Esempio di Algoritmo Lazy: k-NN

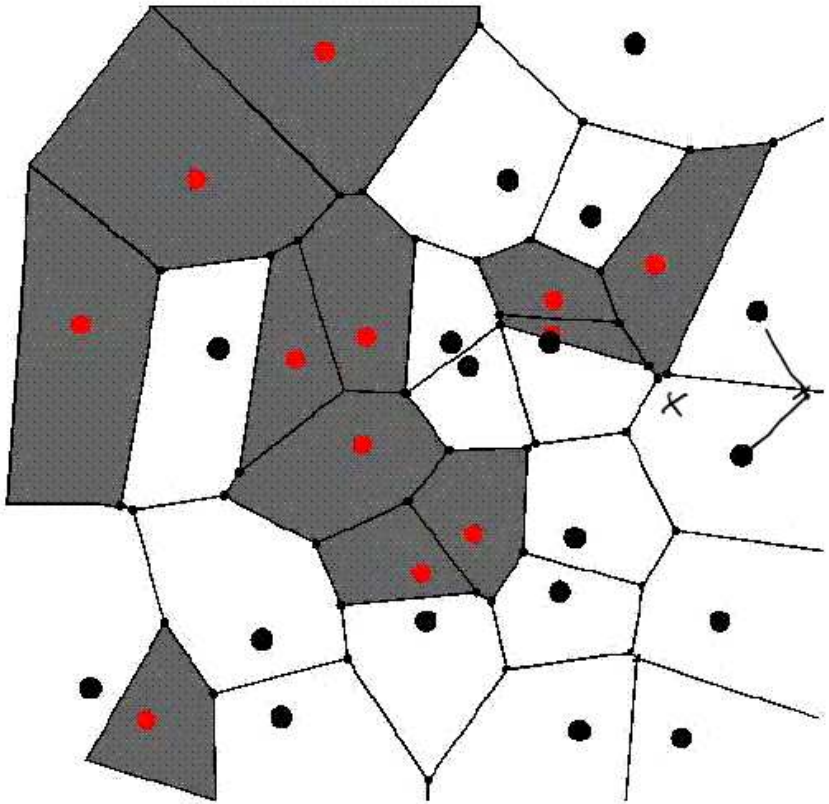
Un esempio di algoritmo Lazy è dato dal k-Nearest Neighbor

Vediamo il caso $k = 1$:

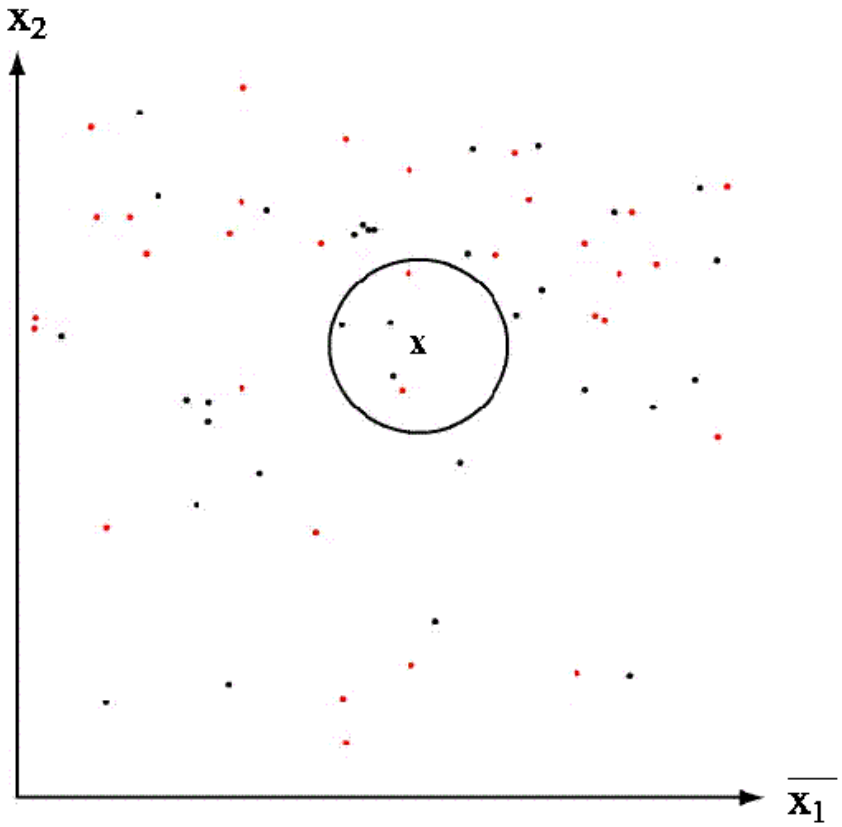
- si memorizzano i dati di apprendimento
- quando si deve effettuare una classificazione di un nuovo ingresso y, si restituisce la classe associata all'esempio pi`u vicino (tramite una metrica opportune: es. Euclidea) memorizzato:

Ovviamente k-NN è lento nel rispondere: il tempo di risposta dipende dal numero di esempi memorizzati

