Self Organizing Maps

History

- Ideas first introduced by C.von der Malsburg (1973), developed and refined by T.Kohonen (Finland) (1982)
- Neural network algorithm using unsupervised ***competitive learning***.
  - The output neurons compete amongst themselves to be activated, with the result that only one is activated at any one time.
  - This activated neuron is called a ***winner-takes-all*** neuron or simply the winning neuron.
- They are called «Self-organizing» because no supervision is required.
  - Neurons are forced to organise themselves.

Topographic Maps

- Neurobiological studies indicate that different sensory inputs (motor, visual, auditory, etc.) are mapped onto corresponding areas of the cerebral cortex in an orderly fashion.
- This form of map, known as a ***topographic map***, has two important properties:
  - At each stage of representation, or processing, each piece of incoming information is kept in its proper context/ neighbourhood.
  - Neurons dealing with closely related pieces of information are kept close together so that they can interact via short synaptic connections.

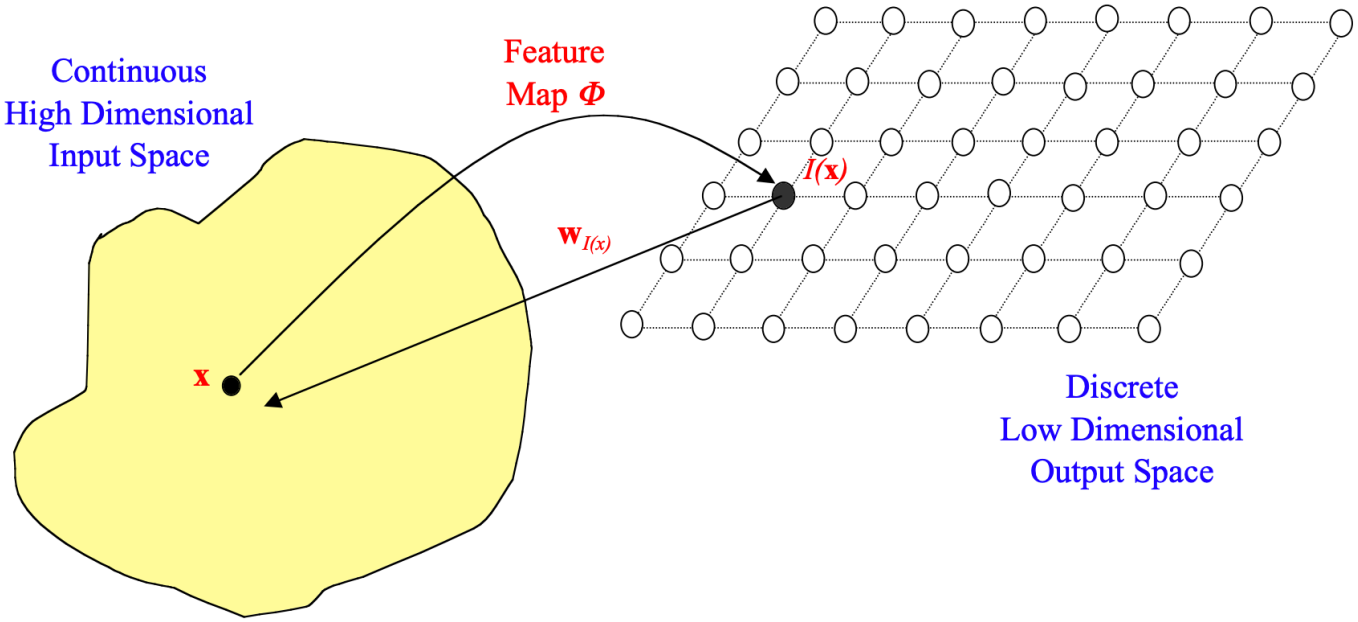
The spatial location of an output neuron in a topographic map corresponds to a particular domain or feature drawn from the input space.

Architecture

- Neurons are arranged on a flat grid.
- There is no hidden layer (only an input and an output layer).
- Each neuron on the grid is an output neuron.
- There are no interconnections among the computational nodes.

Organization of the Mapping

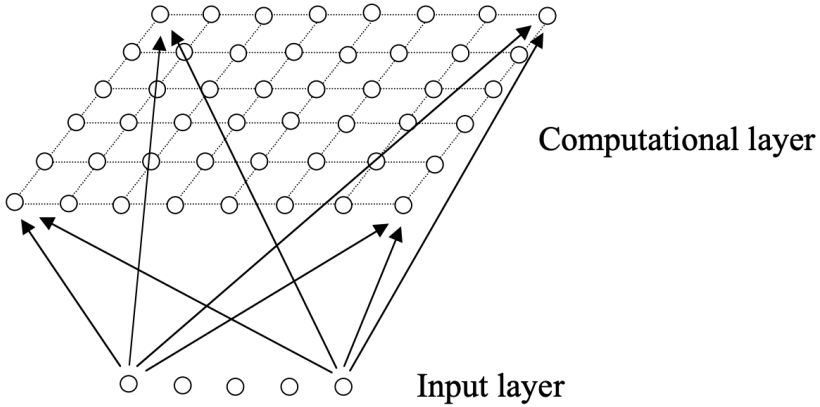
- We have points ***x*** in the input space mapping to points *I(**x**)* in the output space:



- Each point *I* in the output space will map to a corresponding point *w(I)* in the input space.

