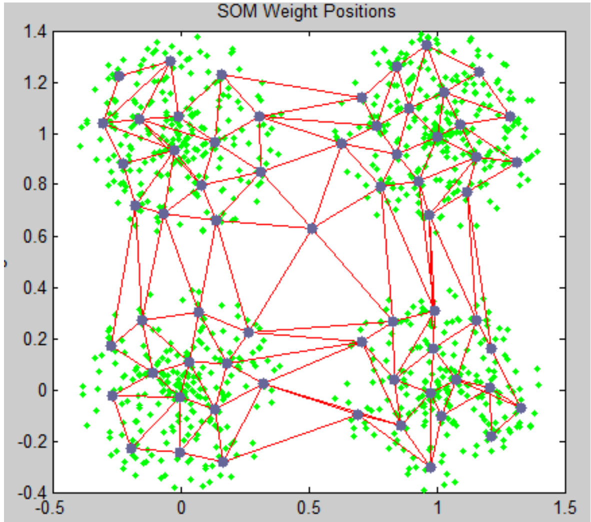
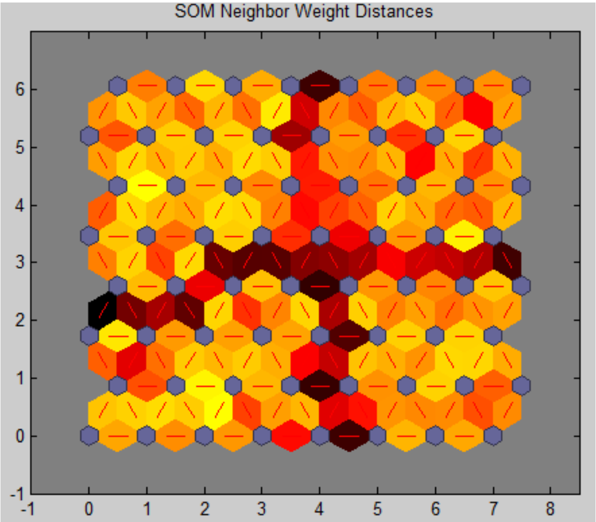
IN THE NAME OF ALLAH

Neural Networks

Self Organizing Feature Maps





Shahrood University of Technology
Hossein Khosravi

Self Organizing Maps - Fundamentals

1. What is a Self Organizing Map?
2. Topographic Maps
3. Setting up a Self Organizing Map
4. Kohonen Networks
5. Components of Self Organization
6. Overview of the SOM Algorithm

What is a Self Organizing Map?

- So far we have looked at networks with **supervised training** techniques
- We now turn to **unsupervised training**, in which the networks learn to form their own classifications of the training data without external help.
- We assume that input patterns share **common features**
- The network will identify those features across the range of input patterns.
- An interesting class of unsupervised system is based on **competitive learning**

What is a Self Organizing Map? (cnt'd)

- The output neurons **compete** amongst themselves to be activated
- Only one is activated at any time.
- This activated neuron is called the **winning neuron** (winner-takes-all neuron).
- Such competition can be implemented by having **lateral inhibition connections** between the neurons.
- The result is that the neurons **are forced to organize themselves**.
- Such a network is called a **Self Organizing Map (SOM)**.

Topographic Maps - Biological motivation

- Mapping 2D continuous inputs from sensory organ (eyes, ears, skin, etc) to two dimensional discrete outputs in the nerve system.
  - Retinotopic map: from eye (retina) to the visual cortex.
  - Tonotopic map: from the ear to the auditory cortex
- These maps preserve topographic orders of input.
- Biological evidence shows that the connections in these maps are not entirely "pre-programmed" or "pre-wired" at birth. **Learning must occur** after the birth to create the necessary connections for appropriate topographic mapping.

Topographic Maps

- **In topographic map**:
  - Neurons dealing with closely related pieces of information are kept close together so that they can interact via short synaptic connections.
  - Our interest is in building artificial topographic maps that learn through self-organization in a neurobiological inspired manner.

Setting up a Self Organizing Map

- The goal of SOM
  - **Transform** input patterns of arbitrary dimension into 1D or 2D discrete map, adaptively in a topologically ordered fashion
- We therefore set up our SOM by placing neurons at the nodes of a **one or two dimensional lattice**.
- Higher dimensional maps are also possible, but not so common

