CSE 5526: Introduction to Neural Networks

Unsupervised learning and Self-organizing maps

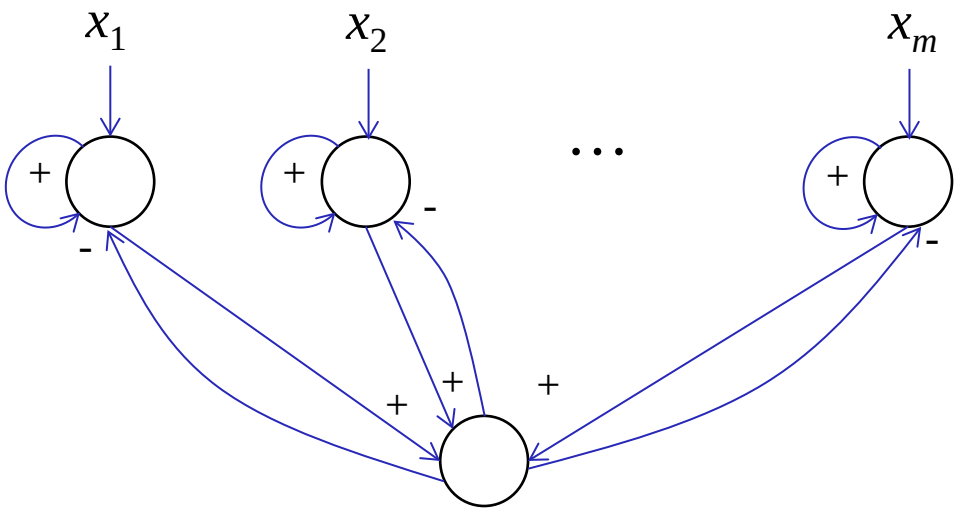
Types of learning

- Supervised learning: Detailed desired output is provided externally
- Reinforcement learning: Desired end state of an interaction with environment is provided
  - Learn best actions to take to get there
- Unsupervised learning: Discover structure in data
  - E.g., competitive learning and self organization

Winner-take-all (WTA) networks implement competitive dynamics

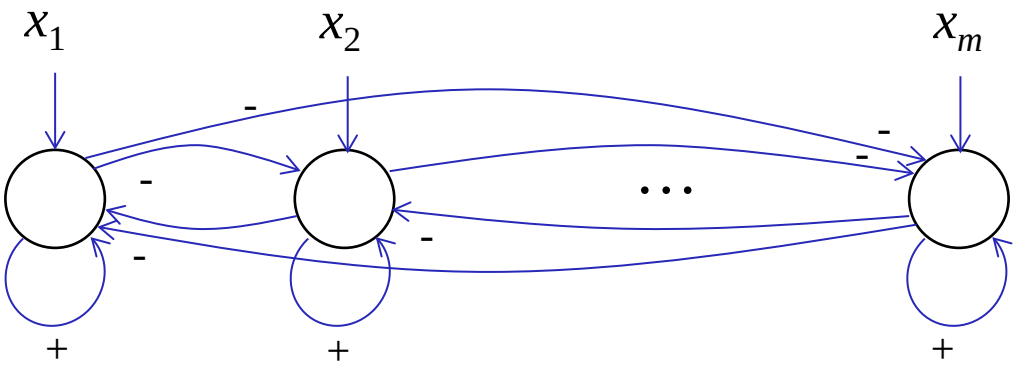
- Recurrent neural network
  - Each neuron excited by input
  - Recurrent dynamics eventually lead to one “winner”
  - Update winning neuron to be more sensitive to that input
- Similar to *K*-means algorithm
- Two different architectures
  - Global inhibition
  - Mutual inhibition

Global inhibition has simple dynamics



- Requires $3m$ inter-connections for m neurons

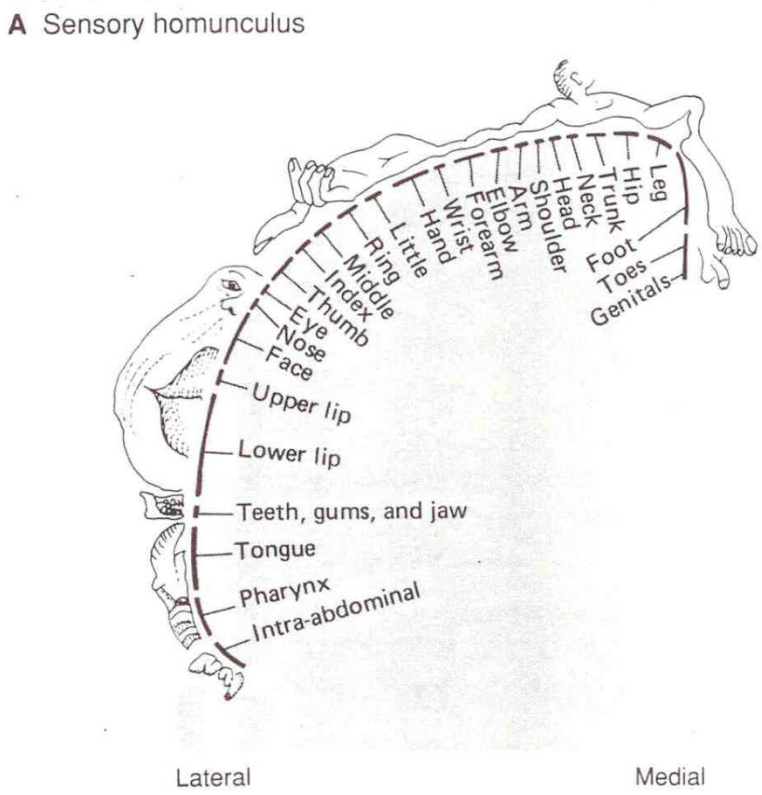
Mutual inhibition has complex dynamics



- Requires m^2 inter-connections for m neurons

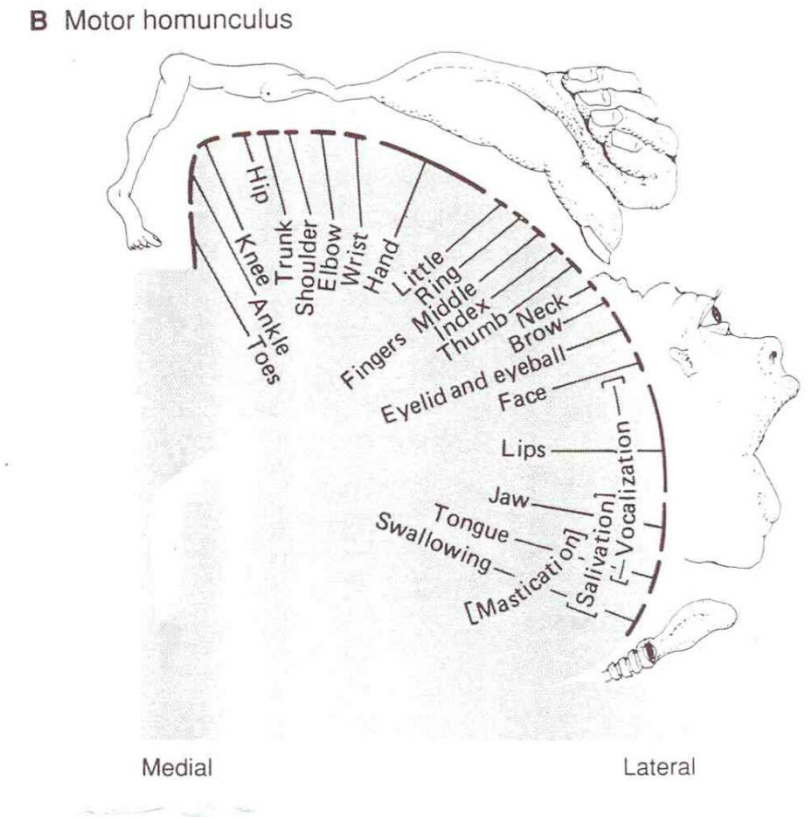
Spatial position in a brain area frequently maps to meaningful perceptual dimensions

- Maps are commonly found in the brain
  - Retinotopic maps
  - Tonotopic maps
  - Somatosensory (tactile) maps
  - Motor maps



Spatial position in a brain area frequently maps to meaningful effector dimensions