Kohonen Networks

- We shall concentrate on the particular kind of SOM known as a ***Kohonen Network***.
  - This SOM has a feed-forward structure with a single computational layer arranged in rows and columns.
  - Each neuron is fully connected to all the source nodes in the input layer:



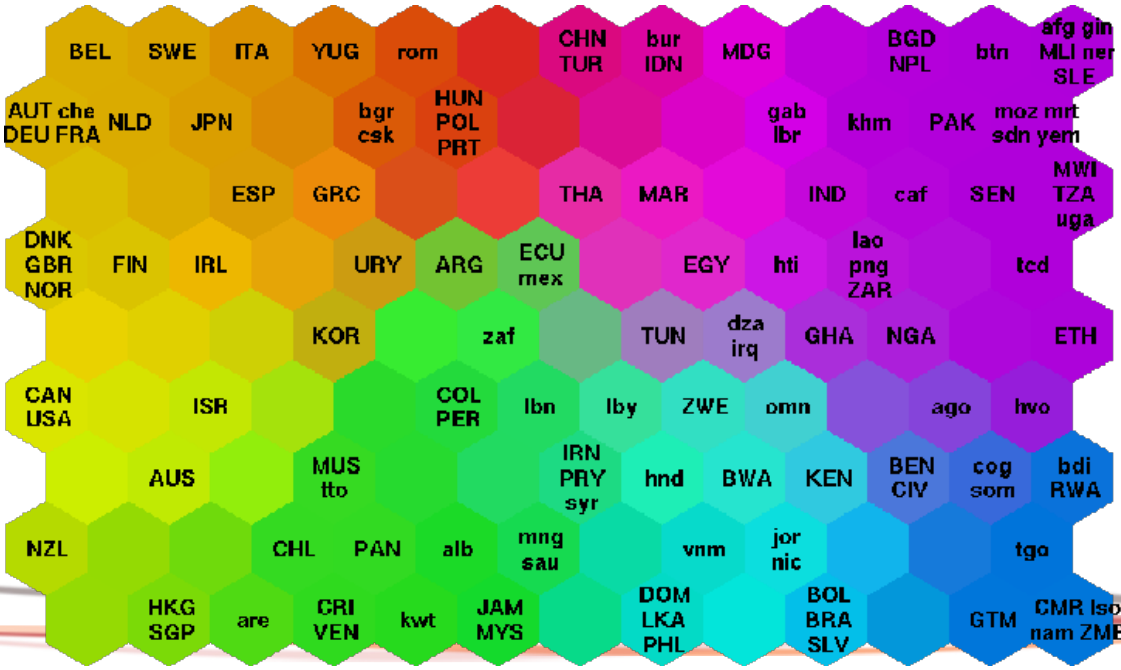
Clearly, a one dimensional map will just have a single row (or a single column) in the computational layer.

Kohonen Networks

Example: World Poverty Map

- The Self-Organizing Map (SOM) can be used to portray complex correlations in statistical data. Here the data consisted of World Bank statistics of countries in 1992. Altogether 39 indicators describing various quality-of-life factors, such as state of health, nutrition, educational services, etc, were used.

Countries organized on a self-organizing map based on indicators related to poverty:



The Country Names

AFG	Afghanistan	GTM	Guatemala	NZL	New Zealand
AGO	Angola	HKG	Hong Kong	OAN	Taiwan, China
ALB	Albania	HND	Honduras	OMN	Oman
ARE	United Arab Emirates	HTI	Haiti	PAK	Pakistan
ARG	Argentina	HUN	Hungary	PAN	Panama
AUS	Australia	HVO	Burkina Faso	PER	Peru
AUT	Austria	IDN	Indonesia	PHL	Philippines
BDI	Burundi	IND	India	PNG	Papua New Guinea
BEL	Belgium	IRL	Ireland	POL	Poland
BEN	Benin	IRN	Iran, Islamic Rep.	PRT	Portugal
BGD	Bangladesh	IRQ	Iraq	PRY	Paraguay
BGR	Bulgaria	ISR	Israel	ROM	Romania
BOL	Bolivia	ITA	Italy	RWA	Rwanda
BRA	Brazil	JAM	Jamaica	SAU	Saudi Arabia
BTN	Bhutan	JOR	Jordan	SDN	Sudan
BUR	Myanmar	JPN	Japan	SEN	Senegal
BWA	Botswana	KEN	Kenya	SGP	Singapore
CAF	Central African Rep.	KHM	Cambodia	SLE	Sierra Leone
CAN	Canada	KOR	Korea, Rep.	SLV	El Salvador
CHE	Switzerland	KWT	Kuwait	SOM	Somalia
CHL	Chile	LAO	Lao PDR	SWE	Sweden
CHN	China	LBN	Lebanon	SYR	Syrian Arab Rep.
CIV	Cote d'Ivoire	LBR	Liberia	TCD	Chad
CMR	Cameroon	LBY	Libya	TGO	Togo
COG	Congo	LKA	Sri Lanka	THA	Thailand
COL	Colombia	LSO	Lesotho	TTO	Trinidad and Tobago
CRI	Costa Rica	MAR	Morocco	TUN	Tunisia
CSK	Czechoslovakia	MDG	Madagascar	TUR	Turkey
DEU	Germany	MEX	Mexico	TZA	Tanzania
DNK	Denmark	MLI	Mali	UGA	Uganda
DOM	Dominican Rep.	MNG	Mongolia	URY	Uruguay
DZA	Algeria	MOZ	Mozambique	USA	United States
ECU	Ecuador	MRT	Mauritania	VEN	Venezuela
EGY	Egypt, Arab Rep.	MUS	Mauritius	VNM	Viet Nam
ESP	Spain	MWI	Malawi	YEM	Yemen, Rep.
ETH	Ethiopia	MYS	Malaysia	YUG	Yugoslavia
FIN	Finland	NAM	Namibia	ZAF	South Africa
FRA	France	NER	Niger	ZAR	Zaire
GAB	Gabon	NGA	Nigeria	ZMB	Zambia
GBR	United Kingdom	NIC	Nicaragua	ZWE	Zimbabwe
GHA	Ghana	NLD	Netherlands		
GIN	Guinea	NOR	Norway		
GRC	Greece	NPL	Nepal		

Components of Self Organisation

- The self-organization process involves four major components:
  - ***Initialization***: All the connection weights are initialized with small random values.
  - ***Competition:*** For each input pattern, the neurons compute their respective values of a discriminant function which provides the basis for competition. The particular neuron with the smallest value of the discriminant function is declared the winner.