SOM Lattice

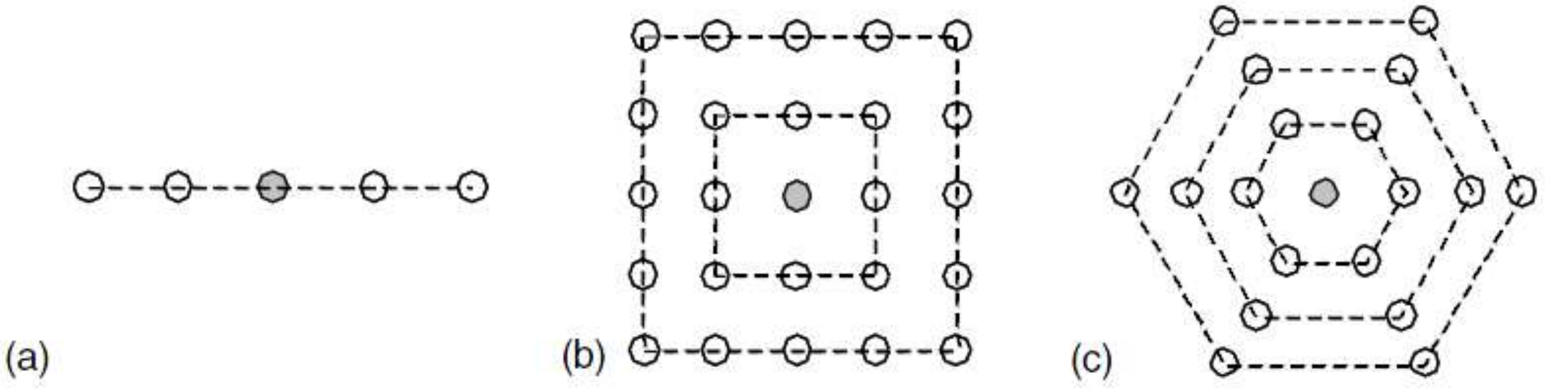
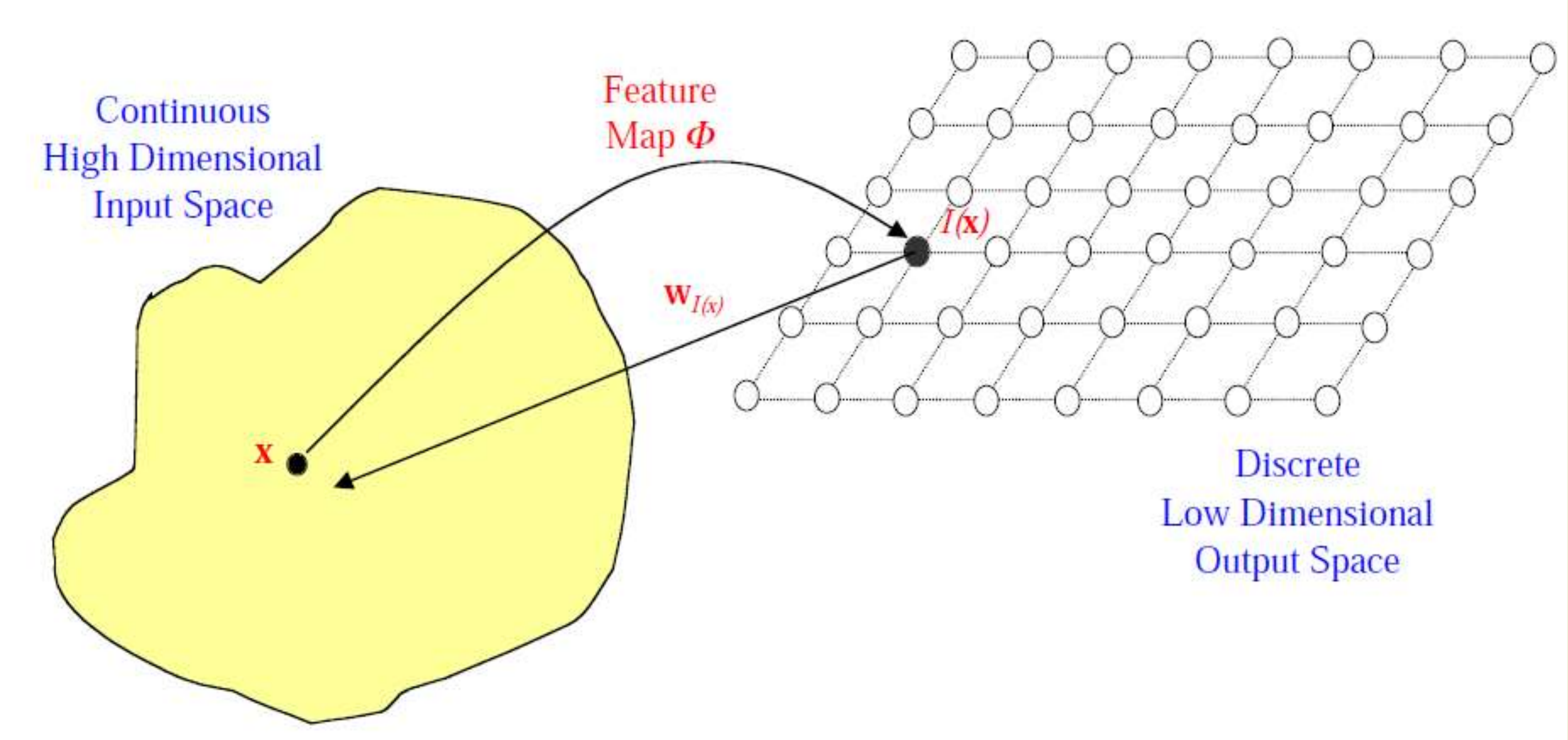


Figure 8.11 Neighborhood definitions: (a) linear (b) square, and (c) hexagonal neighborhood surrounding a winning neuron (solid circle denotes winner and empty circles denote neighbors).

Setting up a Self Organizing Map

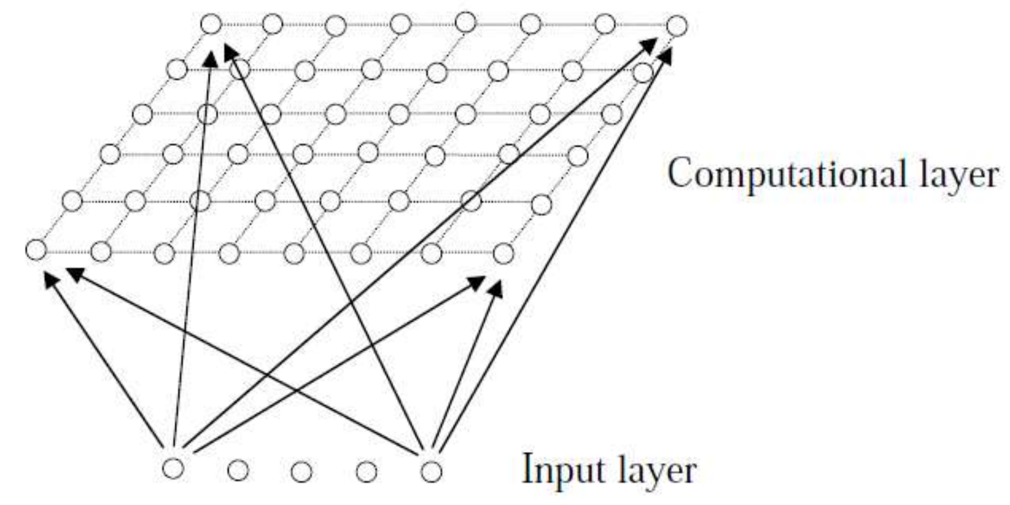
- The neurons become ***selectively tuned*** to various input patterns or classes of input patterns during the course of the competitive learning.
- The locations of the winning neurons become ordered and a meaningful ***coordinate system*** for the ***input features*** is created on the lattice.
- The SOM thus forms the required topographic map of the input patterns.