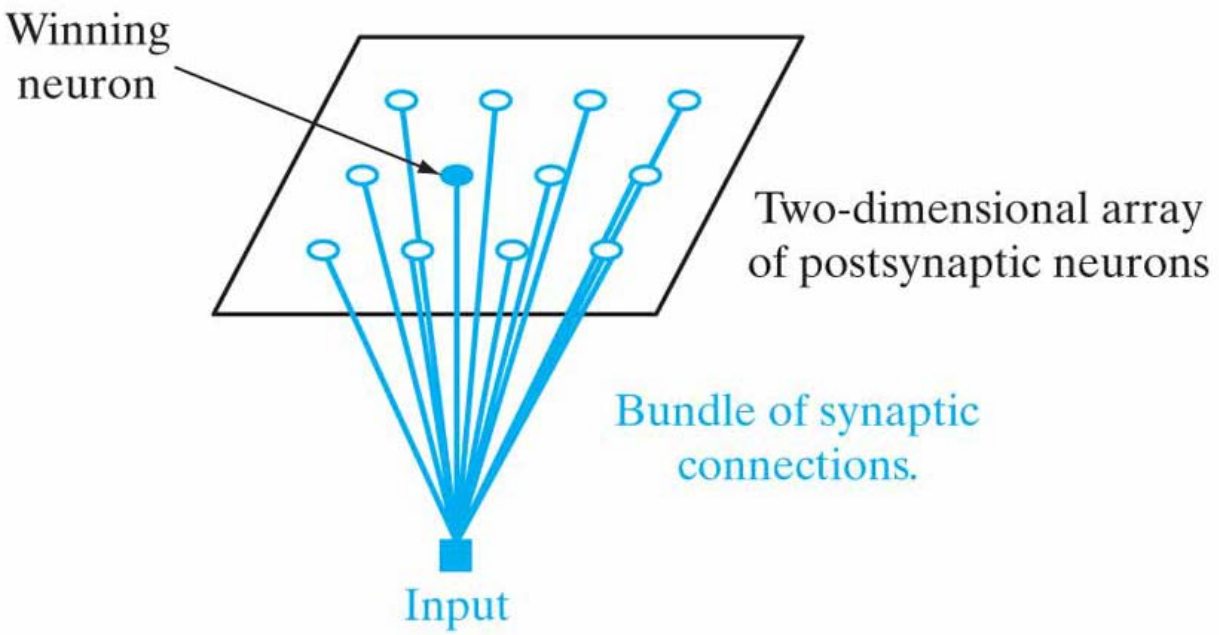
- Maps are commonly found in the brain
  - Retinotopic maps
  - Tonotopic maps
  - Somatosensory (tactile) maps
  - Motor maps



Spatial maps can arise from simple learning rules

- Topological maps in the brain could arise from many factors
  - Genetics via chemical gradients
  - Spontaneous firing of neurons
  - Firing in response to sensory inputs
- It is possible to create them in artificial neural networks based only on “sensory” inputs
  - For example in self-organizing maps (SOMs)
  - Like *K*-means, but with organized means

A self-organizing map is a WTA network with a notion of distance between neurons

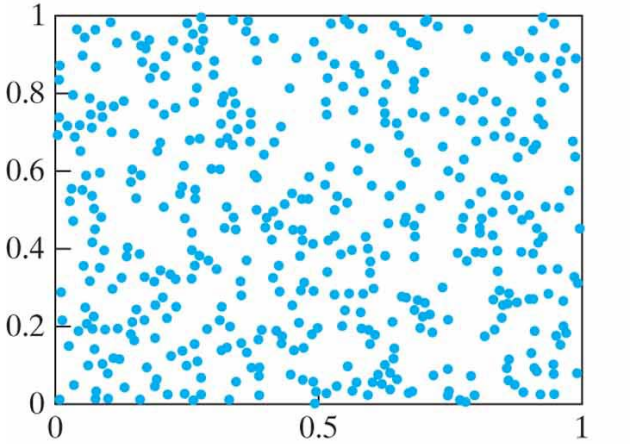


(b) Kohonen model

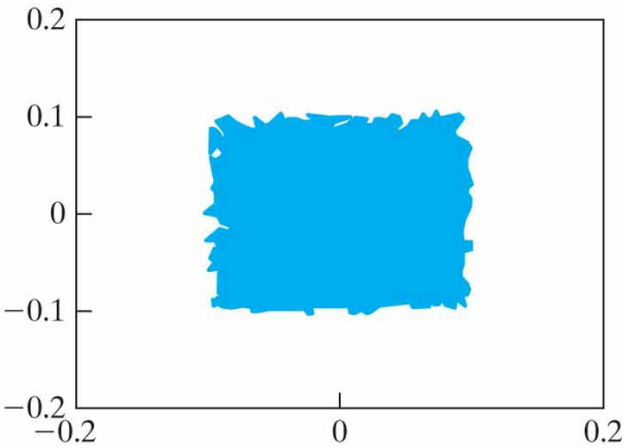
A self-organizing map is a WTA network with a notion of distance between neurons

- Each node in the SOM has a prototype vector
  - Computes activation based on distance to an input
  - What it's looking for or excited by
- Each node in the SOM has a set of neighbors
  - Or a distance function to the rest of the neurons
- Learning in the SOM adjusts the prototypes
  - So that neurons that are "close" to each other have prototypes that are "close" to each other
- Learns a nonlinear dimensionality reduction