The Initialization Process

- If the input space is *D* dimensional (i.e. there are *D* input units), we can write the input patterns as $\mathbf{x} = \{x_i : i = 1, \ldots, D\}$ and the connection weights between the input units *i* and the neurons *j* in the computation layer can be written $\mathbf{w}_j = \{w_{ji} : j = 1, \ldots, N; i = 1, \ldots, D\}$ where *N* is the total number of neurons.

In our example we have D=3 input units, 2 dimensional 4x4 Kohonen map and we have 16x3 dimensional weight vector.

[0.1, 0.1, 0.1]	[0.2,0.2,0.2]	[0.2,0.2,0.2]	[0.1, 0.1, 0.1]
[0.1, 0.1, 0.1]	[0.5 0.5 0.5]	[0.3 0.3 0.3]	[0.8 0.8 0.8]
[0.2,0.2,0.2]	[0.7 0.7 0.7]	[0.9 0.9 0.9]	[0.5 0.5 0.5]
[0.6 0.6 0.6]	[0.8 0.8 0.8]	[0.8 0.8 0.8]	[0.9 0.9 0.9]

Components of Self Organisation

- The self-organization process involves four major components:
  - ***Cooperation***: The winning neuron determines the spatial location of a topological neighbourhood of excited neurons, thereby providing the basis for cooperation among neighbouring neurons.
  - ***Adaptation***: The excited neurons decrease their individual values of the discriminant function in relation to the input pattern through suitable adjustment of the associated connection weights, such that the response of the winning neuron to the subsequent application of a similar input pattern is enhanced.

The Competitive Process

- We can then define our discriminant function to be the squared Euclidean distance between the input vector **x** and the weight vector $\mathbf{w}_j$ for each neuron *j*

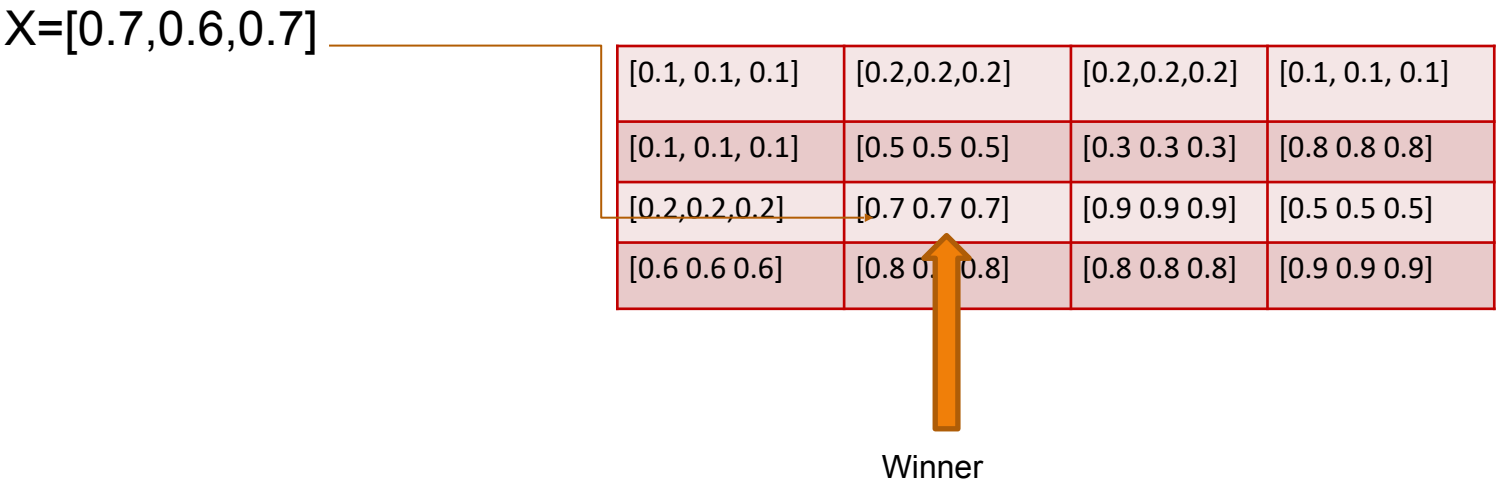
$$dj(x) = \sum_{i=1}^{D} \left(xi - w_{ji} \right) 2$$

X=[0.7,0.6,0.7]

[0.1, 0.1, 0.1]	[0.2,0.2,0.2]	[0.2,0.2,0.2]	[0.1, 0.1, 0.1]
[0.1, 0.1, 0.1]	[0.5 0.5 0.5]	[0.3 0.3 0.3]	[0.8 0.8 0.8]
[0.2,0.2,0.2]	[0.7 0.7 0.7]	[0.9 0.9 0.9]	[0.5 0.5 0.5]
[0.6 0.6 0.6]	[0.8 0.8 0.8]	[0.8 0.8 0.8]	[0.9 0.9 0.9]

The Competitive Process

- In other words, the neuron whose weight vector comes closest to the input vector (i.e. is most similar to it) is declared the winner. In this way the continuous input space can be mapped to the discrete output space of neurons by a simple process of competition between the neurons.



The Cooperative Process

- In neurobiological studies they find that there is ***lateral interaction*** within a set of excited neurons. When one neuron fires, its closest neighbours tend to get excited more than those further away. There is a ***topological neighbourhood*** that decays with distance.
- We want to define a similar topological neighbourhood for the neurons in our SOM. If S_{ij} is the lateral distance between neurons *i* and *j* on the grid of neurons, we take

$$T_{j,\, I(x)} = \exp(-\frac{S^2_{j,\, I(x)}}{2\sigma 2})$$

as our topological neighbourhood, where ***I(x)*** is the index of the winning neuron.

The Cooperative Process

- If S_{ij} is the lateral distance between neurons *i* and *j* on the grid of neurons, we take

$$T_{j,\,I(x)} = \exp\left(-\frac{S^2_{j,\,I(x)}}{2\sigma 2}\right)$$

as our topological neighbourhood, where ***I(x)*** is the index of the winning neuron.