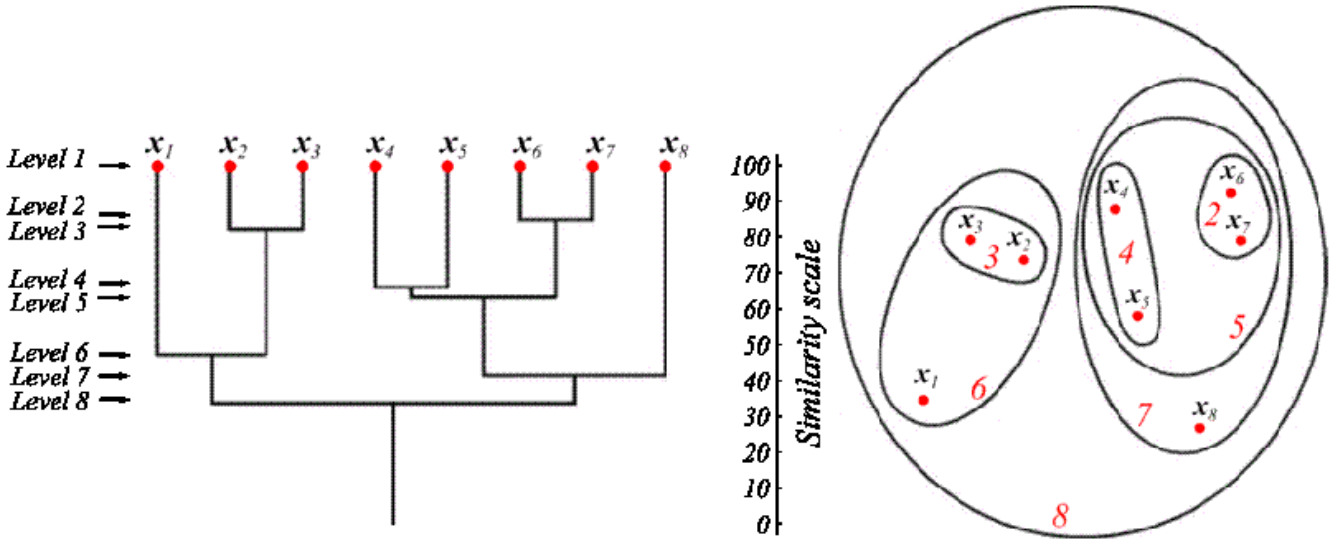
k-NN: diagramma di Voronoi



k-NN: k>1



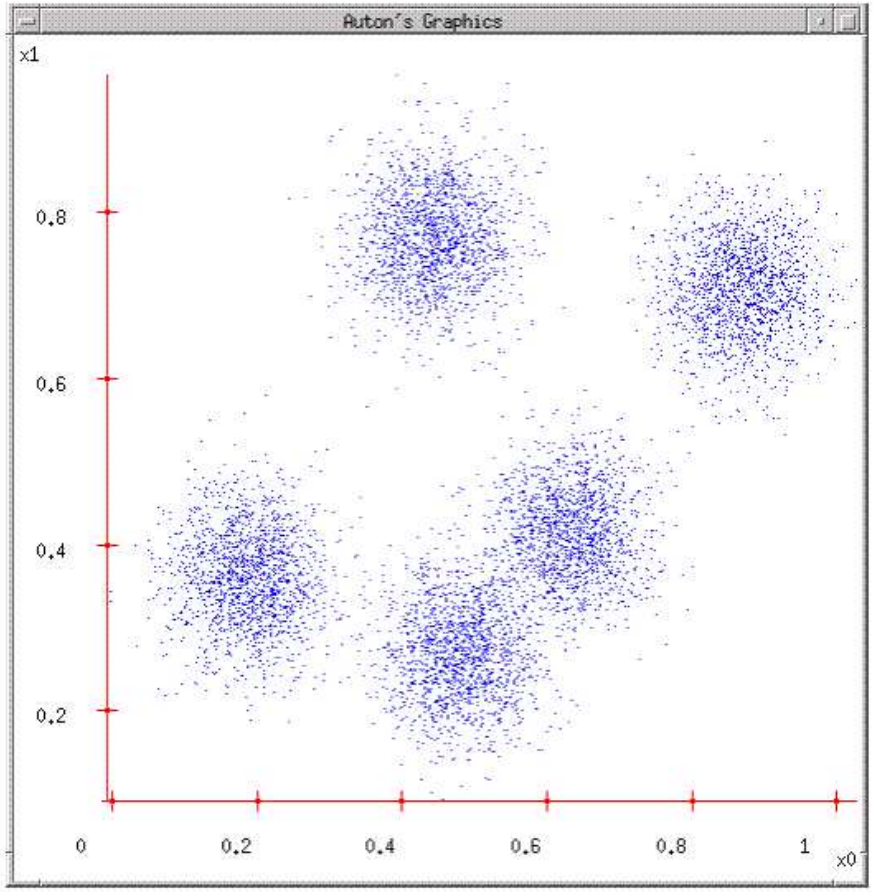
Un esempio di clustering: Clustering Gerarchico



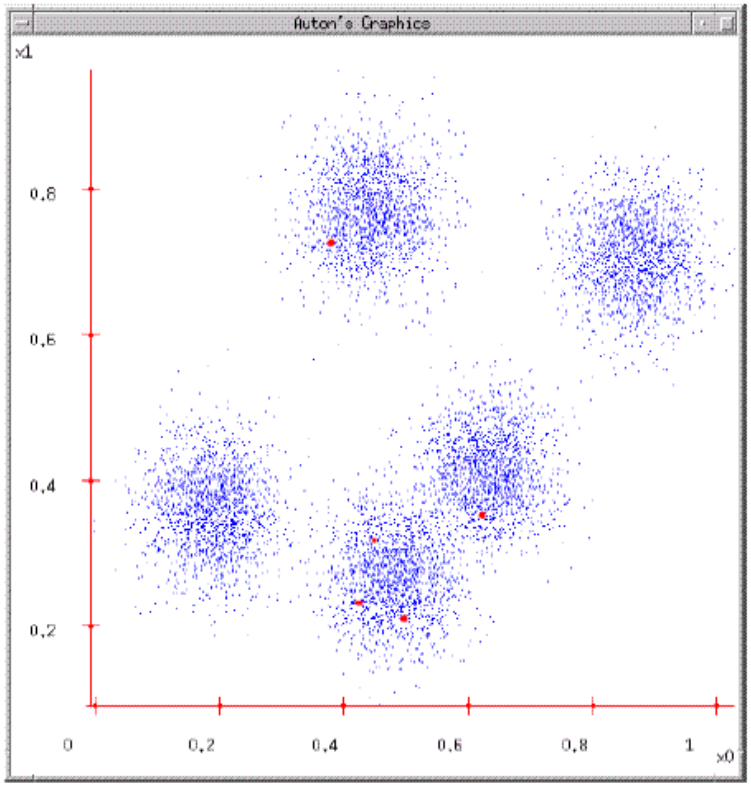
1. si selezionano i 2 vettori più vicini (es., secondo la distanza euclidea) e si costruisce un sottoalbero con figli dati dai due vettori e padre il centroide (media) dei due vettori;

2. i due vettori vengono sostituiti nell'insieme dei vettori correnti dal centroide, e si torna al passo 1.