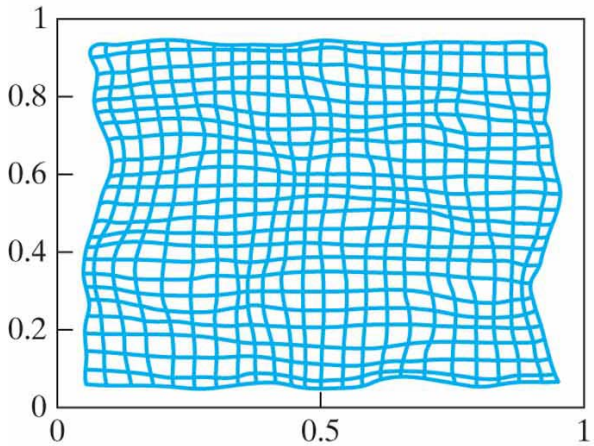
2D SOM learning 2D data



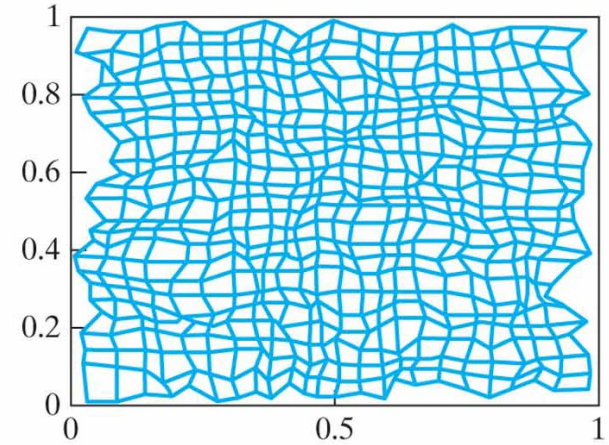
(a) Input distribution



Time = 0
(b) Initial weights

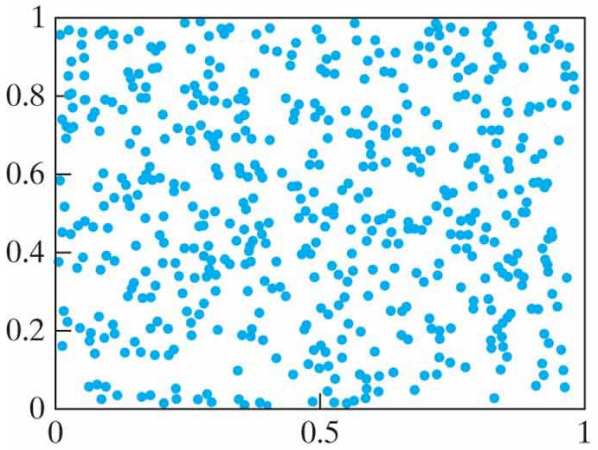


Time = 160 K
(c) Ordering phase

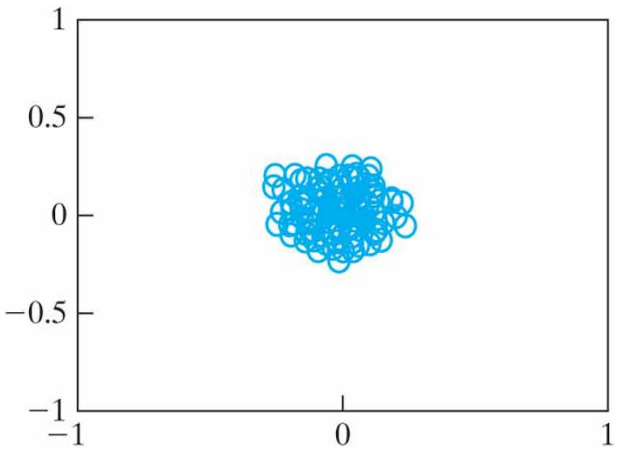


Time = 800 K
(d) Convergence phase

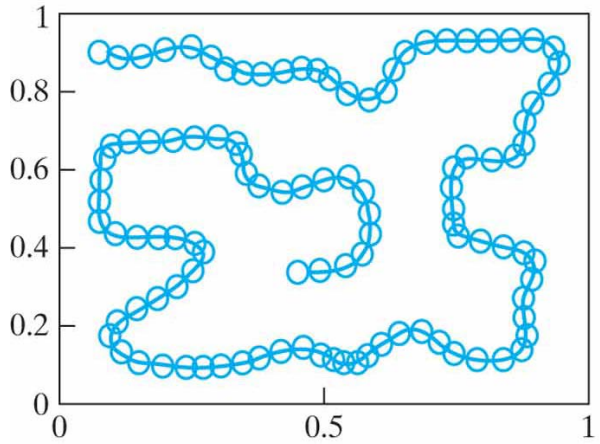
1D SOM learning 2D data



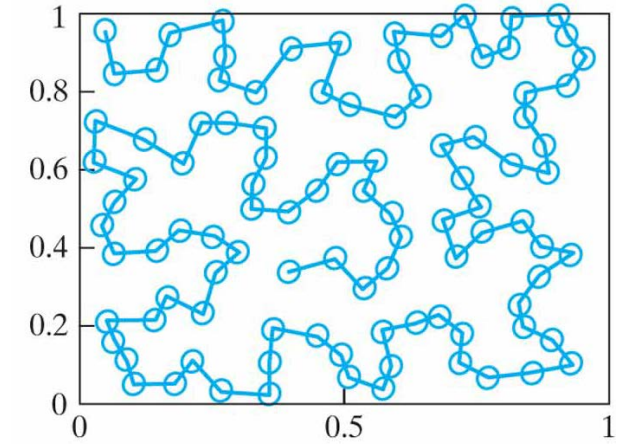
(a) Input distribution



Time = 0
(b) Initial weights



Time = 50 K
(c) Ordering phase



Time = 100 K
(d) Converging phase

SOM training

- Activate neurons based on distance to inputs
  - Find winner, the neuron most activated
- Update neurons based on distance to winner
  - Winner's prototype is updated to be closer to input
  - Neighbors' prototypes are updated less
  - Far away neurons are not updated
- No global objective being optimized
  - But interesting behavior in practice

SOM training equations

- Weight update

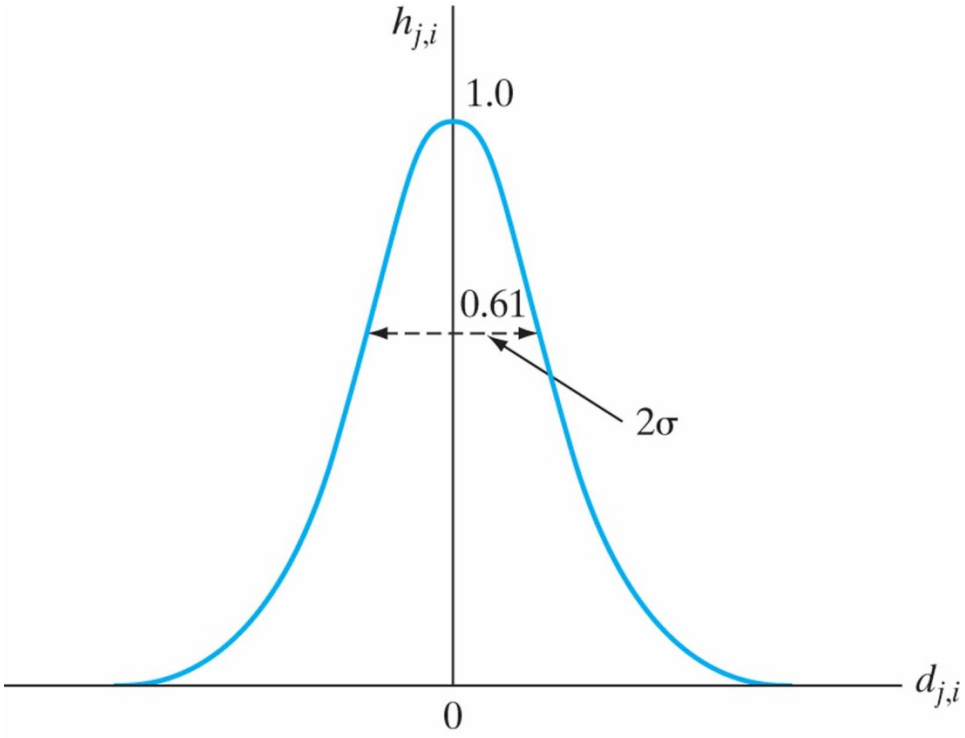
$$\mathbf{w}_j(n+1) = \mathbf{w}_j(n) + \eta(n) h_{j,i(\mathbf{x})}(n)\left[\mathbf{x}(n) - \mathbf{w}_j(n)\right]$$

  - $i(\mathbf{x})$ indicates the winning neuron
  - $h_{j,i}$ denotes a neighborhood function centered at neuron i
- A typical choice for $h_{j,i}$ is a Gaussian function

$$h_{j,i}(n) = \exp\left[-\frac{{d_{j,i}}^2}{2\sigma^2(n)}\right]$$

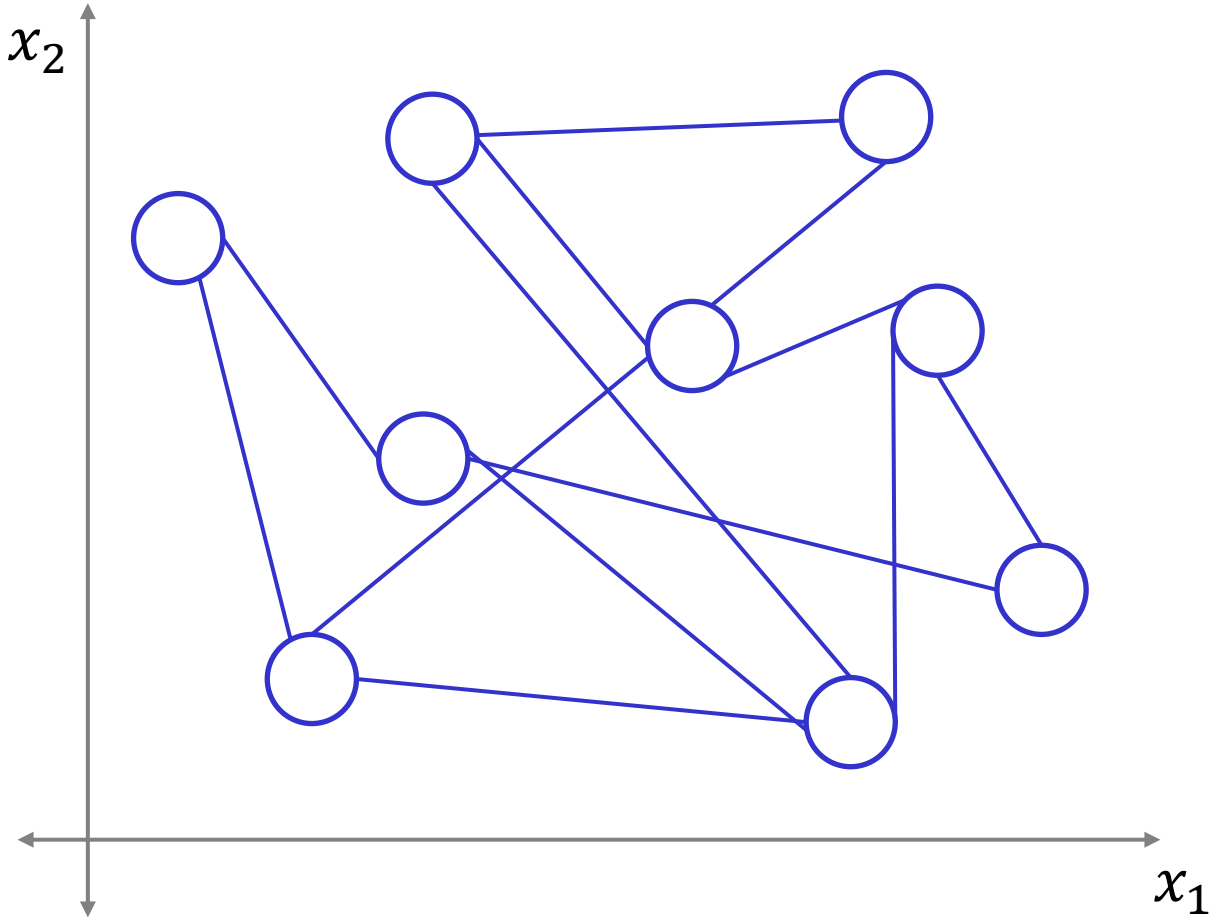
  - $d_{j,i}$ denotes the Euclidean distance between neuron j and i on the output layer