[0.1, 0.1, 0.1]	[0.2,0.2,0.2]	[0.2,0.2,0.2]	[0.1, 0.1, 0.1]
[0.1, 0.1, 0.1]	[0.5 0.5 0.5]	[0.3 0.3 0.3]	[0.8 0.8 0.8]
[0.2,0.2,0.2]	[0.7 0.7 0.7]	[0.9 0.9 0.9]	[0.5 0.5 0.5]
[0.6 0.6 0.6]	[0.8 0.8 0.8]	[0.8 0.8 0.8]	[0.9 0.9 0.9]

Let's select σ=2 meaning that we consider updating 2 neighbors.
Yellow: winner; blue: 1st level neighbors, green: 2nd level neighbors.
For the winning node, $T_{3,2}$=exp(0)=1

$T_{2,1}$=exp(-1/(2x2))=0.778

$T_{1,1}$=exp(-4/(2x2))=0.367

The Cooperative Process

- This has several important properties: it is maximal at the winning neuron, it is symmetrical about that neuron, **it decreases monotonically to zero as the distance goes to infinity,** and it is translation invariant (i.e. independent of the location of the winning neuron)

[0.1, 0.1, 0.1]	[0.2,0.2,0.2]	[0.2,0.2,0.2]	[0.1, 0.1, 0.1]
[0.1, 0.1, 0.1]	[0.5 0.5 0.5]	[0.3 0.3 0.3]	[0.8 0.8 0.8]
[0.2,0.2,0.2]	[0.7 0.7 0.7]	[0.9 0.9 0.9]	[0.5 0.5 0.5]
[0.6 0.6 0.6]	[0.8 0.8 0.8]	[0.8 0.8 0.8]	[0.9 0.9 0.9]

Let's select σ=2 meaning that we consider updating 2 neighbors.
Yellow: winner; blue: 1st level neighbors, green: 2nd level neighbors.
For the winning node, $T_{3,2}=\exp(0)=1$
$T_{2,1}=\exp(-1/(2x2))=0.778$
$T_{1,1}=\exp(-4/(2x2))=0.367$

The Cooperative Process

A special feature of the SOM is that the size σ of the neighbourhood needs to decrease with time. A popular time dependence is an exponential decay:

$$\sigma(t) = \sigma 0 \exp(-\frac{t}{\tau\sigma})$$

[0.1, 0.1, 0.1]	[0.2,0.2,0.2]	[0.2,0.2,0.2]	[0.1, 0.1, 0.1]
[0.1, 0.1, 0.1]	[0.5 0.5 0.5]	[0.3 0.3 0.3]	[0.8 0.8 0.8]
[0.2,0.2,0.2]	[0.7 0.7 0.7]	[0.9 0.9 0.9]	[0.5 0.5 0.5]
[0.6 0.6 0.6]	[0.8 0.8 0.8]	[0.8 0.8 0.8]	[0.9 0.9 0.9]

Let's select σ=2 meaning that we consider updating 2 neighbors. (topological radius =2)
Yellow: winner; blue: 1st level neighbors, green: 2nd level neighbors.
For the winning node, $T_{3,2}$=exp(0)=1
$T_{2,1}$=exp(-1/(2x2))=0.778
$T_{1,1}$=exp(-4/(2x2))=0.367
These T values will not be static and decrease over time!

The Adaptive Process

- Clearly our SOM must involve some kind of adaptive, or learning, process by which the outputs become self-organised and the feature map between inputs and outputs is formed.
- The point of the topographic neighbourhood is that not only the winning neuron gets its weights updated, but its neighbours will have their weights updated as well, although by not as much as the winner itself.
- In practice, the appropriate weight update equation is

$$\Delta wji = \eta(t).T_{j,\, I(x)}(t).(x_i - w_{ji})$$

in which we have a time (epoch) *t* dependent learning rate

$\eta(t) = \eta 0 \exp(-\frac{t}{\tau\eta})$, and the updates are applied for all the training patterns **x** over many epochs.

The Adaptive Process

The effect of each learning weight update is to move the weight vectors $\mathbf{w}_i$ of the winning neuron and its neighbours towards the input vector **x**. Repeated presentations of the training data thus leads to topological ordering.

$$\Delta wji = \eta(t).T_{j,\, I(x)}(t).(x_i - w_{ji})$$

in which we have a time (epoch) *t* dependent learning rate $\eta(t) = \eta 0 \exp(-\frac{t}{\tau\eta})$, and the updates are applied for all the training patterns **x** over many epochs.

[0.1, 0.1, 0.1]	[0.2,0.2,0.2]	[0.2,0.2,0.2]	[0.1, 0.1, 0.1]
[0.1, 0.1, 0.1]	[0.5 0.5 0.5]	[0.3 0.3 0.3]	[0.8 0.8 0.8]
[0.2,0.2,0.2]	[0.7 0.7 0.7]	[0.9 0.9 0.9]	[0.5 0.5 0.5]
[0.6 0.6 0.6]	[0.8 0.8 0.8]	[0.8 0.8 0.8]	[0.9 0.9 0.9]

X=[0.7,0.6,0.7]

For the winner node, at the beginning:

$\Delta w23 = 1.(x_i - w_{ji})$=[0,-0.1,0]