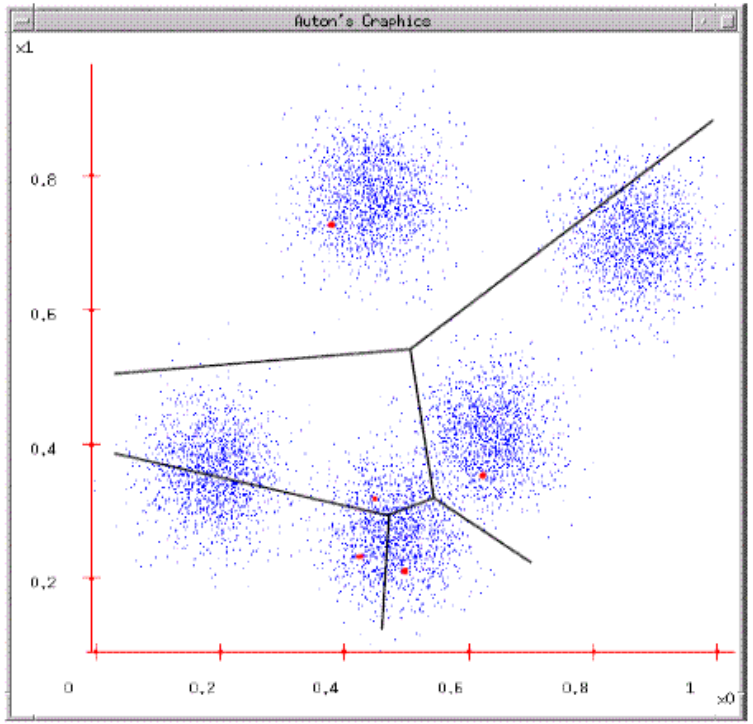
K-Means



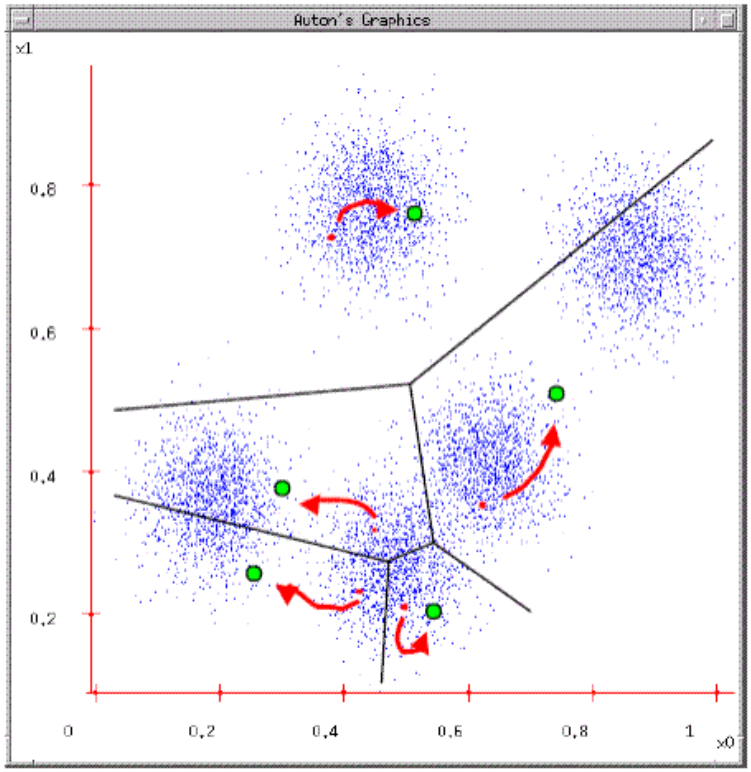
K-Means



K-Means



K-Means



K-Means

1. *Place K points into the space represented by the objects that are being clustered. These points represent initial group centroids.*
2. *Assign each object to the group that has the closest centroid.*
3. *When all objects have been assigned, recalculate the positions of the K centroids.*
4. *Repeat Steps 2 and 3 until the centroids no longer move. This produces a separation of the objects into groups from which the metric to be minimized can be calculated.*

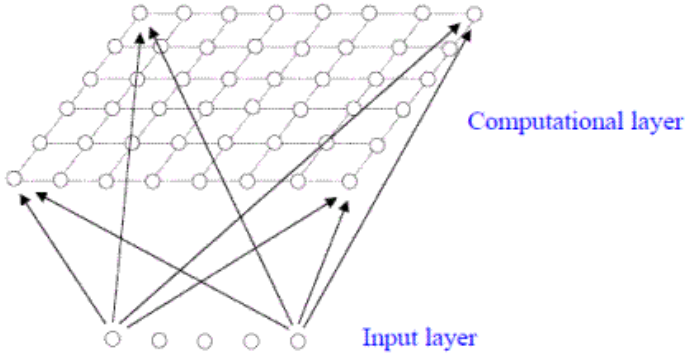
- Make initial guesses for the means $\mathbf{m}_1$, $\mathbf{m}_2$, ..., $\mathbf{m}_k$
- Until there are no changes in any mean
  - Use the estimated means to classify the samples into clusters
  - For i from 1 to k
    - Replace $\mathbf{m}_i$ with the mean of all of the samples for cluster i
  - end_for
- end_until

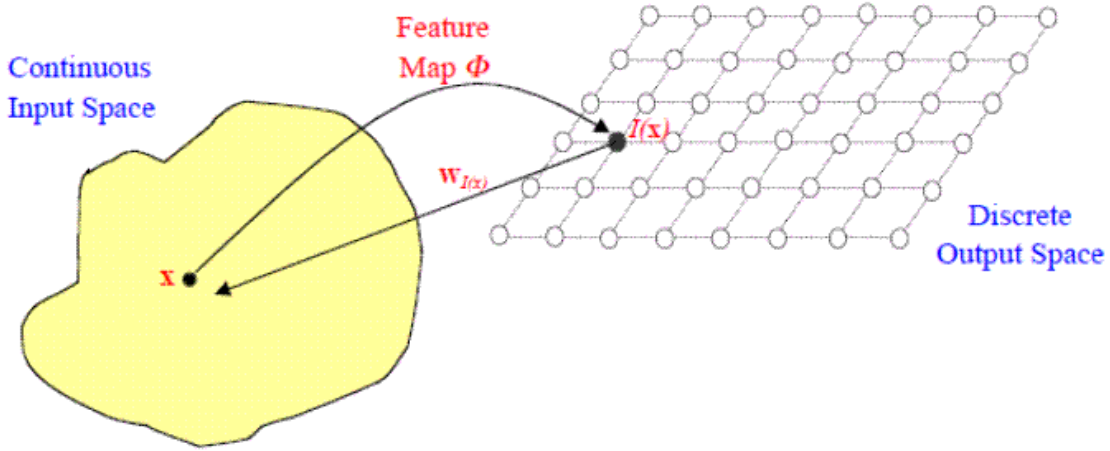
MINIMIZZA $J = \sum_{j=1}^{k} \sum_{i=1}^{n} \left\| x_i^{(j)} - c_j \right\|^2$