Gaussian $h_{j,i}$ as a function of $d_{j,i}$

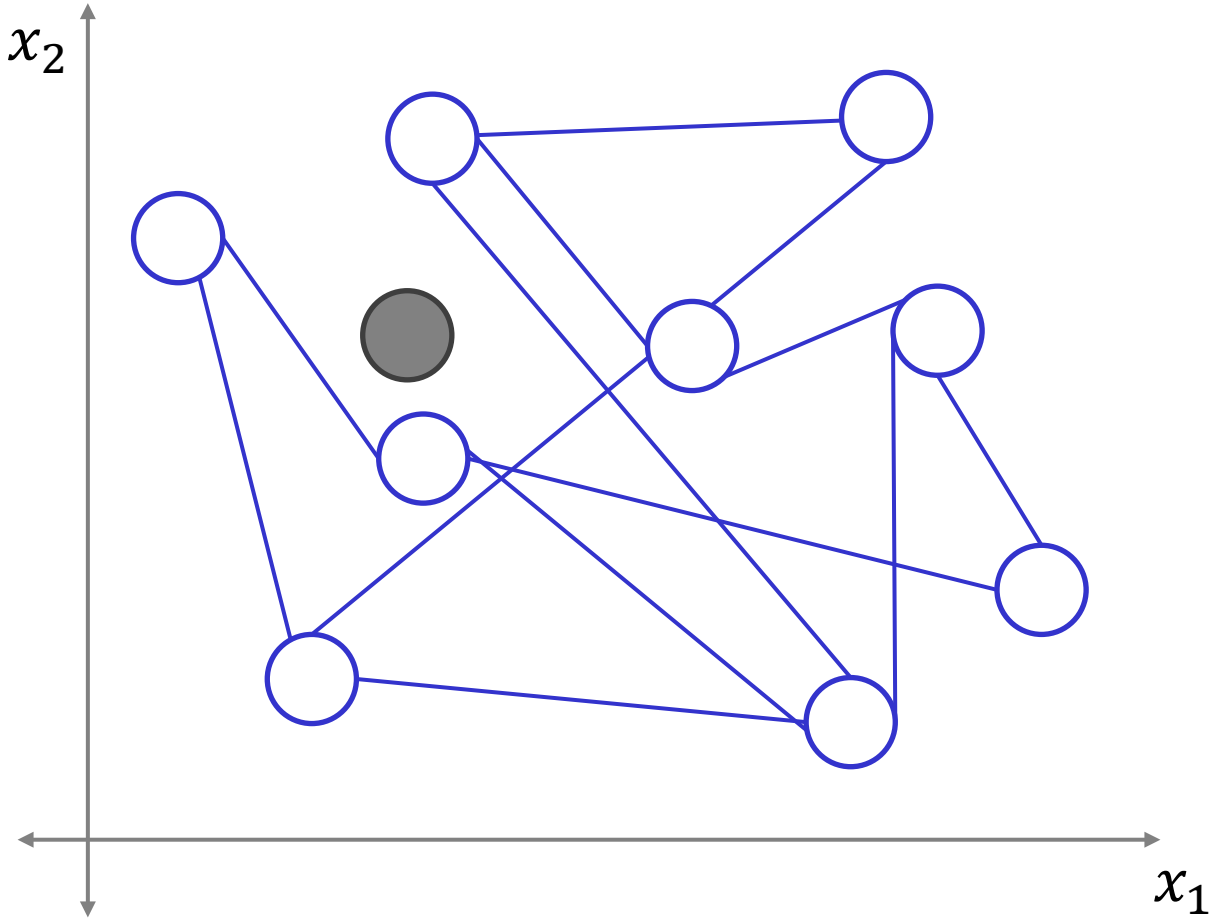


- $h_{j,i}$ or $d_{j,i}$ can also be computed in other ways

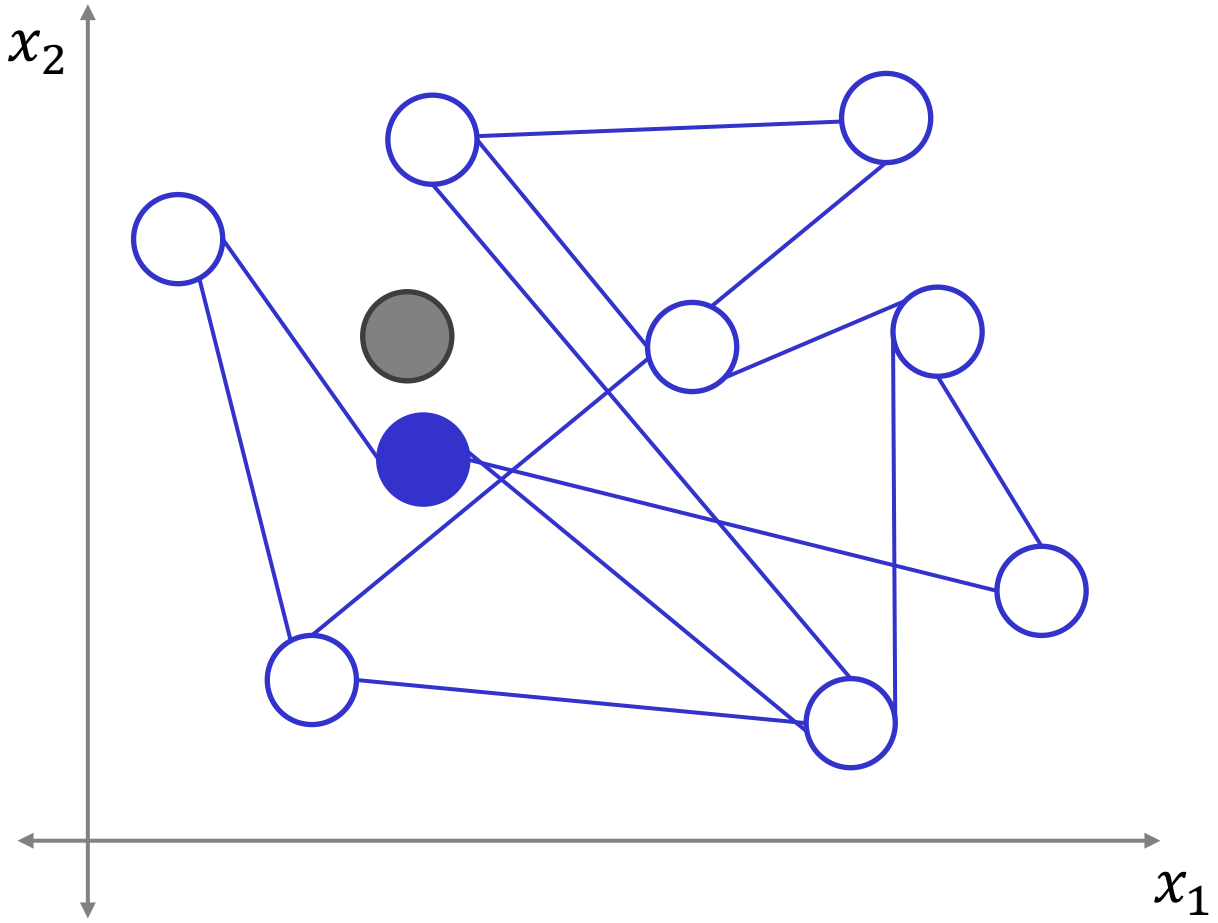
SOM training example:
Initial configuration of neuron prototypes