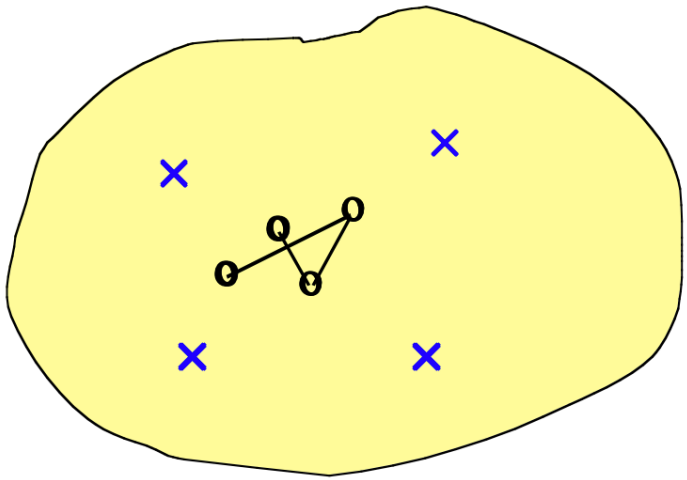
$w23 = \Delta w23 + old(w23)$=[0.7,0.6,0.7]

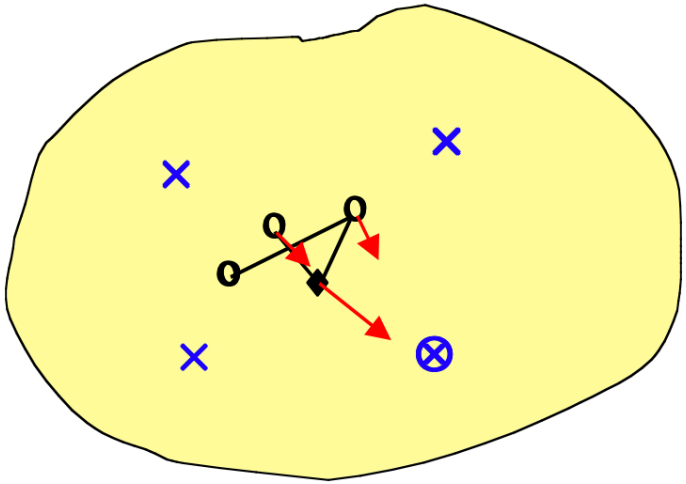
Update the neighbors in the radius similarly.

Ordering and Convergence

- Provided the parameters $(\sigma_0, \tau_\sigma, \eta_0, \tau_\eta)$ are selected properly, we can start from an initial state of complete disorder, and the SOM algorithm will gradually lead to an organized representation of activation patterns drawn from the input space. There are two identifiable phases of this adaptive process:
  - **Ordering or self-organizing phase –** during which the topological ordering of the weight vectors takes place.
    - Typically this will take as many as 1000 iterations of the SOM algorithm, and careful consideration needs to be given to the choice of neighbourhood and learning rate parameters.
  - **Convergence phase –** during which the feature map is fine tuned and comes to provide an accurate statistical quantification of the input space.
    - Typically the number of iterations in this phase will be at least 500 times the number of neurons in the network, and again the parameters must be chosen carefully

Visualising the Self Organisation Process

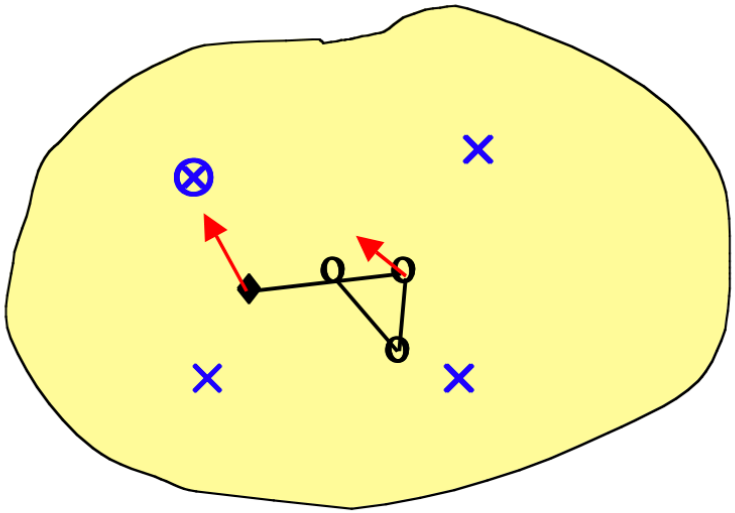




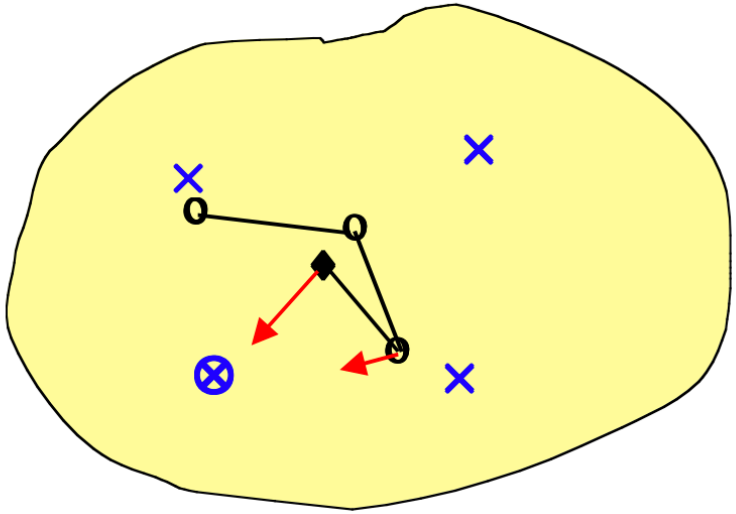
Suppose we have four data points (crosses) in our continuous 2D input space, and want to map this onto four points in a discrete 1D output space. The output nodes map to points in the input space (circles). Random initial weights start the circles at random positions in the centre of the input space.

We randomly pick one of the data points for training (cross in circle). The closest output point represents the winning neuron (solid diamond). That winning neuron is moved towards the data point by a certain amount, and the two neighbouring neurons move by smaller amounts (arrows).

