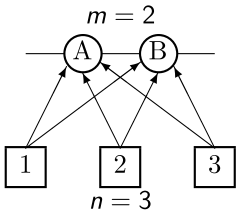
Example

Consider SOM with three inputs and two output nodes (A and B). Let

$$\mathbf{w}_A = (2, -1, 3)\,, \quad \mathbf{w}_B = (-2, 0, 1)$$

Find which node is the winner for the input

$$\mathbf{x} = (1, -2, 2)$$



$$\begin{aligned} \|\mathbf{x} - \mathbf{w}_A\| &= \sqrt{(1-2)^2 + (-2+1)^2 + (2-3)^2} = \sqrt{3} \\ \|\mathbf{x} - \mathbf{w}_B\| &= \sqrt{(1+2)^2 + (-2-0)^2 + (2-1)^2} = \sqrt{14} \end{aligned}$$

- Node A is the winner because it is 'closer' ($\sqrt{3} < \sqrt{14}$)

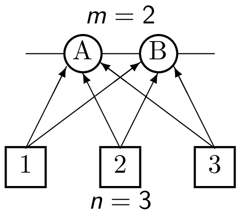
Example

Consider SOM with three inputs and two output nodes (A and B). Let

$$\mathbf{w}_A = (2, -1, 3), \quad \mathbf{w}_B = (-2, 0, 1)$$

Find which node is the winner for the input

$$\mathbf{x} = (1, -2, 2)$$



$$\|\mathbf{x} - \mathbf{w}_A\| = \sqrt{(1-2)^2 + (-2+1)^2 + (2-3)^2} = \sqrt{3}$$

$$\|\mathbf{x} - \mathbf{w}_B\| = \sqrt{(1+2)^2 + (-2-0)^2 + (2-1)^2} = \sqrt{14}$$

- Node A is the winner because it is 'closer' ($\sqrt{3} < \sqrt{14}$)
- What if $\mathbf{x} = (-1, -2, 0)$?

ADAPTATION

- After the input $\mathbf{x}$ has been presented to SOM, the weights of *all* nodes are adapted (adjusted) so that they become more 'similar' to the input $\mathbf{x}$.

ADAPTATION

- After the input $\mathbf{x}$ has been presented to SOM, the weights of *all* nodes are adapted (adjusted) so that they become more 'similar' to the input $\mathbf{x}$.
- The adaptation formula for node j is:

$$\mathbf{w}_j^{\mathrm{new}} = \mathbf{w}_j^{\mathrm{old}} + \alpha\, h_{ij} \left[\mathbf{x} - \mathbf{w}_j^{\mathrm{old}} \right] \ ,$$

  where

  - $\mathbf{w}_j$ is the weight vector of node j;
  - α is the learning rate coefficient;
  - h_{ij} is the neighbourhood of node j with respect to the winner i.

ADAPTATION (cont.)

To understand better the adaptation formula, let us check how the weights change for different values of α and h_{ij}.

$$\mathbf{w}_j^{\mathrm{new}} = \mathbf{w}_j^{\mathrm{old}} + \alpha\, h_{ij} \left[\mathbf{x} - \mathbf{w}_j^{\mathrm{old}}\right] \ ,$$

ADAPTATION (cont.)

To understand better the adaptation formula, let us check how the weights change for different values of α and h_{ij}.

$$\mathbf{w}_j^{\mathrm{new}} = \mathbf{w}_j^{\mathrm{old}} + \alpha\, h_{ij} \left[\mathbf{x} - \mathbf{w}_j^{\mathrm{old}} \right] \ ,$$

- Suppose $\alpha = 0$ or $h_{ij} = 0$. Then

  $$\mathbf{w}_j^{\mathrm{new}} = \mathbf{w}_j^{\mathrm{old}} + 0 \cdot 0 \left[\mathbf{x} - \mathbf{w}_j^{\mathrm{old}} \right] = \mathbf{w}_j^{\mathrm{old}}$$

  The weight does not change ($\mathbf{w}_j^{\mathrm{new}} = \mathbf{w}_j^{\mathrm{old}}$).

ADAPTATION (cont.)

To understand better the adaptation formula, let us check how the weights change for different values of α and h_{ij}.

$$\mathbf{w}_j^{\mathrm{new}} = \mathbf{w}_j^{\mathrm{old}} + \alpha\, h_{ij} \left[\mathbf{x} - \mathbf{w}_j^{\mathrm{old}}\right] \ ,$$

- Suppose $\alpha = 0$ or $h_{ij} = 0$. Then

  $$\mathbf{w}_j^{\mathrm{new}} = \mathbf{w}_j^{\mathrm{old}} + 0 \cdot 0 \left[\mathbf{x} - \mathbf{w}_j^{\mathrm{old}}\right] = \mathbf{w}_j^{\mathrm{old}}$$

  The weight does not change ($\mathbf{w}_j^{\mathrm{new}} = \mathbf{w}_j^{\mathrm{old}}$).
- Suppose $h_{ij} = 1$ and $\alpha = 1$. Then

  $$\mathbf{w}_j^{\mathrm{new}} = \mathbf{w}_j^{\mathrm{old}} + \mathbf{x} - \mathbf{w}_j^{\mathrm{old}} = \mathbf{x}$$

  The new weight is equal to the input ($\mathbf{w}_j^{\mathrm{new}} = \mathbf{x}$).