Organization of the Mapping



Kohonen Networks

- A particular kind of SOM known
- 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

Components of Self Organization

- The self-organization process involves four major components:
- **Initialization:**
  - All the connection weights are initialized with small random values.
- **Competition:**
  - For each input, the neurons compute their respective values of a *discriminant function* which provides the basis for competition.
  - Neuron with the smallest value of the discriminant function is the winner
- **Cooperation:**
  - Winning neuron determines the spatial location of a topological neighborhood of excited neurons for cooperation
- **Adaptation:**
  - Excited neurons decrease their individual values of the discriminant function, such that the response of the winning neuron to the subsequent application of a similar input pattern is enhanced.

The Competitive Process

- Assume input space is D dimensional
  - $\boldsymbol{x} = \{x_i : i = 1, \ldots, D\}$
- And the connection weights between the input units i and the neurons j in the computation layer is:
  - $\boldsymbol{w}_j = \{w_{ji} : j = 1, \ldots, N;\ i = 1, \ldots, D\}$
  - where N is the total number of neurons.
- Discriminant function
  - Can be the squared Euclidean distance between input **x** and the weight vector **w**$_j$ for each neuron *j:*

$$d_j(\mathbf{x}) = \sum_{i=1}^{D} (x_i - w_{ji})^2$$

The Competitive Process (cnt'd)

- 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 there is ***lateral interaction*** within a set of excited neurons.
- When one neuron fires, its closest neighbors tend to get excited more than those further away.
- There is a ***topological neighborhood*** that decays with distance.
- Similar topological neighborhood in SOM:
  - If S_{ij} is the lateral distance between neurons *i* and *j* on the grid of neurons, we take the following as our topological neighborhood

$$T_{j,I(\mathbf{x})} = \exp(-S_{j,I(\mathbf{x})}^2 / 2\sigma^2)$$

The Cooperative Process (cnt'd)

- $I(\mathbf{x})$ is the index of the winning neuron.
- This has 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)

$$T_{j,I(\mathbf{x})} = \exp(-S_{j,I(\mathbf{x})}^2 / 2\sigma^2)$$

- A special feature of the SOM is that the size σ of the neighborhood needs to decrease with time.
- A popular time dependence is an exponential decay:
  - $\sigma(t) = \sigma_0 \exp(-t/\tau_\sigma)$.
  - σ_0 = radius of the lattice
  - $\tau_\sigma = 1000/\log \sigma_0$

The Adaptive Process

- SOM must involve some kind of adaptive, or learning process, by which the outputs become self-organized and the feature map between inputs and outputs is formed.

- The point of the topographic neighborhood is that not only the winning neuron gets its weights updated, but its neighbors will have their weights updated as well, although by not as much as the winner itself.

The Adaptive Process (cnt'd)

- Weight update:
  - $\Delta wji = \eta(t)\, T_{jI(x)}(t)\, (xi - wji)$
  - Where $\eta(t)$ is a time (epoch) dependent learning rate $\eta(t) = \eta_0 \exp(-t/\tau_\eta)$
  - $0.01 < \eta < 0.1$ (e.g. $\eta_0 = 0.1$ and $\tau_\eta = 1000$)

- Updates are applied for all the training patterns **x** over many epochs.

- The effect of each learning weight update is to move the weight vectors **w***i* of the winning neuron and its neighbors towards the input vector **x**.

- Repeated presentations of the training data thus leads to topological ordering.

Remarks

- Reduction of neighboring distance *T* (or D) should be slower than η reduction
- D can be a constant through out the learning
- Effect of learning
  - For each input i, not only the weight vector of winner is pulled closer to i, but also the weights of close neighbors (within the radius of ***D***).
- Eventually, W_j becomes close (similar) to $W_{j\pm1}$. The classes they represent are also similar.
- May need large initial ***D*** in order to establish topological order of all nodes

Remarks (cnt'd)

- Find j^* for a given input i_l:
  - With minimum distance between w_j and i_l.
  - Distance: $\text{dist}(w_j, i_l) = \left\| w_j - i_l \right\|_2 = \sum_{k=1}^{n} (i_{l,k} - w_{j,k})^2$
  - Minimizing dist(w_j, i_l) can be realized by maximizing

$$o_j = i_l \cdot w_j = \sum_k w_{j,k} \cdot i_{l,k}$$

  - Assume that i_l *and* w_j *have* *unity magnitude*

$$\text{dist}(w_j, i_l) = \sum_{k=1}^{n} (i_{l,k}^2 + w_{j,k}^2 - 2 i_{l,k} \cdot w_{j,k})$$