Visualising the Self Organisation Process



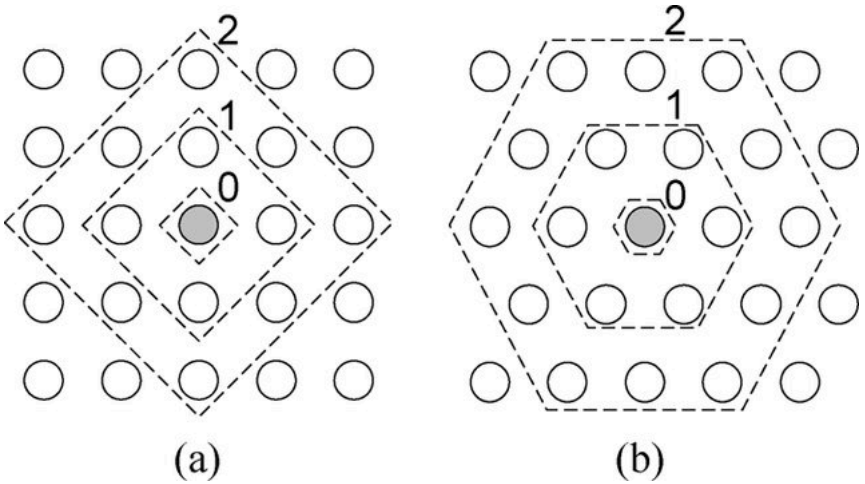
Next we randomly pick another data point for training (cross in circle). The closest output point gives the new winning neuron (solid diamond). The winning neuron moves towards the data point by a certain amount, and the one neighbouring neuron moves by a smaller amount (arrows).



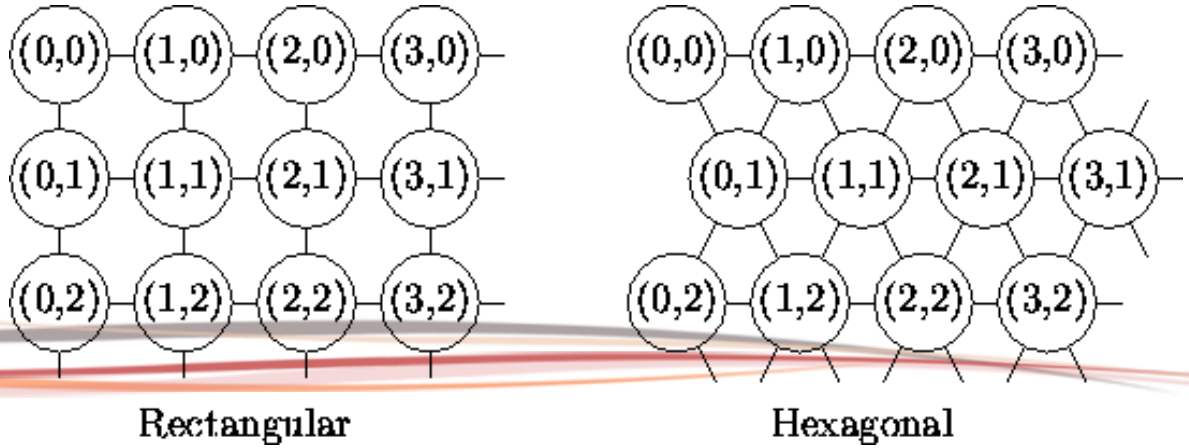
We carry on randomly picking data points for training (cross in circle). Each winning neuron moves towards the data point by a certain amount, and its neighbouring neuron(s) move by smaller amounts (arrows). Eventually the whole output grid unravels itself to represent the input space.

Visualising the Self Organisation Process

Neighborhood of a given winner unit



- Topologies:



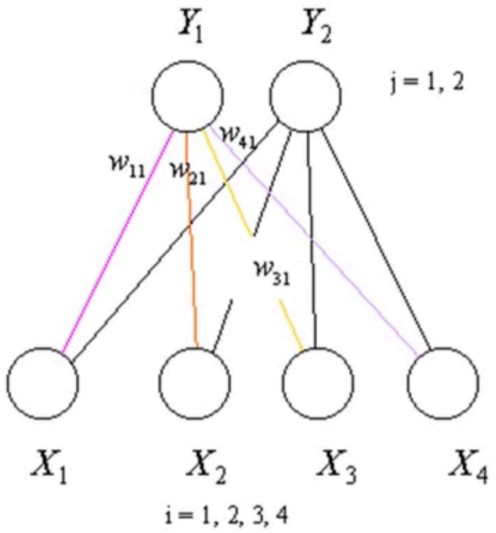
Overview of the SOM Algorithm

- We have a spatially ***continuous input space***, in which our input vectors live. The aim is to map from this to a low dimensional spatially ***discrete output space***, the topology of which is formed by arranging a set of neurons in a grid. Our SOM provides such a nonlinear transformation called a feature map.
- The stages of the SOM algorithm can be summarised as follows:
  - **Initialization** – Choose random values for the initial weight vectors $\boldsymbol{w}_j$.
  - **Sampling** – Draw a sample training input vector **x** from the input space.
  - **Matching** – Find the winning neuron ***I(x)*** with weight vector closest to input vector.
  - **Updating** – Apply the weight update equation $\Delta wji = \eta(t)T_{j,I(x)}(t)(xi - wji)$.
  - **Continuation** – keep returning to step 2 until the feature map stops changing.