Self-Organizing Maps

Si ispirano alla Corteccia Cerebrale

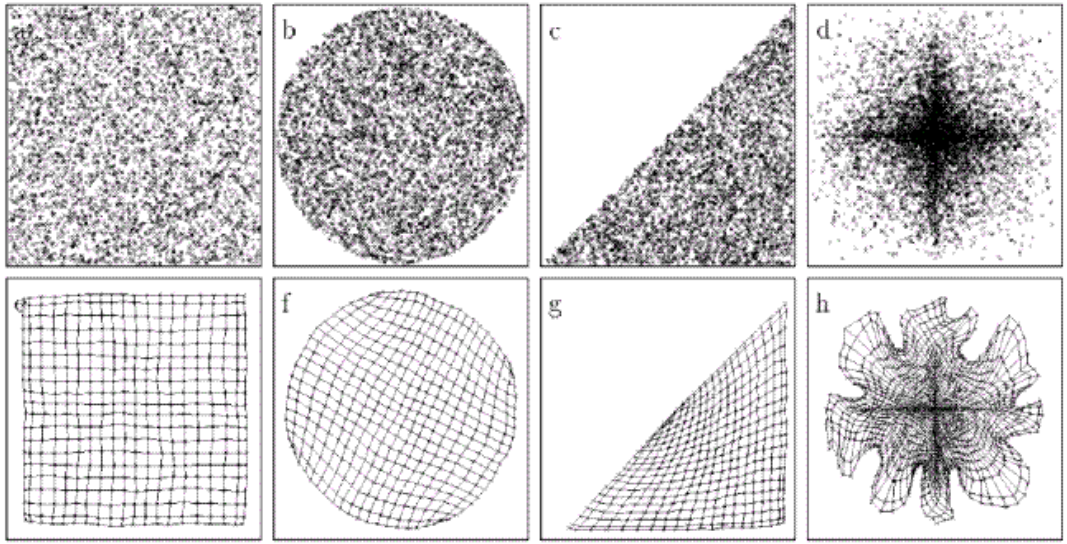
Dal punto di vista matematico, eseguono una proiezione di dati da uno spazio ad elevata dimensione ad uno spazio bidimensionale (più in generale, con poche dimensioni) cercando di preservare le relazioni topologiche fra i dati

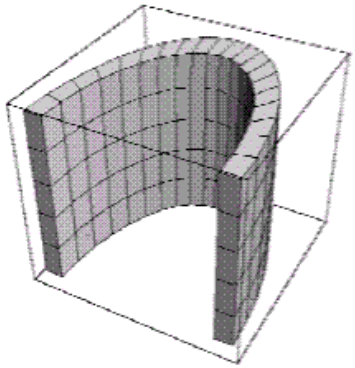


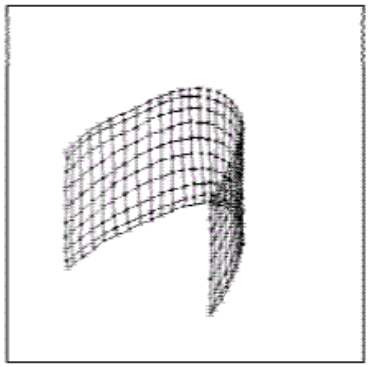


Self-Organizing Maps

Esempi di proiezioni



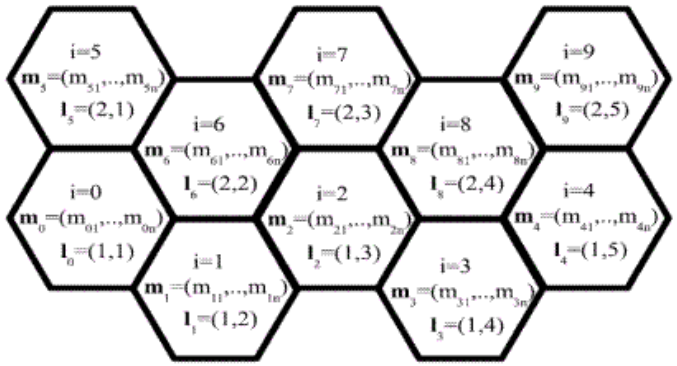




Self-Organizing Maps

i neuroni sono organizzati in una rete bidimensionale con relazioni di vicinanza topologica fra neuroni

Esempio di rete con relazione di vicinanza esagonale:



w_j

↑ vettore dei pesi associato al neurone i-esimo

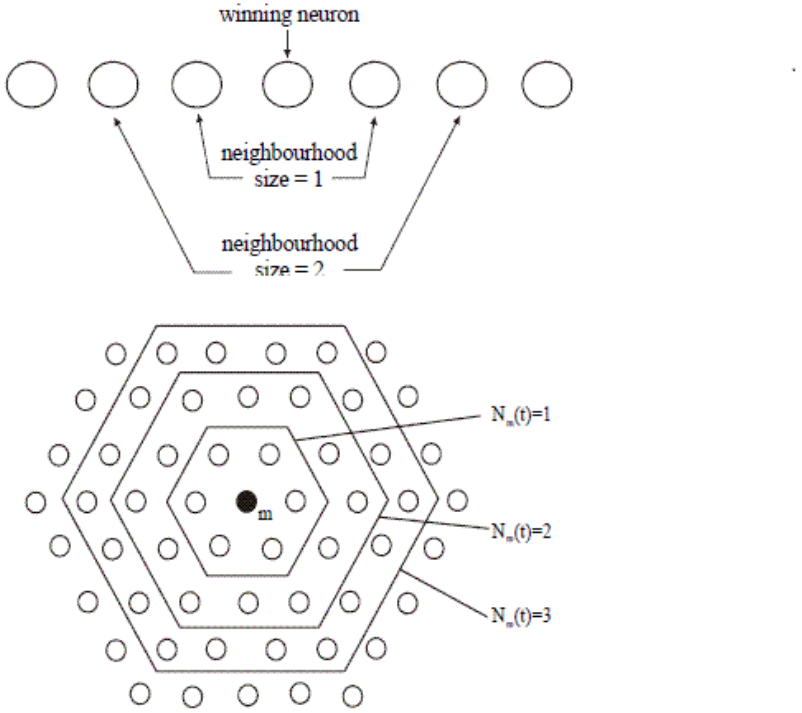
dato un vettore in ingresso u, i neuroni della rete competono fra di loro:

"vince" il neurone che ha vettore dei pesi associato più vicino ad u

$$r = \arg\min_i \| \mathbf{x} - w_i \|$$

Self-Organizing Maps

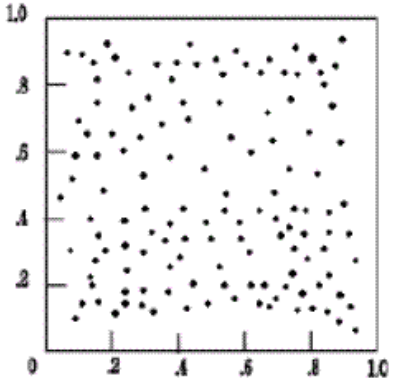
i neuroni sono organizzati in una rete bidimensionale con relazioni di vicinanza topologica fra neuroni

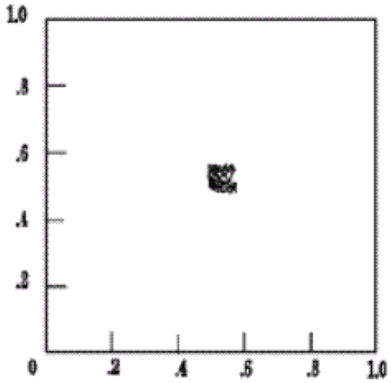


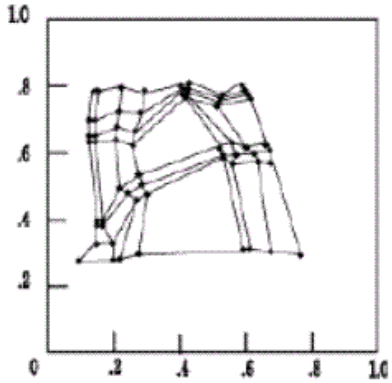
Self-Organizing Maps

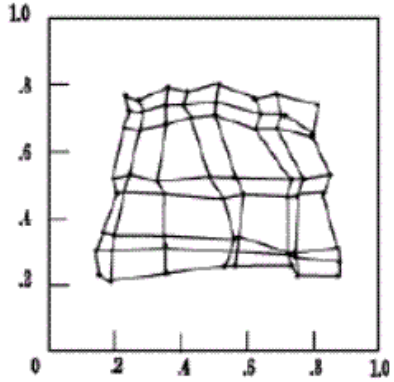
Training Algorithm

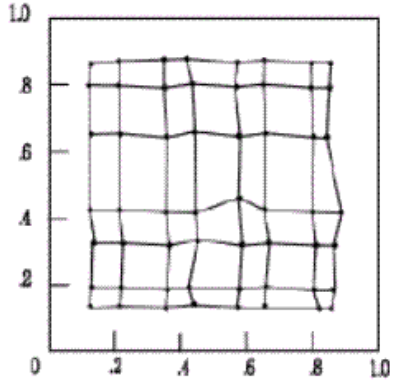
1. ***Initialization*** – Choose random values for the initial weight vectors $\mathbf{w}_j$.
2. ***Sampling*** – Draw a sample training input vector $\mathbf{x}$ from the input space.
3. ***Matching*** – Find the winning neuron $I(\mathbf{x})$ that has weight vector closest to the input vector, i.e. the minimum value of $d_j(\mathbf{x}) = \sum_{i=1}^{D} (x_i - w_{ji})^2$.
4. ***Updating*** – Apply the weight update equation $\Delta w_{ji} = \eta(t)\, T_{j,I(\mathbf{x})}(t)\, (x_i - w_{ji})$ where $T_{j,I(\mathbf{x})}(t)$ is a Gaussian neighbourhood and $\eta(t)$ is the learning rate.
5. ***Continuation*** keep returning to step 2 until the feature map stops changing.