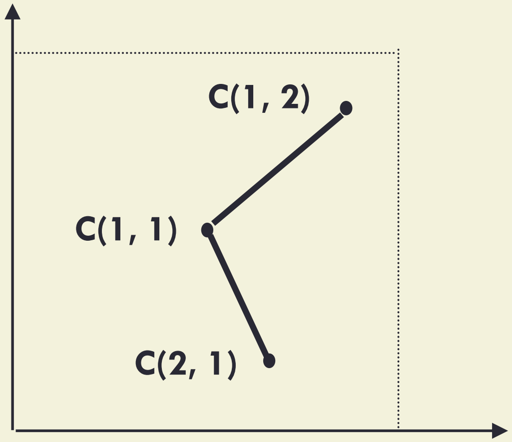
$$= \sum_{k=1}^{n} i_{l,k}^2 + \sum_{k=1}^{n} w_{j,k}^2 - 2 \sum_{k=1}^{n} i_{l,k} \cdot w_{j,k}$$

$$= -2 \sum_{k=1}^{n} i_{l,k} \cdot w_{j,k} + 2 \quad = \quad -2 i_l \cdot w_j + 2$$

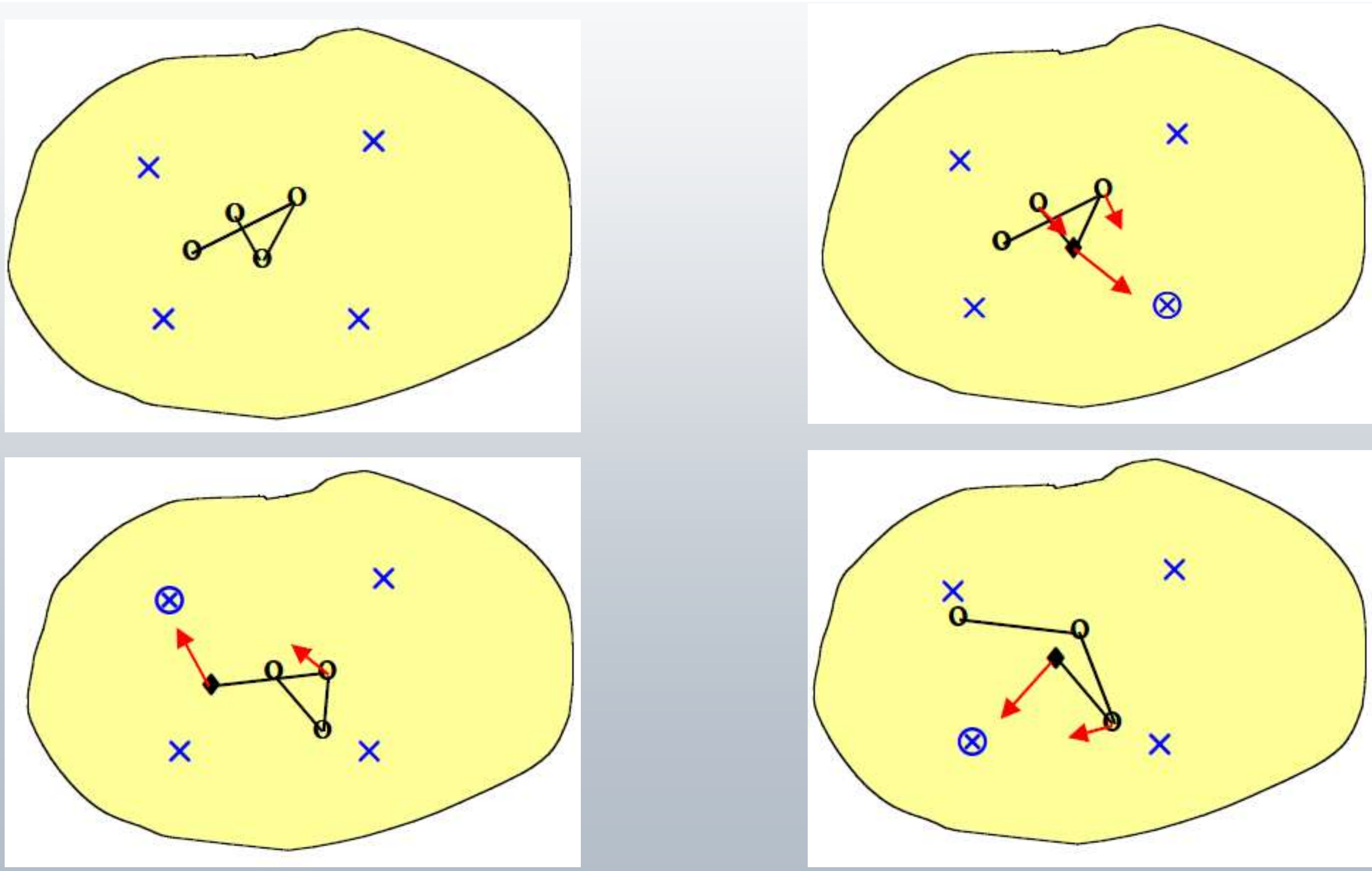
How to illustrate Kohonen map (for 2-D patterns)

- Input vector: 2 dimensional
  - Output vector: 1 dimensional line/ring or 2 dimensional grid.
  - Weight vector is also 2 dimensional
- Represent the topology of output nodes by points on a 2 dimensional plane. Plotting each output node on the plane with its weight vector as its coordinates.
- Connecting neighboring output nodes by a line