Toy Example

- Consider a simple example in which there are only 4 input patterns.
- Let the learning rate at time t+1 is $\alpha(t+1) = \dfrac{\alpha(t)}{2}$ and $\alpha(t=0) = 0.6$.
- Let the topological radius is zero.

x_1	x_2	x_3	x_4
1	1	0	0
0	0	0	1
1	0	0	0
0	0	1	1



Let the initial weight matrix be

$$\begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \\ w_{41} & w_{42} \end{bmatrix} = \begin{bmatrix} 0.2 & 0.8 \\ 0.6 & 0.4 \\ 0.5 & 0.7 \\ 0.9 & 0.3 \end{bmatrix}$$

Toy Example

For vector 1100 (We use Euclidean distance)

$D(1)=(1-0.2)^2+(1-0.6)^2+(0-0.5)^2+(0-0.9)^2=1.86$

$D(1)=1.86$ $D(2)=0.98$

- Hence the winner node is 2. As the radius is set to zero, we don't need to update the weights of any neighbourhood weights.

$$wij(new) = wij(old) + \alpha(xi - wij(old))$$

The new weight matrix is:

Let the initial weight matrix be

$$\begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \\ w_{41} & w_{42} \end{bmatrix} = \begin{bmatrix} 0.2 & 0.8 \\ 0.6 & 0.4 \\ 0.5 & 0.7 \\ 0.9 & 0.3 \end{bmatrix} \Rightarrow \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \\ w_{41} & w_{42} \end{bmatrix} = \begin{bmatrix} 0.2 & 0.92 \\ 0.6 & 0.76 \\ 0.5 & 0.28 \\ 0.9 & 0.12 \end{bmatrix}$$

$$w_{12}(new) = w_{12}(old) + \alpha(x_1 + w_{12}(old)$$
$$= 0.8 + 0.6(1 - 0.8) = 0.92$$

The weights connecting to the second node will be updated.

Toy Example

Likewise for remains training patterns

For vector 0001
D(1) = 0.66, D(2) = 2.2768
Hence J = 1.

$$\begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \\ w_{41} & w_{42} \end{bmatrix} = \begin{bmatrix} 0.08 & 0.92 \\ 0.24 & 0.76 \\ 0.20 & 0.28 \\ 0.96 & 0.12 \end{bmatrix}$$

For vector 0011
D(1) = 0.7056, D(2) = 2.724
Hence J = 1

$$\begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \\ w_{41} & w_{42} \end{bmatrix} = \begin{bmatrix} 0.032 & 0.968 \\ 0.096 & 0.304 \\ 0.680 & 0.112 \\ 0.984 & 0.048 \end{bmatrix}$$

For vector 1000
D(1) = 1.8656, D(2) = 0.6768
Hence J = 2

$$\begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \\ w_{41} & w_{42} \end{bmatrix} = \begin{bmatrix} 0.08 & 0.968 \\ 0.24 & 0.304 \\ 0.20 & 0.112 \\ 0.96 & 0.048 \end{bmatrix}$$

Toy Example

Now reduce learning rate (step 6): $\alpha(1) = \frac{\alpha(0)}{2} = \frac{0.6}{2} = 0.3$

It can be shown that after **100 presentations** of all the input vector, the final weight matrix is

$$\begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \\ w_{41} & w_{42} \end{bmatrix} = \begin{bmatrix} 6.7x10^{-17} & 1 \\ 2x10^{-16} & 0.49 \\ 0.51 & 2.3x10^{-16} \\ 1 & 1x10^{-16} \end{bmatrix}$$

This matrix seems to converge to

$$\begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \\ w_{41} & w_{42} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0.5 \\ 0.5 & 0 \\ 1 & 0 \end{bmatrix}$$

Toy Example

TEST NETWORK

Suppose the input pattern is 1100. Then

$$D(j) = (w_{1j} - x_1)^2 + (w_{2j} - x_2)^2 + (w_{3j} - x_3)^2 + (w_{4j} - x_4)^2$$

$$D(1) = (0-1)^2 + (0-1)^2 + (0.5-0)^2 + (1-0)^2 = 3.25$$

$$D(2) = (1-1)^2 + (0.5-1)^2 + (0-0)^2 + (0-0)^2 = 0.25$$

Thus neuron 2 is the "**winner**", and is the localized active region of the SOM. Notice that we may label this input pattern to belong to cluster 2.

This matrix seems to converge to

$$\begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \\ w_{41} & w_{42} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0.5 \\ 0.5 & 0 \\ 1 & 0 \end{bmatrix}$$

Cluster 1 Cluster 2

x_1	x_2	x_3	x_4	Cluster
1	1	0	0	2
0	0	0	1	1
1	0	0	0	2
0	0	1	1	1

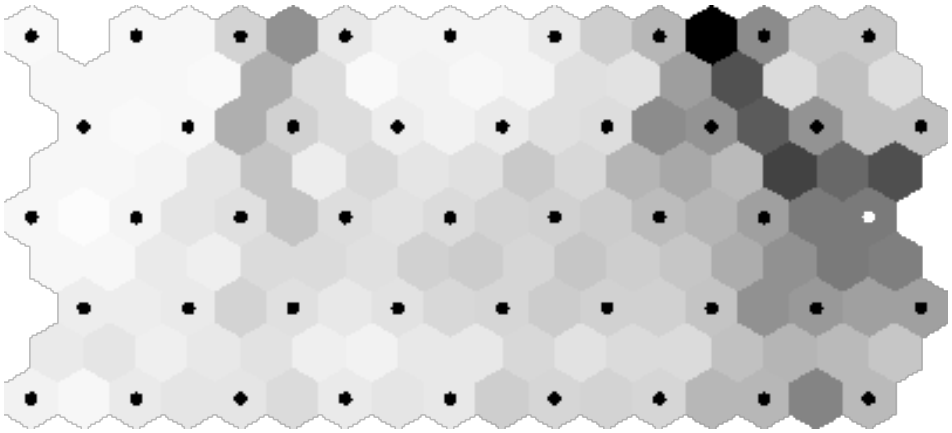
Visualisation: U-matrix

- U-Matrix (Ultsch, Guimarães et al. 1993), or UMAT
- U-matrix (unified distance matrix) representation of the SOM visualizes the distances between the neurons. The distance between the adjacent neurons is calculated and presented with different colourings between the adjacent nodes.
  - A **dark colouring** between the neurons corresponds to a large distance and thus a gap between the codebook values in the input space.
  - A **light colouring** between the neurons signifies that the codebook vectors are close to each other in the input space. Light areas can be thought as clusters and dark areas as cluster separators.
- This can be a helpful presentation when one tries to find clusters in the input data without having any a priori information about the clusters.