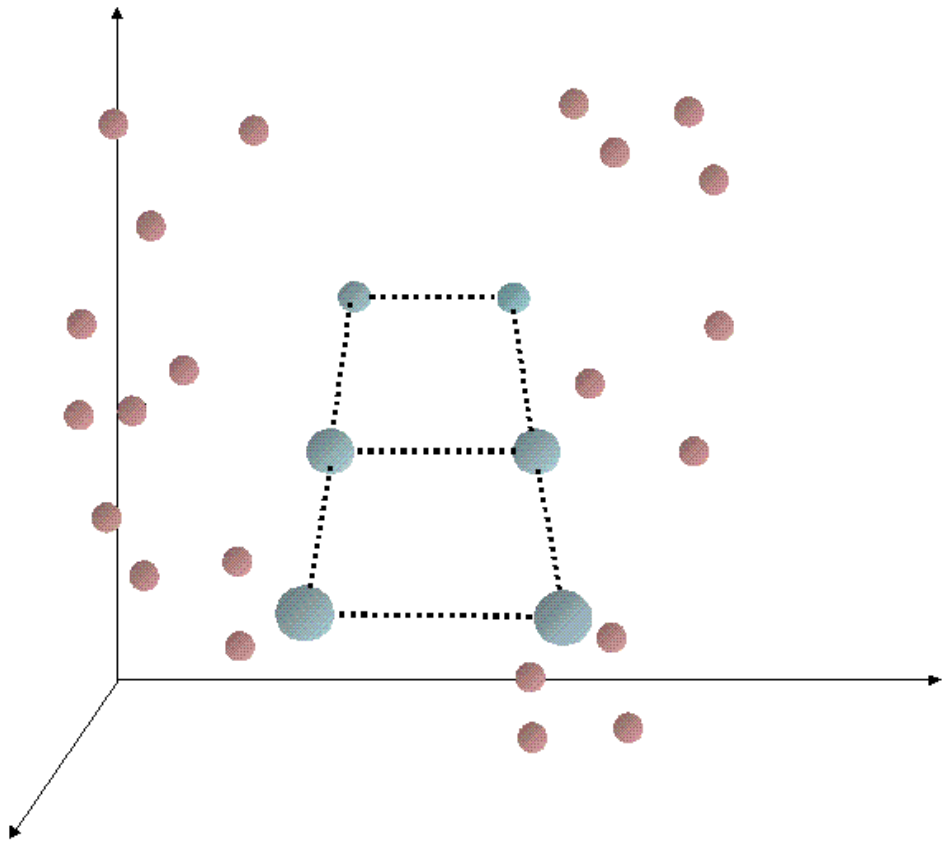

1.0
.8
.6
.4
.2
0
.2
.4
.6
.8
1.0

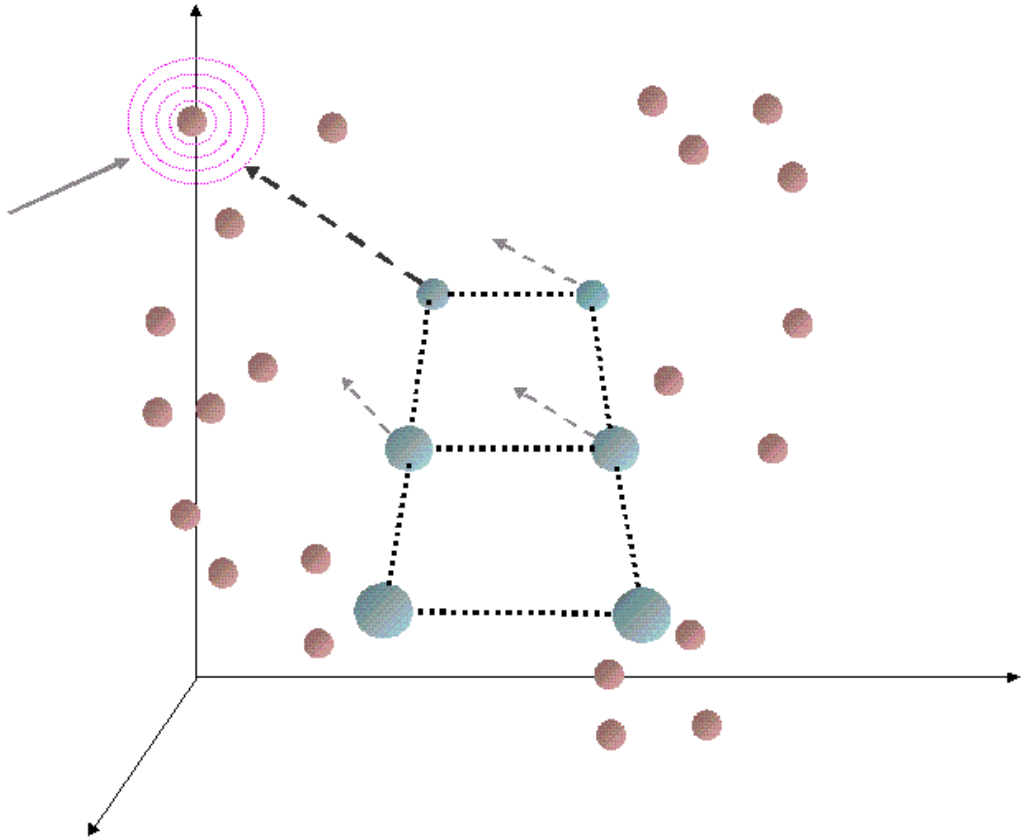

1.0
.8
.6
.4
.2
0
.2
.4
.6
.8
1.0

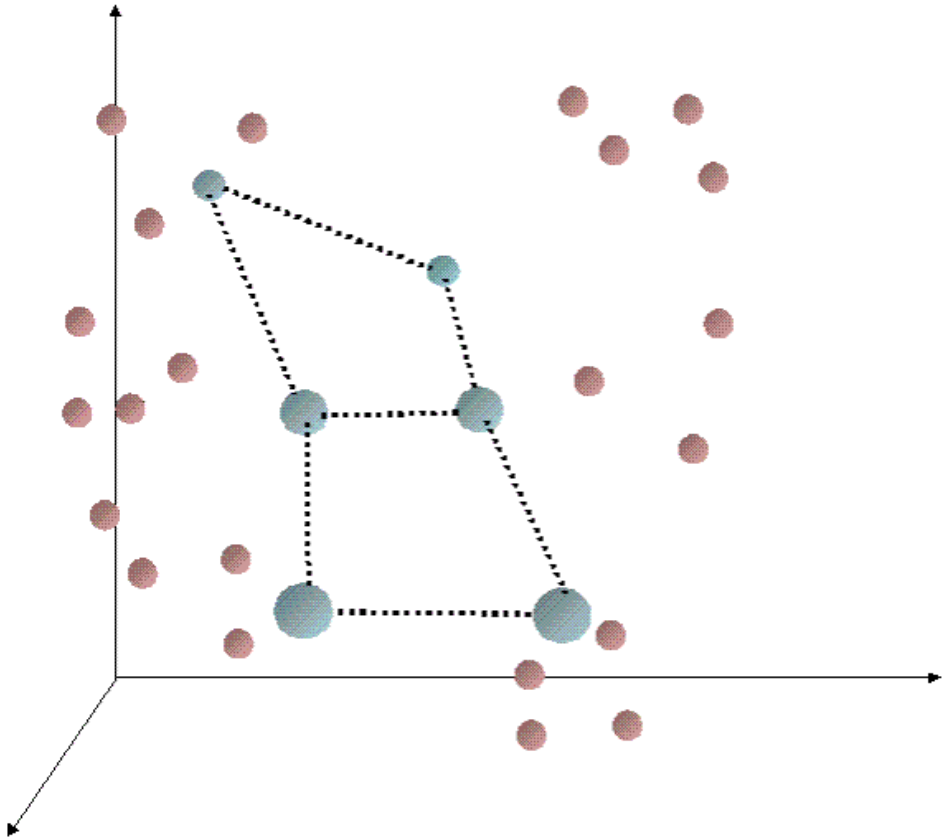

1.0
.8
.6
.4
.2
0
.2
.4
.6
.8
1.0

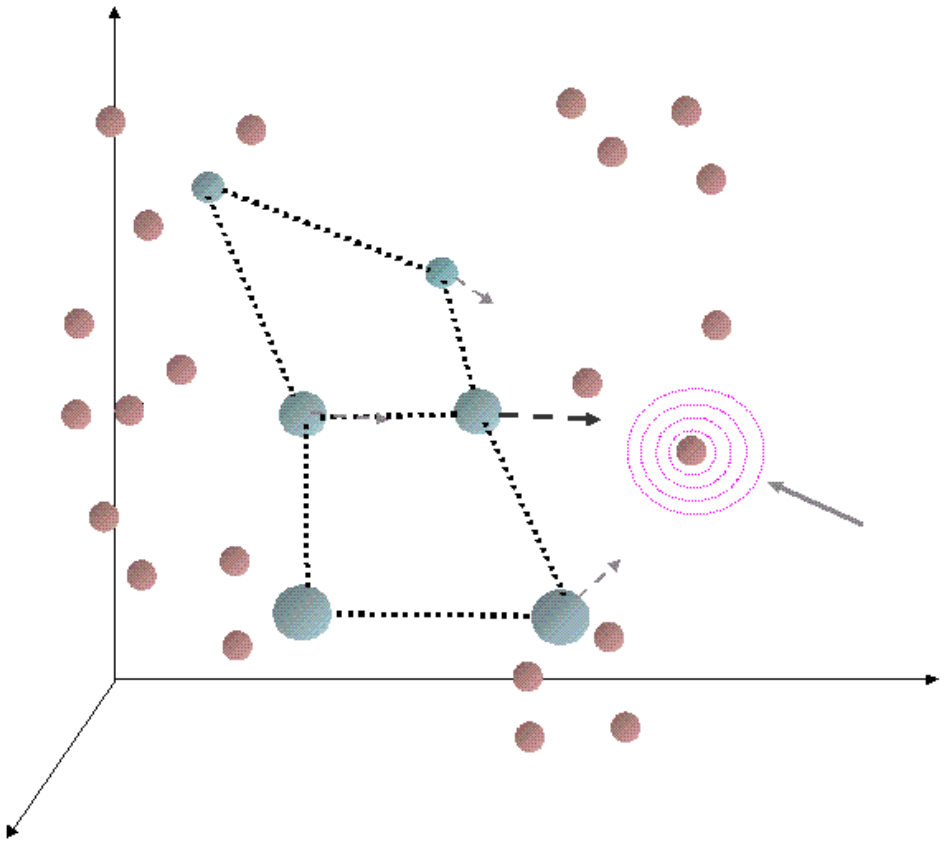

1.0
.8
.6
.4
.2
0
.2
.4
.6
.8
1.0

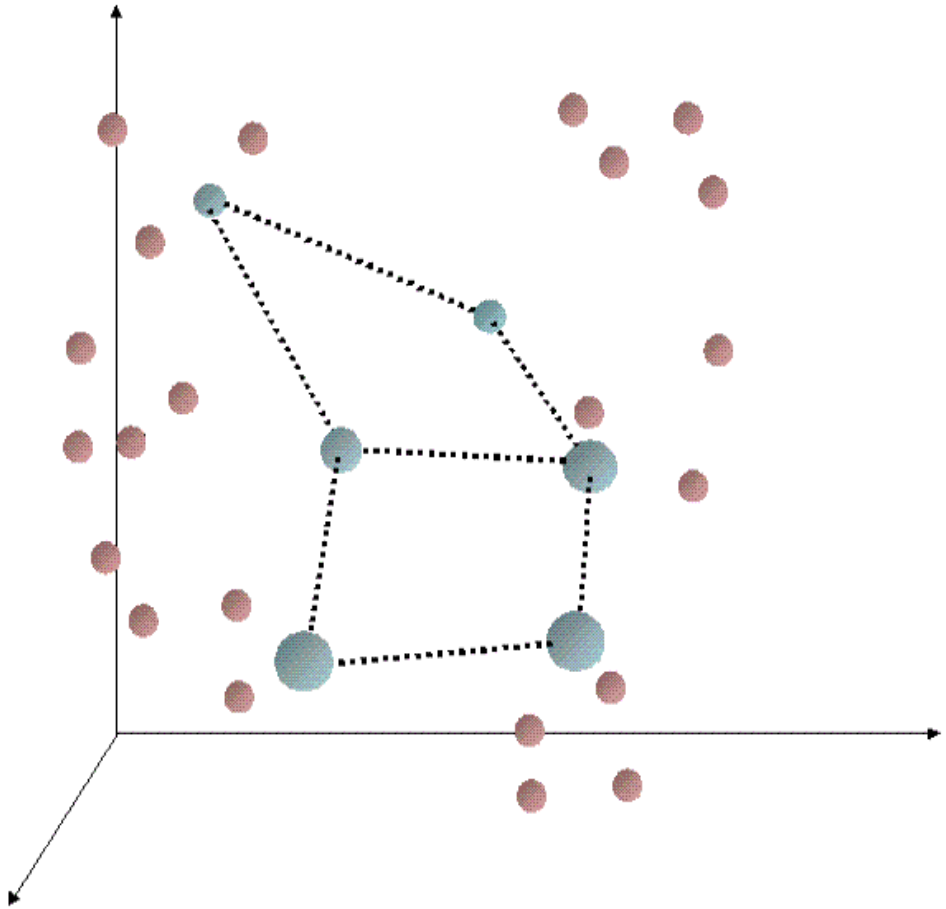

1.0
.8
.6
.4
.2
0
.2
.4
.6
.8
1.0

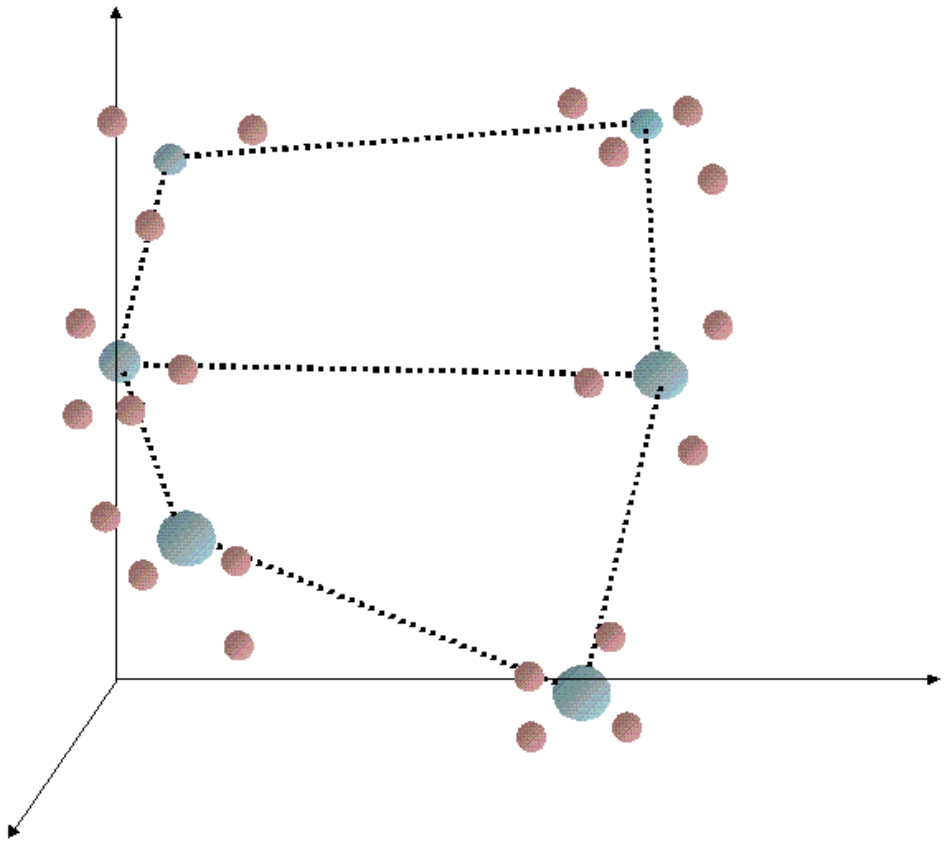




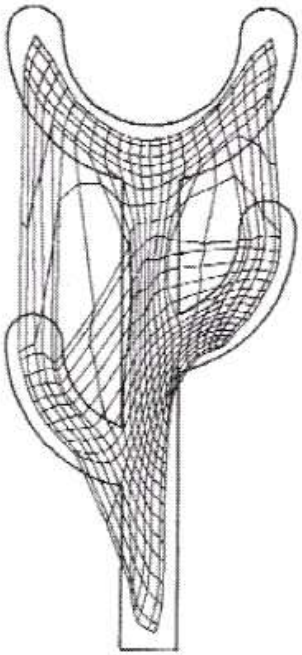


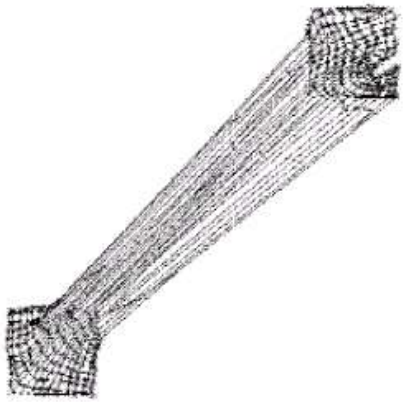




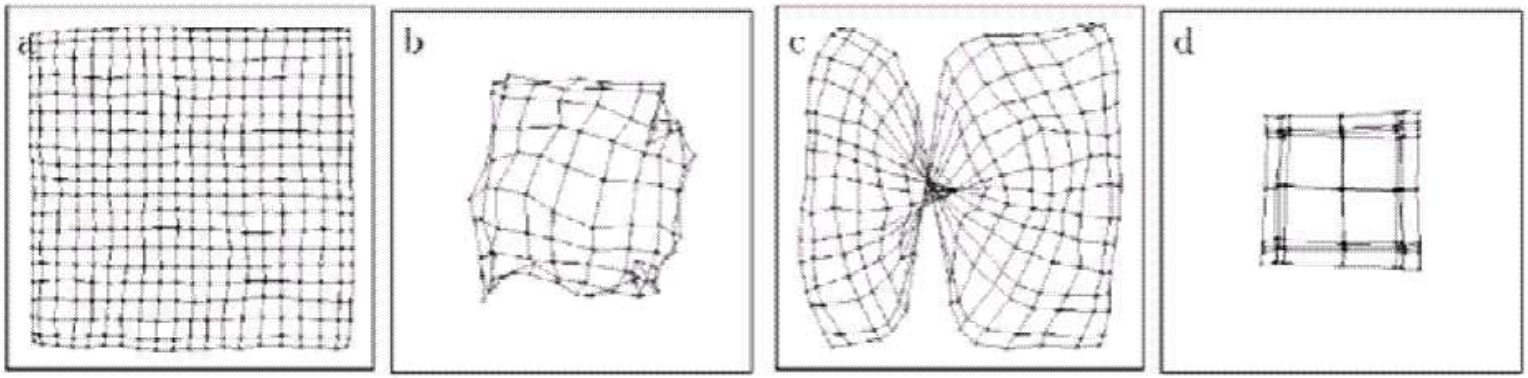


Self-Organizing Maps





Problemi di
Apprendimento



Property 1 : Approximation of the Input Space

The feature map Φ *represented by the set of weight vectors* $\{\mathbf{w}_i\}$ *in the output space, provides a good approximation to the input space.*

Property 2 : Topological Ordering

The feature map Φ *computed by the SOM algorithm is topologically ordered in the sense that the spatial location of a neuron in the output lattice/grid corresponds to a particular domain or feature of the input patterns.*

Property 3 : Density Matching

$$m(\mathbf{x}) \propto p^{2/3}(\mathbf{x})$$

The feature map Φ *reflects variations in the statistics of the input distribution: regions in the input space from which the sample training vectors* $\mathbf{x}$ *are drawn with high probability of occurrence are mapped onto larger domains of the output space, and therefore with better resolution than regions of input space from which training vectors are drawn with low probability.*

Property 4 : Feature Selection

Given data from an input space with a non-linear distribution, the self organizing map is able to select a set of best features for approximating the underlying distribution.