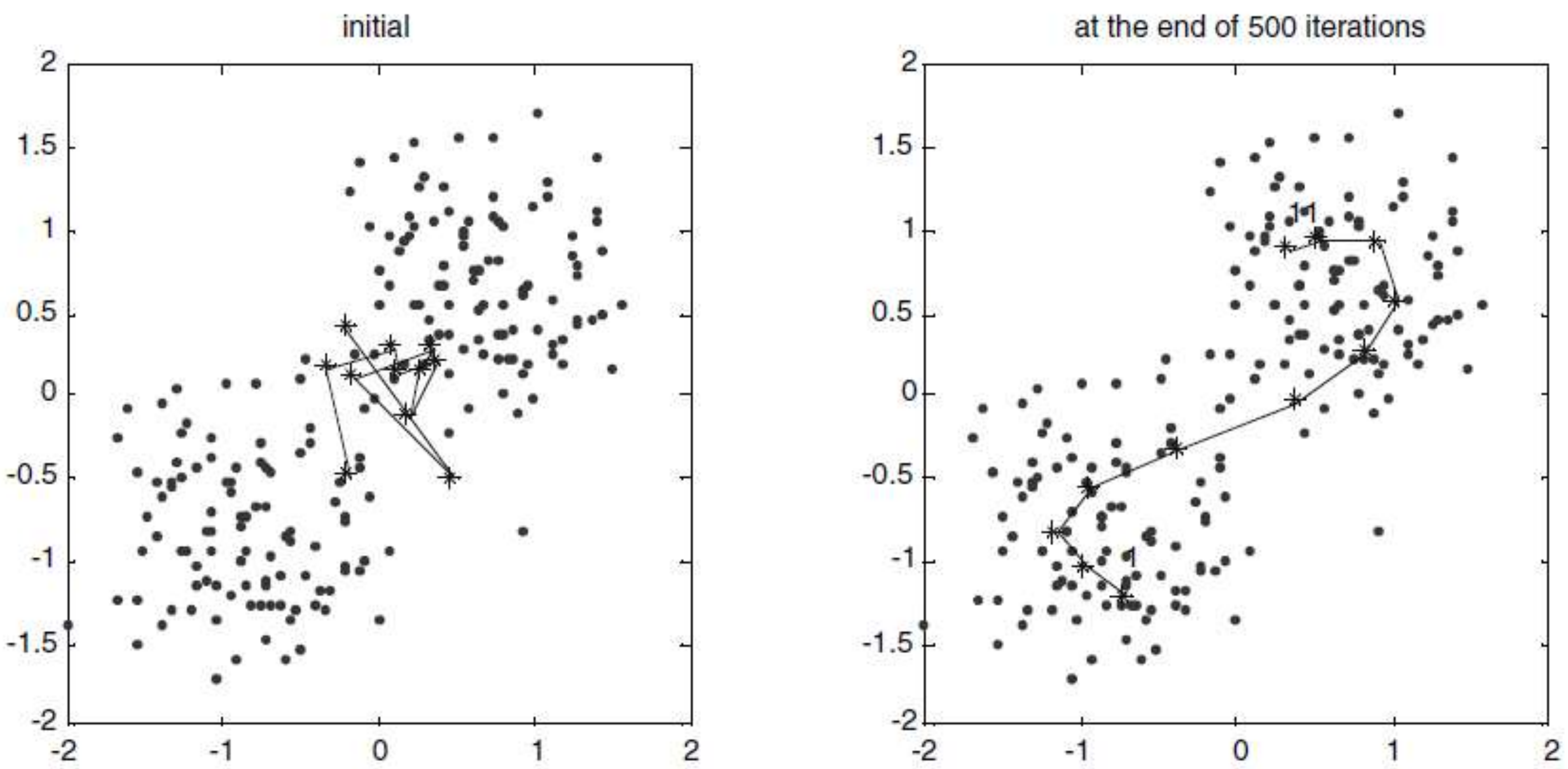
output nodes:	(1, 1)	(2, 1)	(1, 2)
weight vectors:	(0.5, 0.5)	(0.7, 0.2)	(0.9, 0.9)



SOM Visualization - Linear



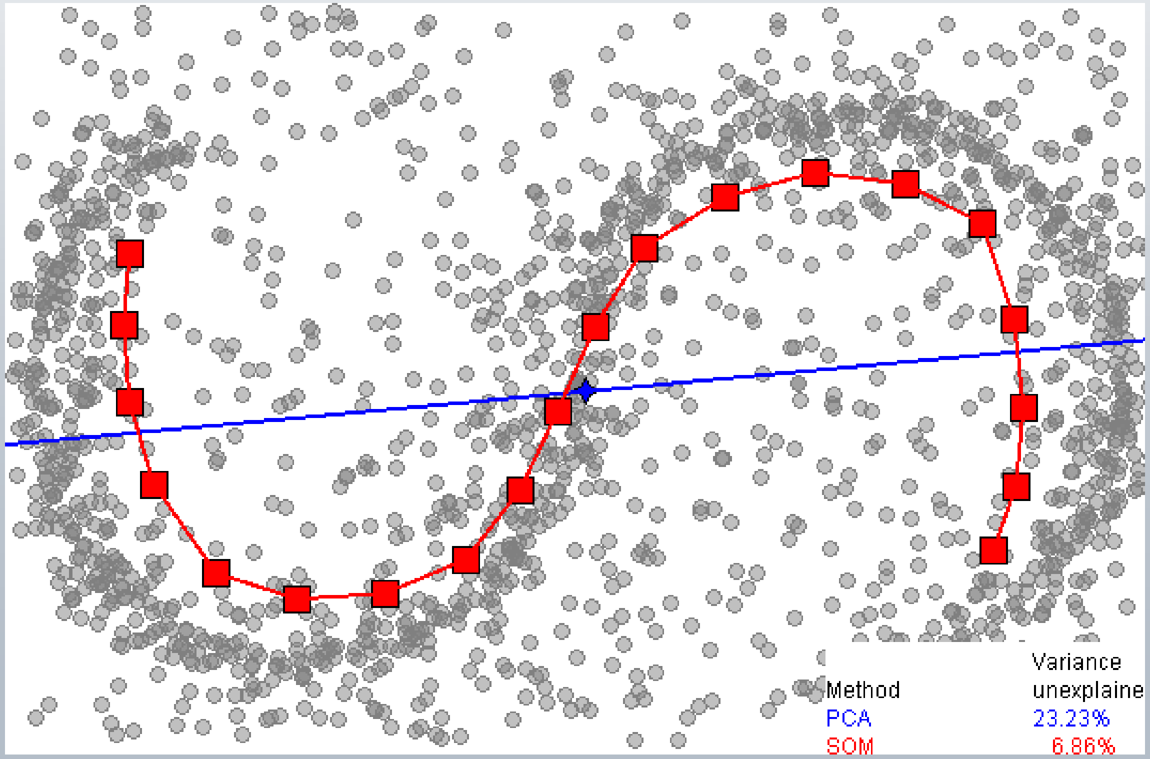
SOM Illustration - Linear



1.16 The neurons are initially placed randomly in the feature space as shown to the left of the figure. After 500 iterations, they are distributed evenly to represent the underlying feature vector distribution.