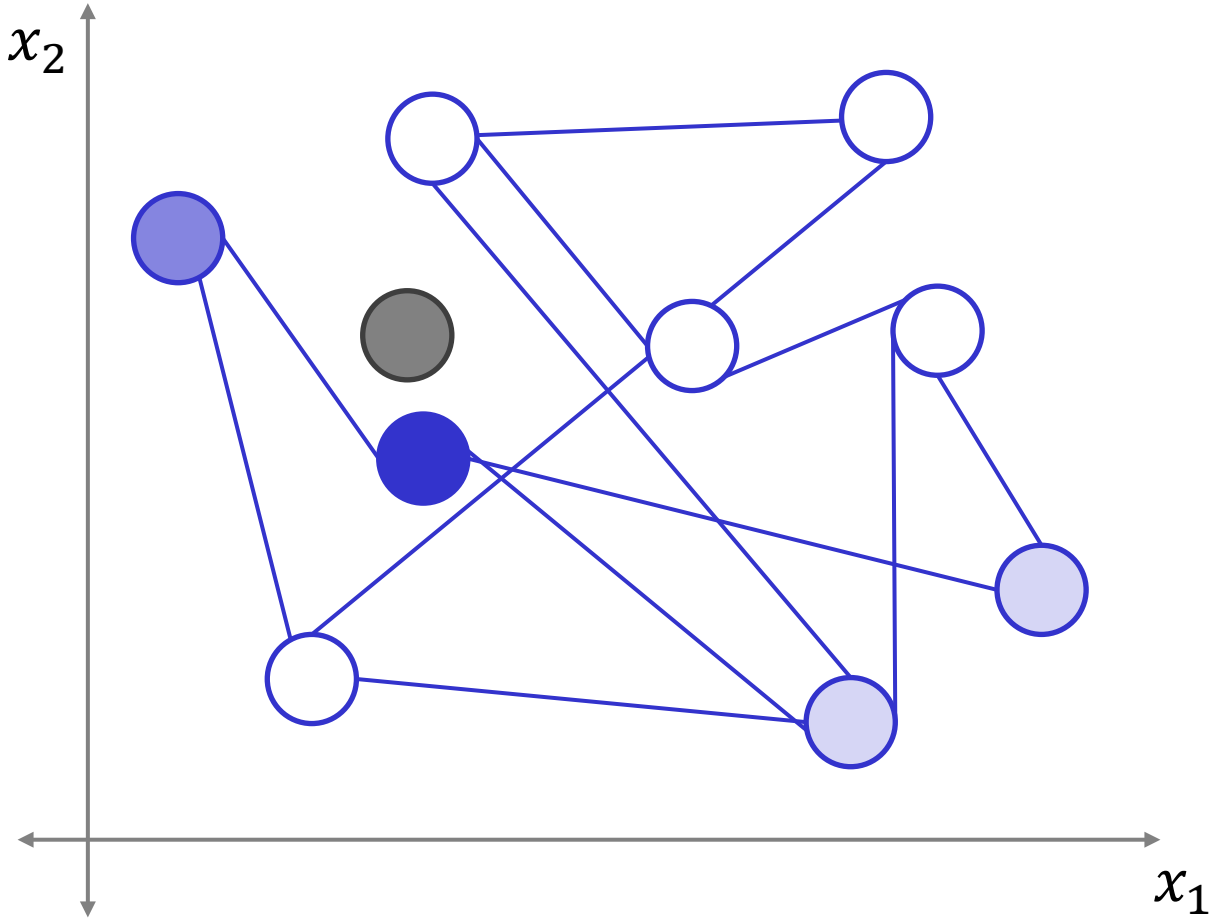
SOM training example: Observe point



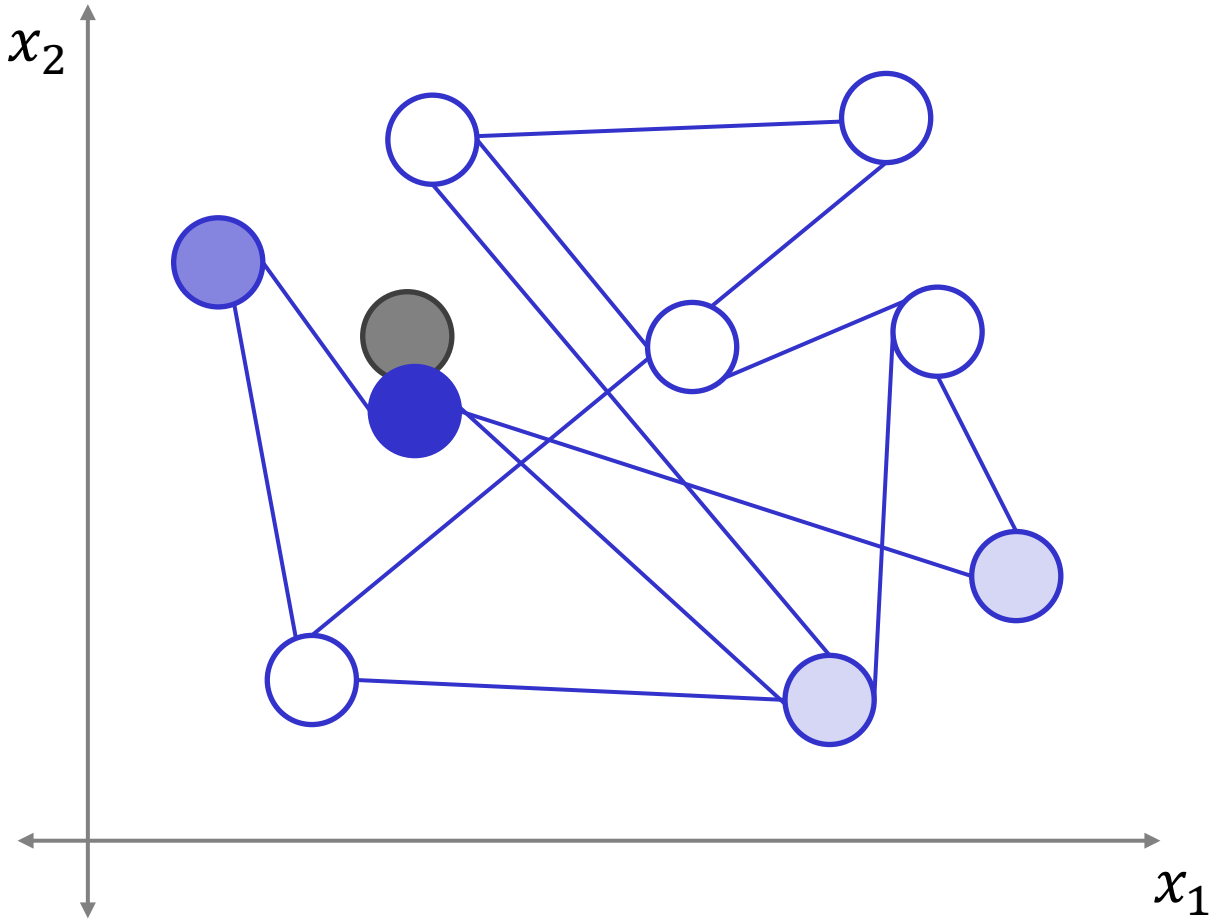
SOM training example:
Find closest neuron to observation



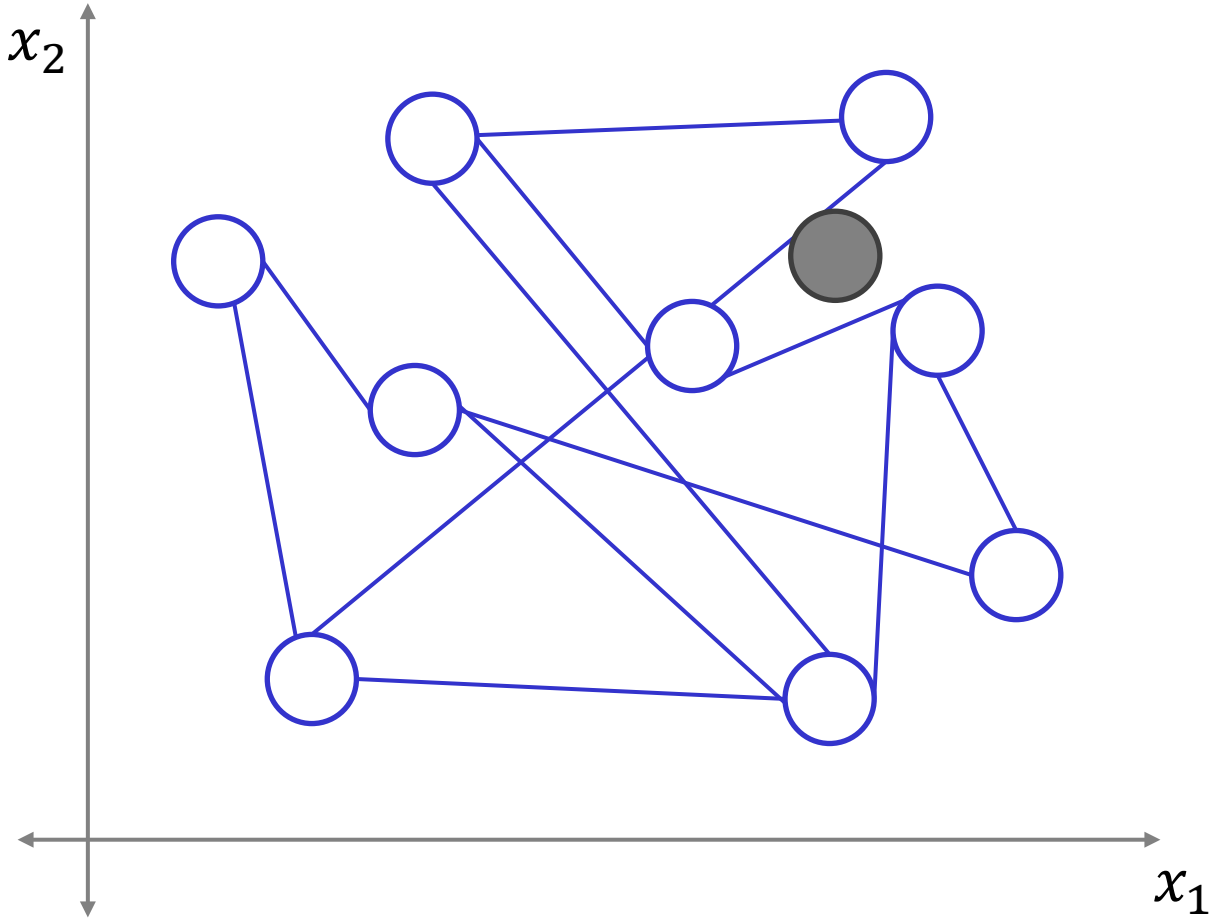
SOM training example:
Activate neurons close **in grid** to that neuron