COOPERATION

Although weights of *all* nodes are adapted, they do not adapt equally. Adaptation depends on how close the nodes are from the winner in the output lattice.

COOPERATION

Although weights of *all* nodes are adapted, they do not adapt equally. Adaptation depends on how close the nodes are from the winner in the output lattice.

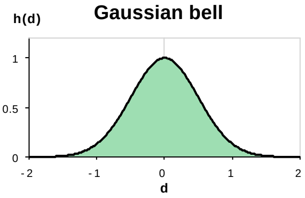
- If the winner is node i, then the level of adaptation for node j is defined by the neighbourhood function $h = h_{ij}(d_{ij})$, where $d(i, j)$ is the distance in the lattice.

COOPERATION

Although weights of *all* nodes are adapted, they do not adapt equally. Adaptation depends on how close the nodes are from the winner in the output lattice.

- If the winner is node i, then the level of adaptation for node j is defined by the neighbourhood function $h = h_{ij}(d_{ij})$, where $d(i, j)$ is the distance in the lattice.
- The neighbourhood is defines in such a way that it is smaller as the distance $d(i, j)$ gets larger. For example, the Gaussian bell function

$$h_{ij}(d) = e^{-\frac{d^2}{2\sigma^2}}$$

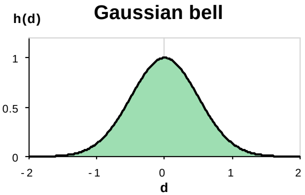


COOPERATION

Although weights of *all* nodes are adapted, they do not adapt equally. Adaptation depends on how close the nodes are from the winner in the output lattice.

- If the winner is node i, then the level of adaptation for node j is defined by the neighbourhood function $h = h_{ij}(d_{ij})$, where $d(i, j)$ is the distance in the lattice.
- The neighbourhood is defines in such a way that it is smaller as the distance $d(i, j)$ gets larger. For example, the Gaussian bell function

$$h_{ij}(d) = e^{-\frac{d^2}{2\sigma^2}}$$



- The winner 'helps' mostly its neighbours to adapt. Note also that the winner is adapted more than any other node (i.e. because $d(i, i) = 0$).

Example

Let $\alpha = 0.5$ and $h = 1$, and let us adapt the winning node A from previous example:

$$\mathbf{w}_A = (2, -1, 3)\,, \quad \mathbf{x} = (1, -2, 2)$$

We use adaptation formula: $\mathbf{w}_j^{\mathrm{new}} = \mathbf{w}_j^{\mathrm{old}} + \alpha\, h_{ij}[\mathbf{x} - \mathbf{w}_j^{\mathrm{old}}]$

Example

Let $\alpha = 0.5$ and $h = 1$, and let us adapt the winning node A from previous example:

$$\mathbf{w}_A = (2, -1, 3)\,, \quad \mathbf{x} = (1, -2, 2)$$

We use adaptation formula: $\mathbf{w}_j^{\text{new}} = \mathbf{w}_j^{\text{old}} + \alpha\, h_{ij}[\mathbf{x} - \mathbf{w}_j^{\text{old}}]$

$$\begin{aligned}
\mathbf{w}_A &= \begin{pmatrix} 2 \\ -1 \\ 3 \end{pmatrix} + 0.5 \cdot 1 \cdot \left[\begin{pmatrix} 1 \\ -2 \\ 2 \end{pmatrix} - \begin{pmatrix} 2 \\ -1 \\ 3 \end{pmatrix} \right] \\
&= \begin{pmatrix} 2 \\ -1 \\ 3 \end{pmatrix} + 0.5 \cdot \begin{pmatrix} -1 \\ -1 \\ -1 \end{pmatrix} = \begin{pmatrix} 1.5 \\ -1.5 \\ 2.5 \end{pmatrix}
\end{aligned}$$

TRAINING PROCEDURE

- Initially set all the weights to some random values

TRAINING PROCEDURE

- Initially set all the weights to some random values
- Repeat

- Until the network stabilises

TRAINING PROCEDURE

- Initially set all the weights to some random values
- Repeat
  1. Feed an input pattern from the set of data
- Until the network stabilises

TRAINING PROCEDURE

- Initially set all the weights to some random values
- Repeat
    1. Feed an input pattern from the set of data
    2. Find the winner

- Until the network stabilises

TRAINING PROCEDURE

- Initially set all the weights to some random values
- Repeat
  1. Feed an input pattern from the set of data
  2. Find the winner
  3. Adapt the weights of the winner and its neighbours
- Until the network stabilises

WHAT SHOULD BE THE RESULT?