SOM vs. PCA

One-dimensional SOM versus principal component analysis (PCA) for data approximation. SOM is a red broken line with squares, 20 nodes. The first principal component is presented by a blue line. Data points are the small grey circles. For PCA, the fraction of variance unexplained in this example is 23.23%, for SOM it is 6.86%.



Ordering and Convergence

- Having the parameters (σ_0, τ_σ, η_0, τ_η) 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.

- However, it is possible to end up in a **metastable state** in which the feature map has a topological defect.

Ordering and Convergence

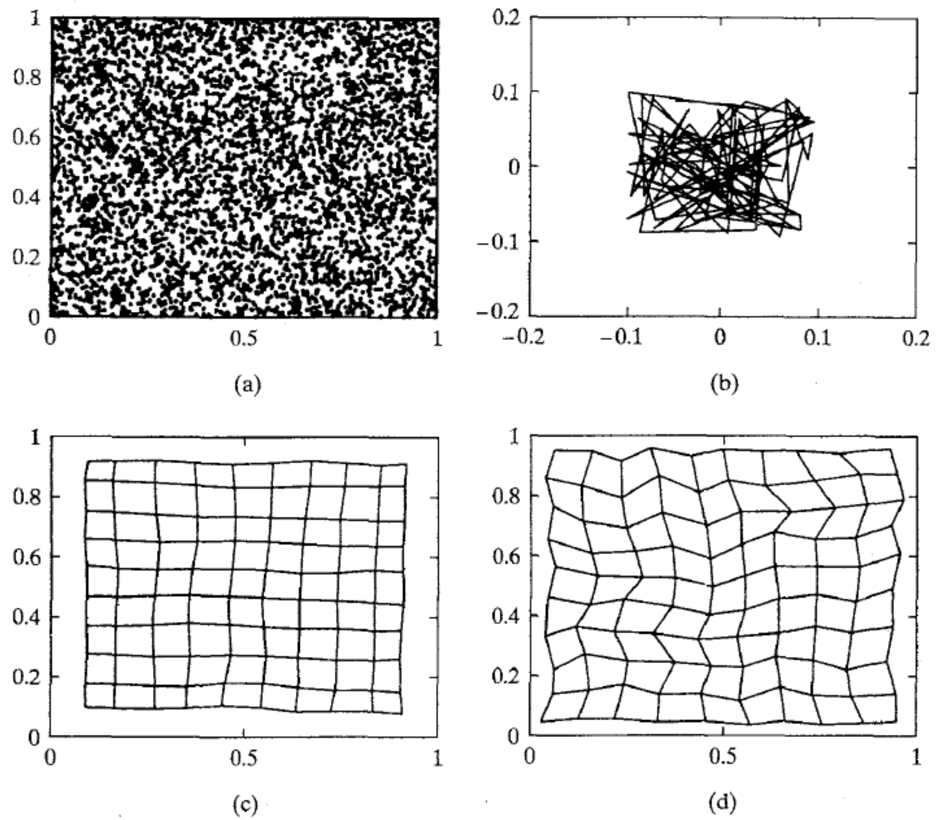


FIGURE 9.8 (a) Input data distribution. (b) Initial condition of the two-dimensional lattice. (c) Condition of the lattice at the end of the oidering phase. (d) Condition of the lattice at the end of the convergence phase.

Ordering and Convergence

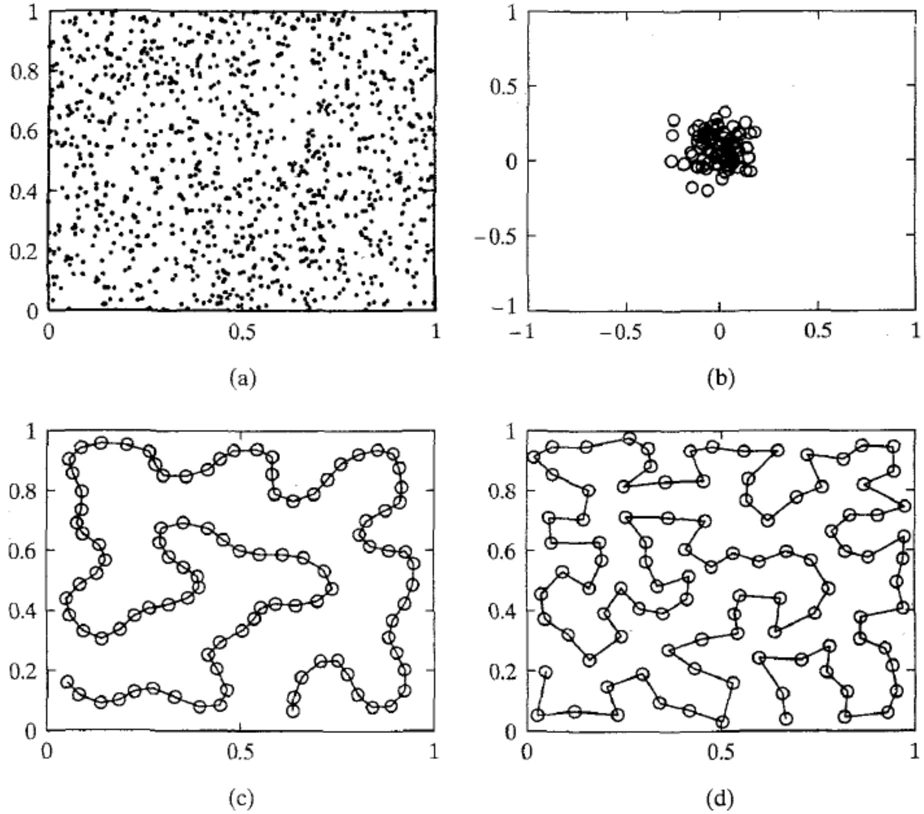


FIGURE 9.9 (a) Two-dimensional input data distribution. (b) Initial condition of the one-dimensional lattice. (c) Condition of the lattice at the end of the ordering phase. (d) Condition of the lattice at the end of the convergence phase.