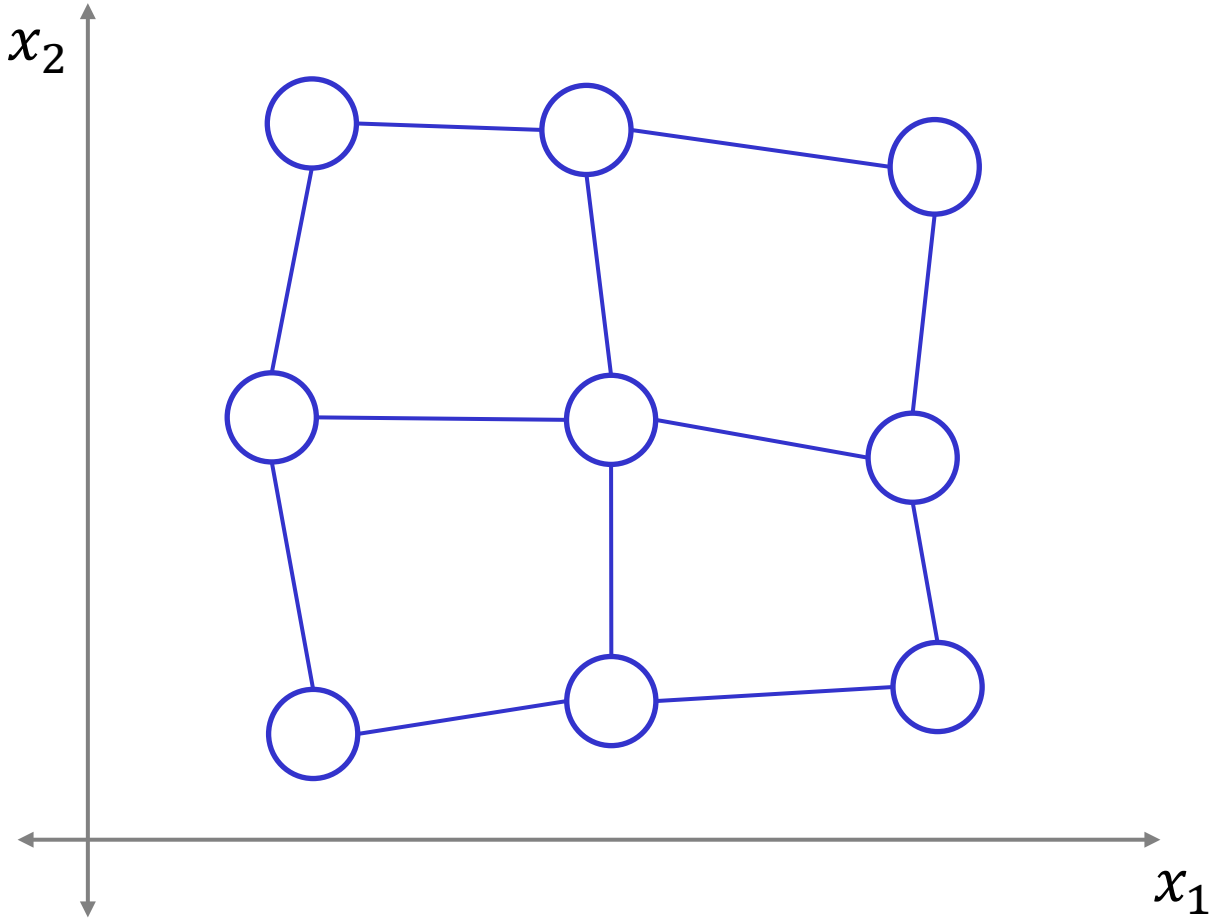
SOM training example:
Move selected neurons towards observation



SOM training example: Observe next point



SOM training example: After many iterations



SOMs use two phases of learning

- Initial ordering phase
  - Align output manifold to data manifold
  - Bring neural neighbors' representations together
  - η and σ are high
- Subsequent convergence phase
  - Fine-tune representation at each neuron
  - Little or no interaction across neurons
  - η and σ are low

Two phases of SOM training

- **Ordering phase**: This phase is to achieve topological ordering of weight vectors
- One approach is to set

$$\sigma(n) = \sigma_0 \cdot \left(1 - \frac{n}{N_0}\right) \quad \eta(n) = \eta_0 \cdot \left(1 - \frac{n}{N_0 + K}\right)$$

  - σ_0 is the initial (large) Gaussian width and N_0 is the number of iterations for the phase
  - η_0 is the initial learning rate and K is another parameter

Two phases of training (cont.)

- Alternatively, we can set $\sigma(n)$ and $\eta(n)$ as given in textbook

$$\sigma(n) = \sigma_0 \cdot \exp\left(-\frac{n}{\tau_1}\right)$$

$$\eta(n) = \eta_0 \cdot \exp\left(-\frac{n}{\tau_2}\right)$$

where τ_1 and τ_2 are called time constants

Two phases of training (cont.)

- **Convergence phase**. This phase fine-tunes the output neurons to match the input distribution
- For the convergence phase, $h_{j,i}(n)$ should contain just the nearest neighbors, which may reduce to one neuron. η should be small.

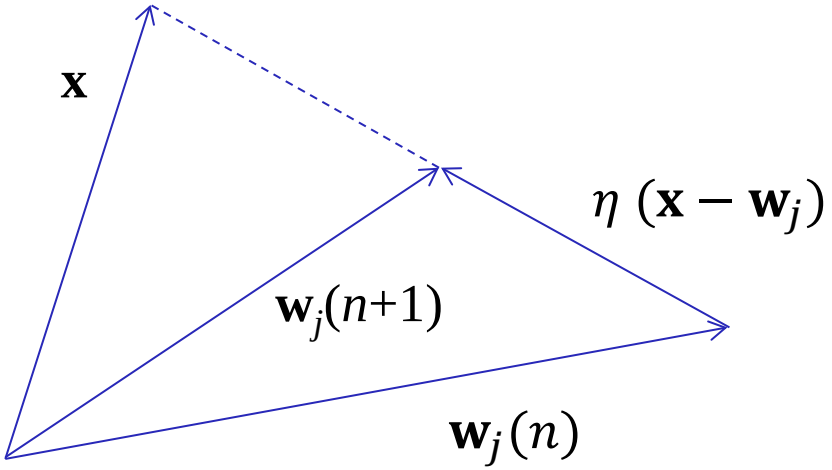
SOM with a trivial neighborhood reduces to competitive learning

- For the special case of a neighborhood function that includes just the winning neuron, SOM reduces to competitive learning:

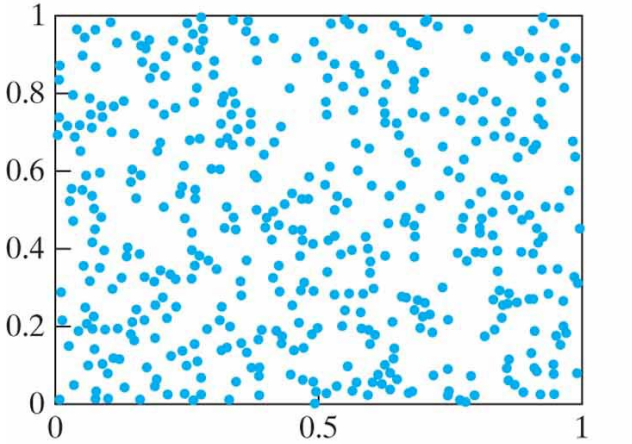
$$\Delta \mathbf{w}_j = \eta y_j (\mathbf{x} - \mathbf{w}_j)$$

Here y_j is the (binary) response of neuron j (1 if it is the winner, 0 if not)

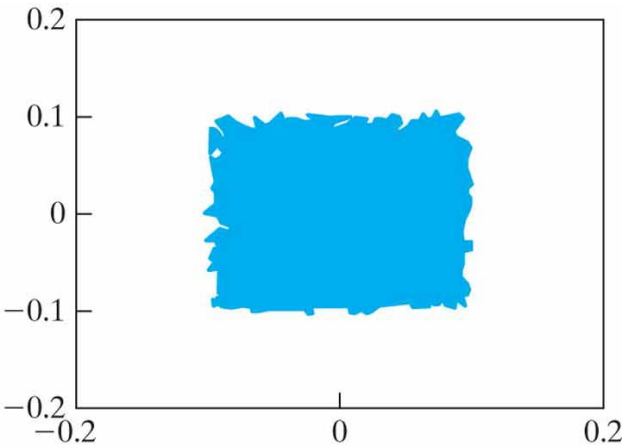
Competitive learning is online *K*-means