- Initially, there is no relation between closeness (similarity) in the input space and closeness in the output lattice.

WHAT SHOULD BE THE RESULT?

- Initially, there is no relation between closeness (similarity) in the input space and closeness in the output lattice.
- After training, nodes close to each other in the lattice correspond to points close to each other in the input space.

WHAT SHOULD BE THE RESULT?

- Initially, there is no relation between closeness (similarity) in the input space and closeness in the output lattice.
- After training, nodes close to each other in the lattice correspond to points close to each other in the input space.
- The number of input dimensions (m) is usually very large (e.g. $m = 100$), so we cannot see the clusters directly.

WHAT SHOULD BE THE RESULT?

- Initially, there is no relation between closeness (similarity) in the input space and closeness in the output lattice.
- After training, nodes close to each other in the lattice correspond to points close to each other in the input space.
- The number of input dimensions (m) is usually very large (e.g. $m = 100$), so we cannot see the clusters directly.
- The output lattice is usually one or two dimensional, so we can visualise and ‘see’ the clusters.

EXAMPLE OF SOM

	dove	hen	duck	goose	owl	hawk	eagle	fox	dog	wolf	cat	tiger	lion	horse	zebra	cow
small	1	1	1	1	1	1	0	0	0	0	1	0	0	0	0	0
medium	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0
big	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1
2 legs	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
4 legs	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
hair	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
hooves	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
mane	0	0	0	0	0	0	0	0	0	1	0	0	1	1	1	0
feathers	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
hunt	0	0	0	0	1	1	1	1	0	1	1	1	1	0	0	0
run	0	0	0	0	0	0	0	0	1	1	0	1	1	1	1	0
fly	1	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0
swim	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0

FEATURE MAP

dog	·	·	fox	·	·	cat	·	·	eagle
·	·	·	·	·	·	·	·	·	·
·	·	·	·	·	·	·	·	·	owl
·	·	·	·	·	·	tiger	·	·	·
wolf	·	·	·	·	·	·	·	·	hawk
·	·	·	lion	·	·	·	·	·	·
·	·	·	·	·	·	·	·	·	dove
horse	·	·	·	·	·	·	hen	·	·
·	·	·	·	cow	·	·	·	·	goose
zebra	·	·	·	·	·	·	duck	·	·

USEFUL PROPERTIES OF SOM

- Reducing dimensions (Indeed, SOM is a map $f : \mathbf{R}^m \longrightarrow \mathbf{R}^n$)

USEFUL PROPERTIES OF SOM

- Reducing dimensions (Indeed, SOM is a map $f : \mathbf{R}^m \longrightarrow \mathbf{R}^n$)
- Visualisation of clusters

USEFUL PROPERTIES OF SOM

- Reducing dimensions (Indeed, SOM is a map $f : \mathbf{R}^m \longrightarrow \mathbf{R}^n$)
- Visualisation of clusters
- Ordered display (preserving the topology)

USEFUL PROPERTIES OF SOM

- Reducing dimensions (Indeed, SOM is a map $f : \mathbf{R}^m \longrightarrow \mathbf{R}^n$)
- Visualisation of clusters
- Ordered display (preserving the topology)
- Handles missing data

USEFUL PROPERTIES OF SOM

- Reducing dimensions (Indeed, SOM is a map $f : \mathbf{R}^m \longrightarrow \mathbf{R}^n$)
- Visualisation of clusters
- Ordered display (preserving the topology)
- Handles missing data
- The learning algorithm is unsupervised.

APPLICATIONS OF SOM IN BUSINESS

- SOM can be very useful during the intelligence phase of decision making. It helps to visualise very complex and highly–dimensional data.

APPLICATIONS OF SOM IN BUSINESS

- SOM can be very useful during the intelligence phase of decision making. It helps to visualise very complex and highly–dimensional data.
- Visualisation of multi–dimensional data can be used for presentations and reports.

APPLICATIONS OF SOM IN BUSINESS

- SOM can be very useful during the intelligence phase of decision making. It helps to visualise very complex and highly–dimensional data.
- Visualisation of multi–dimensional data can be used for presentations and reports.
- Identifying clusters in the data (e.g. typical groups of customers) can help in optimising busibess operations.

APPLICATIONS OF SOM IN BUSINESS

- SOM can be very useful during the intelligence phase of decision making. It helps to visualise very complex and highly–dimensional data.
- Visualisation of multi–dimensional data can be used for presentations and reports.
- Identifying clusters in the data (e.g. typical groups of customers) can help in optimising busibess operations.
- Can be used for pattern recognition (e.g. to identify credit–card fraud, errors in data, etc).