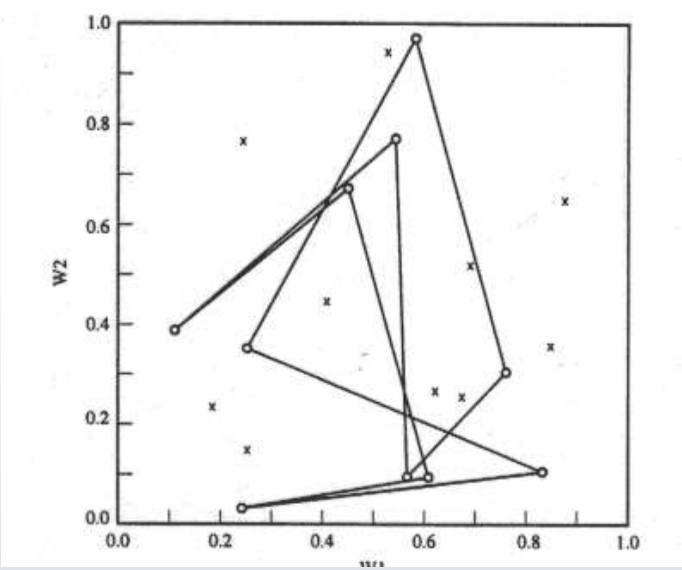
Traveling Salesman Problem (TSP)

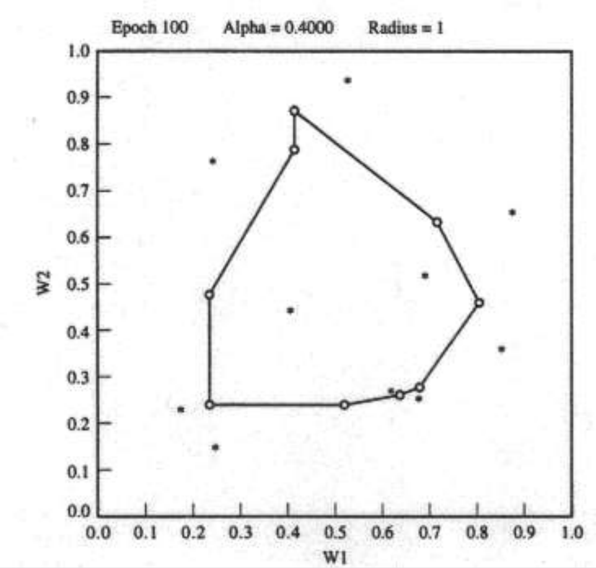
- Given a road map of n cities, find the **shortest** tour which visits every city on the map exactly once and then return to the original city
- (Geometric version):
  - A complete graph of n vertices on a unit square.
  - Each city is represented by its coordinates(x_i, y_i)
  - $n!/2n$ legal tours
  - Find one legal tour that is shortest

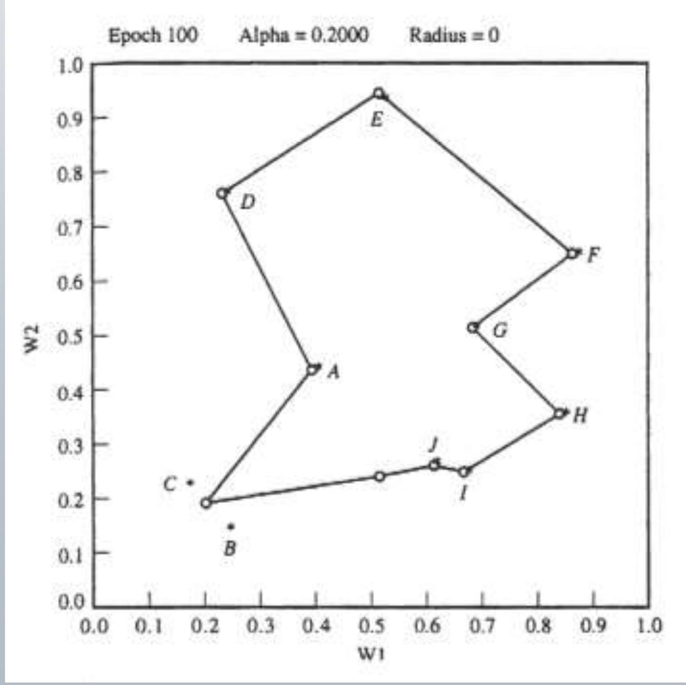
Approximating TSP by SOM

- Each city is represented as a 2 dimensional **input** vector (x_i, y_i)
- Output nodes C_j, form a SOM of one dimensional **ring**, $(\boldsymbol{C_1}, C_2, \dots, C_n, \boldsymbol{C_1})$.
- Initially, $\boldsymbol{C_1}, C_2, \dots, C_n$ have random weight vectors, so we don't know how these nodes correspond to individual cities.
- During learning, a winner C_j on an input (x_i, y_i) of city i, not only moves its w_j toward (x_i, y_i), but also that of its neighbors w_{j+1} and w_{j-1}.
- As a result, C_{j-1} and C_{j+1} will later be more likely to win with input vectors similar to (x_i, y_i) , i.e. those cities closer to i.
- At the end, if a node j represents city i, it would end up to have its neighbors $j+1$ or $j-1$ to represent cities similar to city i (i,e., cities close to city i).