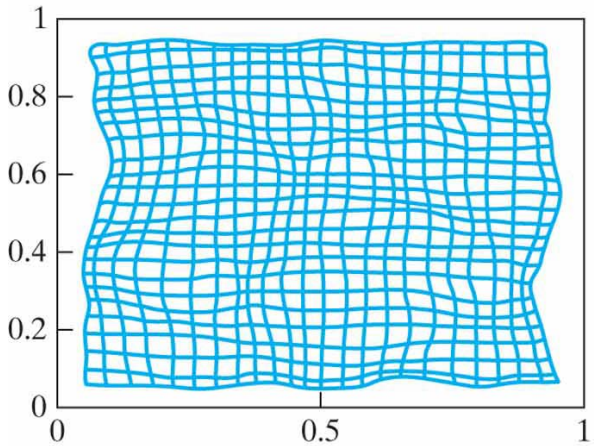
2D SOM on 2D data



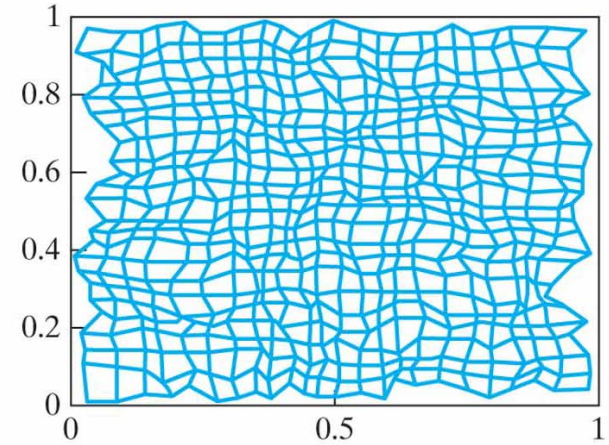
(a) Input distribution



Time = 0
(b) Initial weights

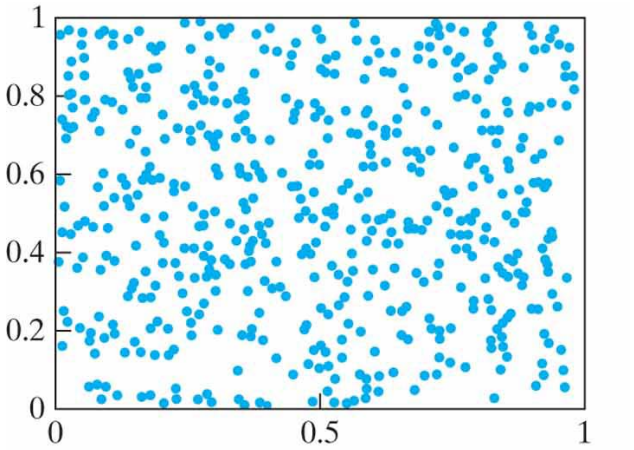


Time = 160 K
(c) Ordering phase

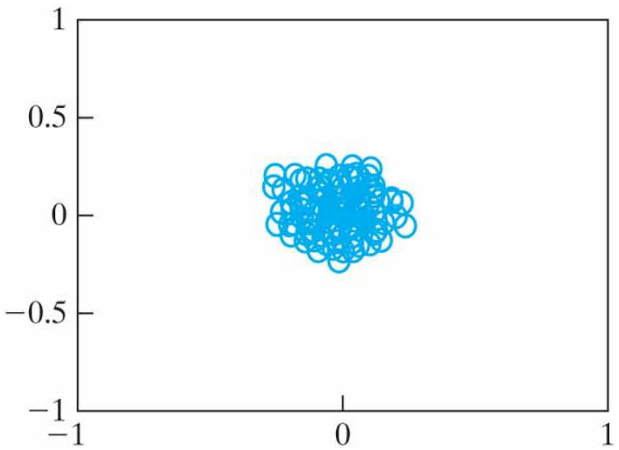


Time = 800 K
(d) Convergence phase

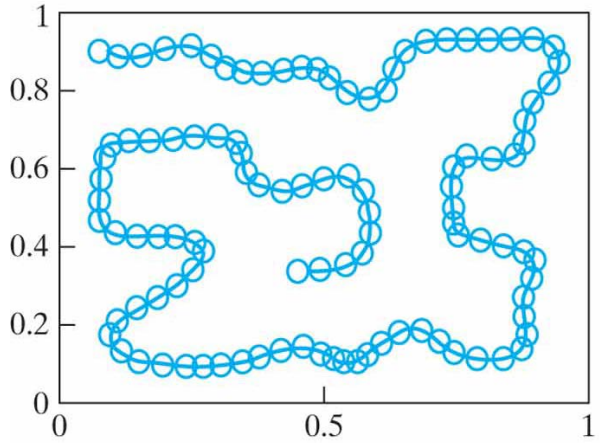
1D SOM on 1D data



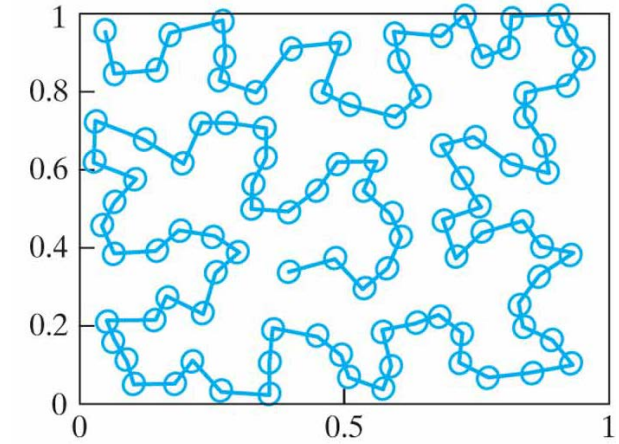
(a) Input distribution



Time = 0
(b) Initial weights

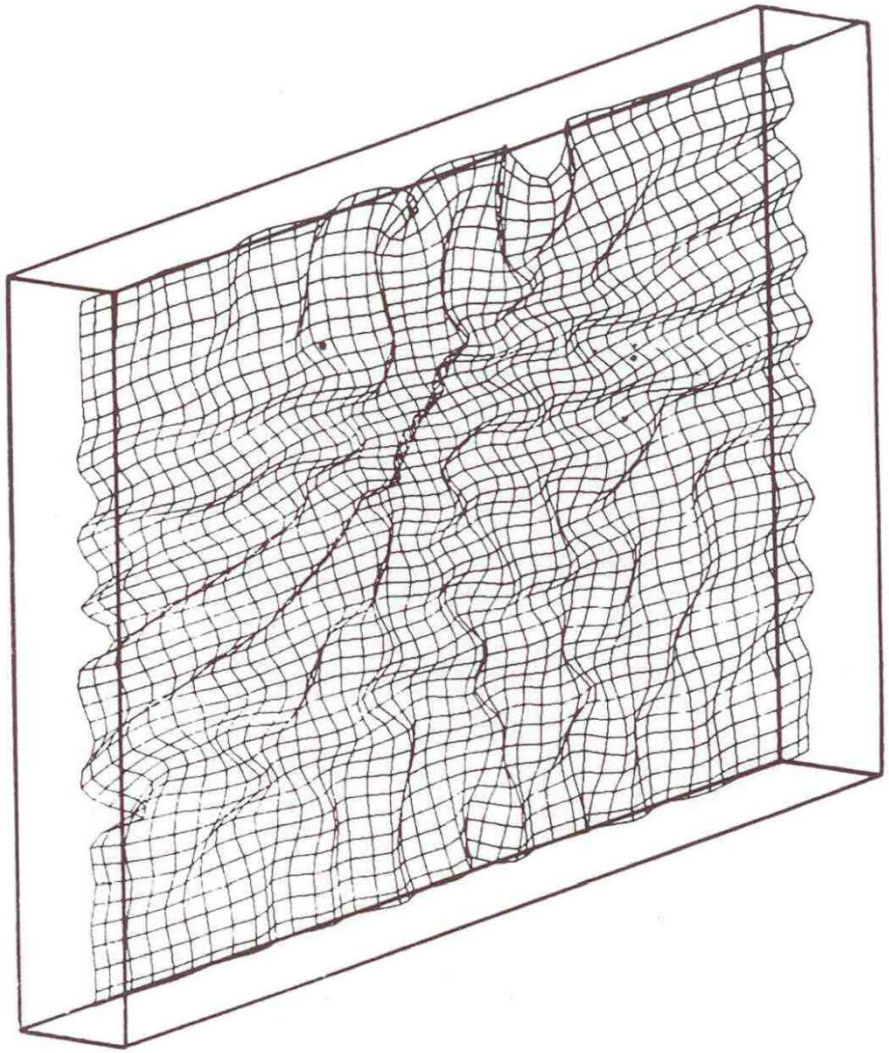


Time = 50 K
(c) Ordering phase



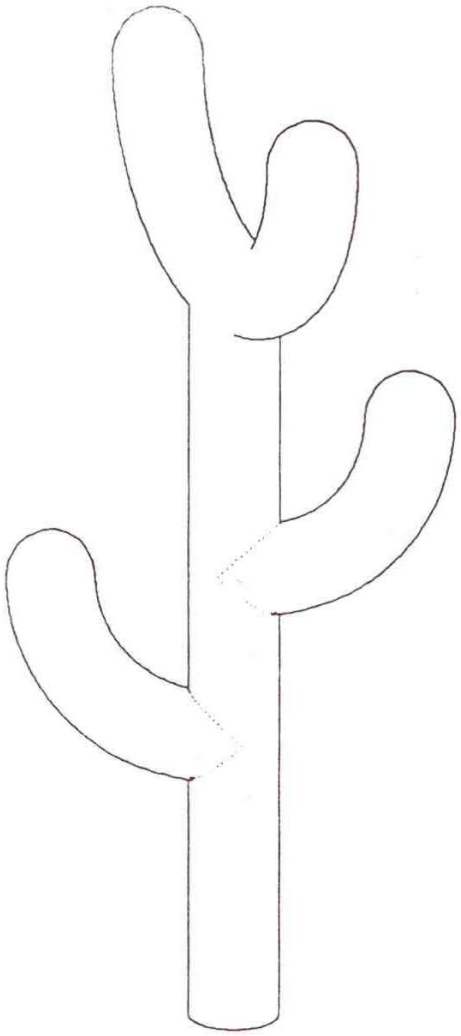
Time = 100 K
(d) Converging phase

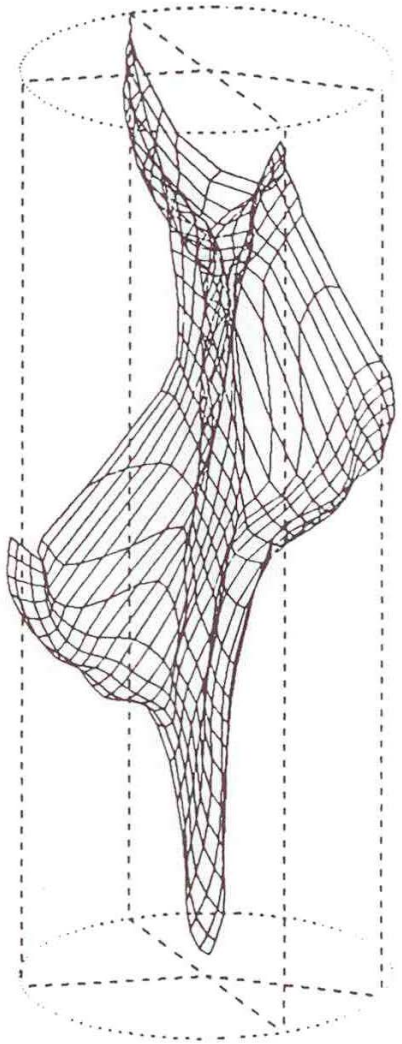
SOM illustrations (cont.)