Ultsch, A., G. Guimarães, et al. (1993). Knowledge Extraction from Artificial Neural Networks and Applications. Transputer-Anwender-Treffen, Aachen, Springer Verlag.

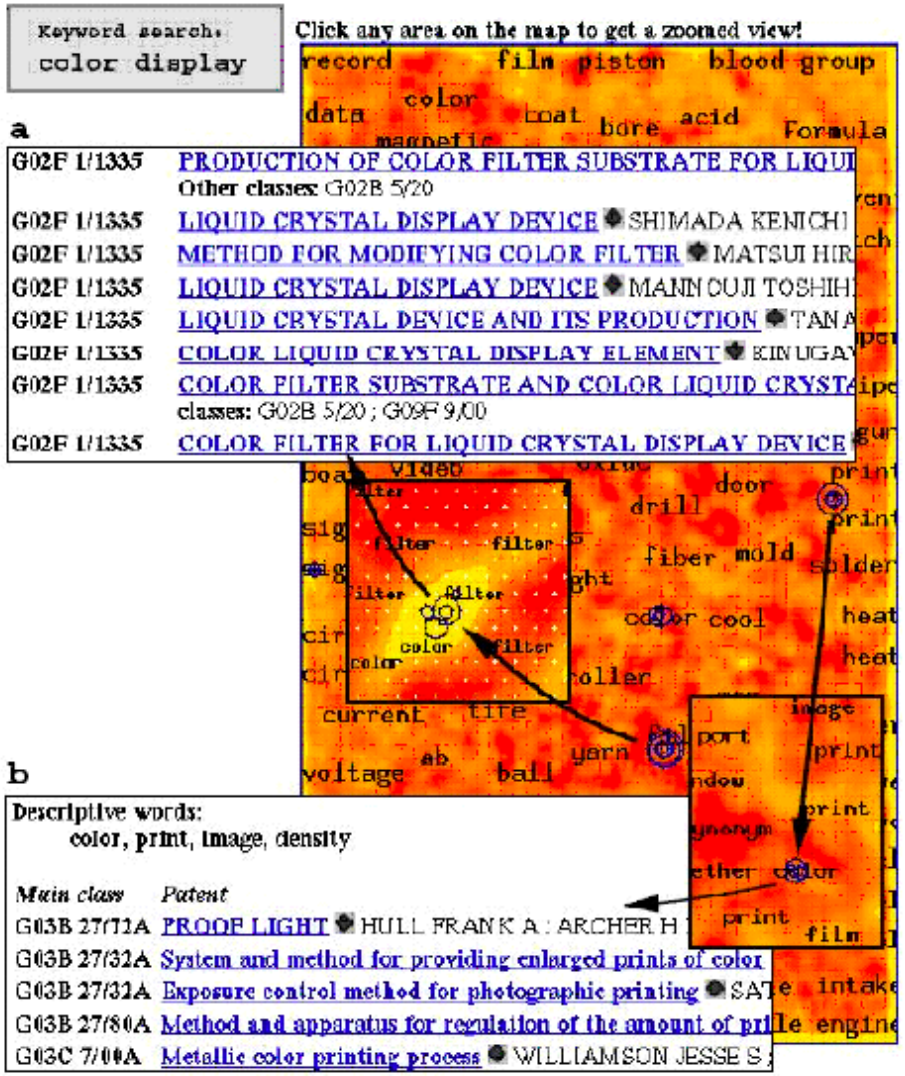
Visualisation: U-matrix

- We can see the neurons of the network marked as black dots. The representation reveals that these are a separate cluster in the upper right corner of this representation.
- The clusters are separated by a dark gap. This result was achieved by unsupervised learning, that is, without human intervention. Teaching a SOM and representing it with the U-matrix offers a fast way to get insight of the data distribution.



Application: WEBSOM - Self-Organizing Maps of Document Collections

- WEBSOM is a SOM that organizes massive document collections. It was developed by Professor Teuvo Kohonen at the Neural Networks Research Center in Helsinki University of Technology in the late 1990's.
- The project successfully created a SOM that acted as a graphical search engine, classifying over 7,000,000 patent abstracts based on the frequency of occurence of a set of words.



Honkela, T., Kaski, S., Lagus, K., & Kohonen, T. (1997, June). WEBSOM—self-organizing maps of document collections. In *Proceedings of WSOM* (Vol. 97, pp. 4-6).

Advantages and Disadvantages

- Advantages:
  - Projects high dimensional data onto lower dimensional map
  - Preserve the topological properties of the input space so that similar objects will be mapped to nearby locations on the map
  - Useful for visualization
  - Preserve the topology relations of input space
- Disadvantages:
  - Clustering result depends on initial weight vector
  - Requires large quantity of good training data
  - Difficult to determine optimal map size
  - Edge/boundary effect (Neurons on the map boundary have fewer neighbors) in traditional SOM (two-dimensional regular spacing in rectangular or hexagonal grid)
  - High computation cost

Why Need Feature Map Optimization?

- Feature map size
  - Important to detect the deviation of the data
  - It directly influences the clustering
- Usually, it is recommended that
  - Choose large feature map size even when there is few input data samples
    - On some nodes, there will be none or few instances.
    - Therefore this method does not cluster instances properly (some use agglomerative clustering on the results of the SOM)
  - Small map has relatively no freedom to align topological relations between neurons

Why Need Feature Map Optimization?

- If map size is too large
  - The differences between neurons are too detail
  - The computation cost will increase extremely high due to a huge number of neurons
- If map size is too small
  - The output patterns will be more general
  - Could miss some important differences that should be detected.

Conclusion

- SOM is an algorithm that projects high dimensional data onto a two-dimensional map.
- The projection preserves the topology of the data so that similar data items will be mapped to nearby locations on the map
- SOM still have many practical applications in pattern recognition, speech analysis, industrial and medical diagnostics, data mining
- Large quantity of good quality representative training data required

References

- John A. Bullinaria, 2004, https://www.cs.bham.ac.uk/~jxb/NN/l16.pdf