Initial position







Two candidate solutions:

ADEFGHIJBC

ADEFGHIJCB

SOM Architecture in brief

- Two layer network:
  - Output layer:
    - Each node represents a class (of inputs)
    - Node function: $o_j = i_l \cdot w_j = \sum_k w_{j,k} \cdot i_{l,k}$
    - Neighborhood relation is defined over these nodes
      - $N_j(t)$: set of nodes within distance $D(t)$ to node j.
    - Each node cooperates with all its neighbors and competes with all other output nodes.
    - Cooperation and competition of nodes

      $D = 0$: all nodes are competitors (no cooperative)
      → random map

      $D > 0$: → topology preserving map

Algorithm SelfOrganize;

- Select network topology (neighborhood relation);
- Initialize weights randomly, and select $D(0) > 0$;
- while computational bounds are not exceeded, do

1. Select an input sample i_ℓ;

2. Find the output node $j*$ with minimum $\sum_{k=1}^{n}(i_{\ell,k}(t) - w_{j,k}(t))^2$;

3. Update weights to all nodes within a topological distance of $D(t)$ from $j*$, using

$$w_j(t+1) = w_j(t) + \eta(t)(i_\ell(t) - w_j(t)),$$

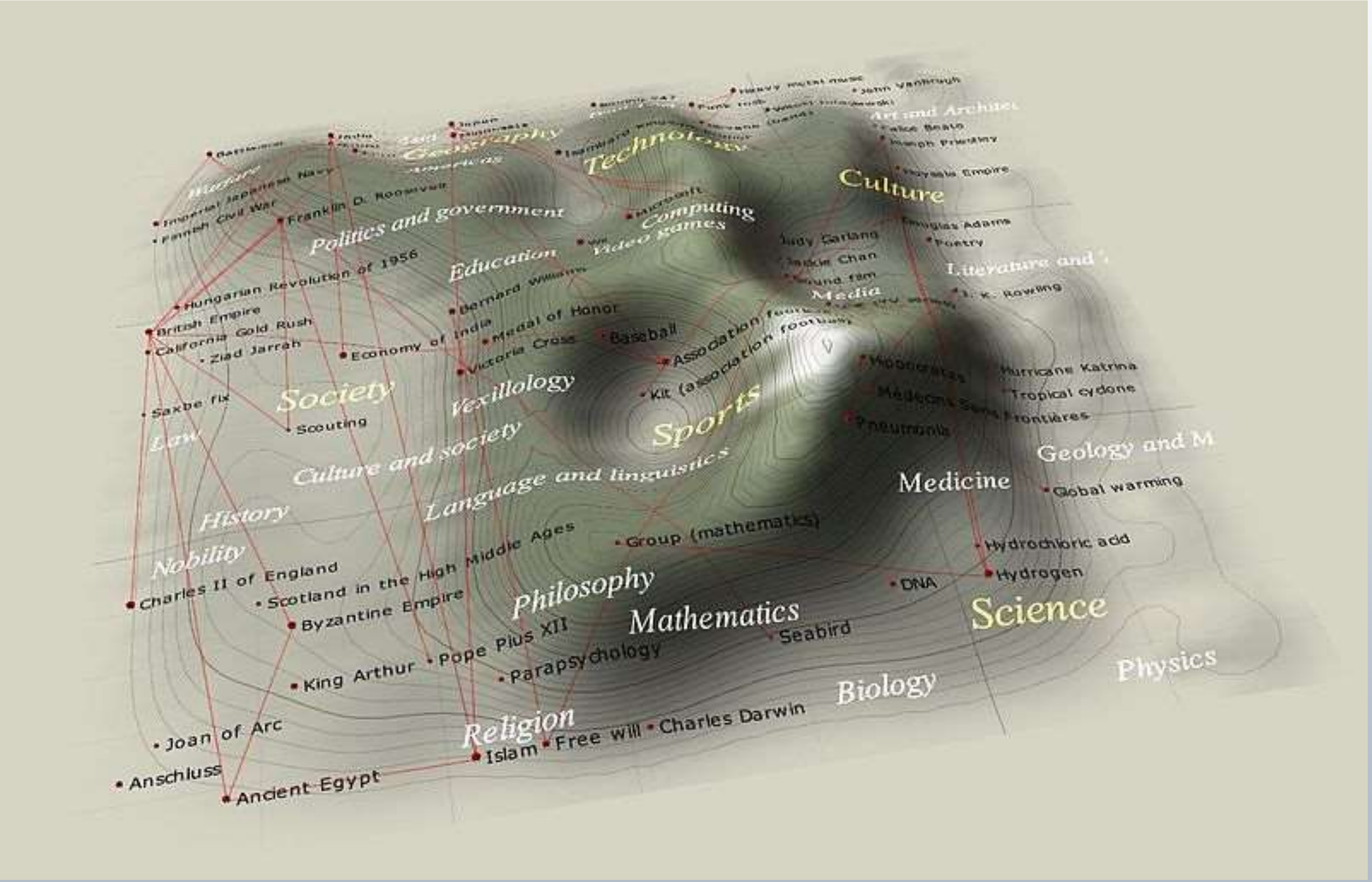
where $0 < \eta(t) \leq \eta(t-1) \leq 1$;

4. Increment t;

end-while.

Example:

Cartographical representation of a self-organizing map (U-Matrix) based on Wikipedia featured article data (word frequency). Distance is inversely proportional to similarity. The "mountains" are edges between clusters. The red lines are links between articles.



- Run SOM.m
- newlvq, selforgmap, newsom (obsoleted)