Kohonen (1990)

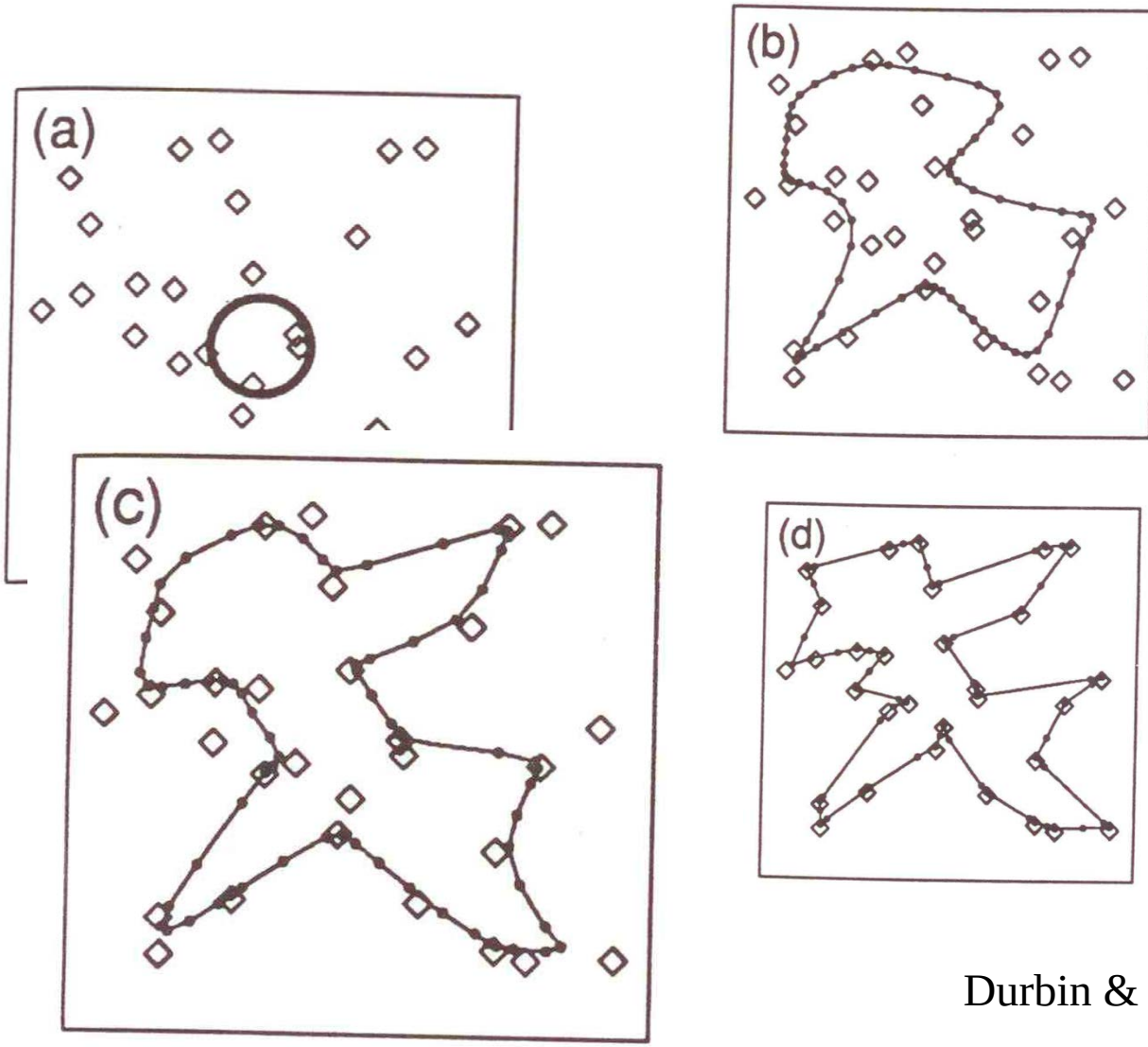
SOM illustrations (cont.)





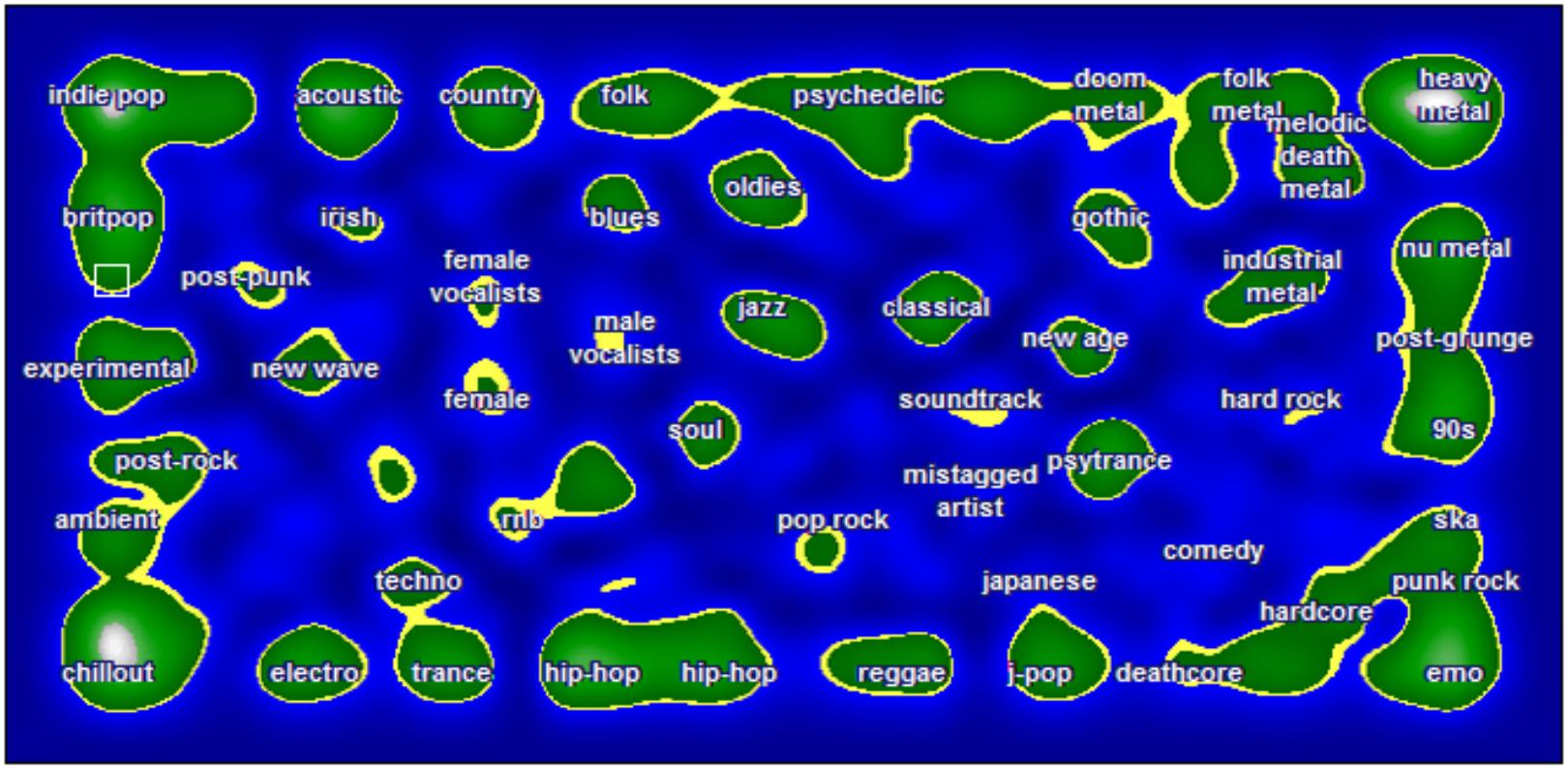
Kohonen (1990)

Elastic net for traveling salesman problem



Durbin & Willshaw (1987)

Example: Islands of Music



Pampalk, Rauber, Merkl (2002)