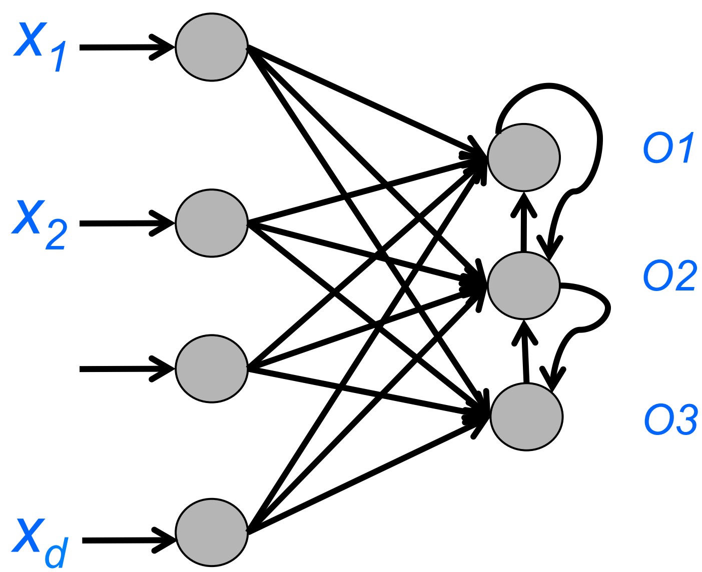
Competitive Learning

Lecture 10

Competitive Learning

- **A form of unsupervised training where output units are said to be in competition for input patterns**
  - During training, the output unit that provides the highest activation to a given input pattern is declared the weights of the winner and is moved closer to the input pattern, whereas the rest of the neurons are left unchanged
  - This strategy is also called winner-take-all since only the winning neuron is updated
  - Output units may have lateral inhibitory connections so that a winner neuron can inhibit others by an amount proportional to its activation level



Competitive Learning

- **With normalized vectors, the activation function of the i^th unit can be computed as the inner product of the unit's weight vector w_i and a particular input pattern $x^{(n}$**

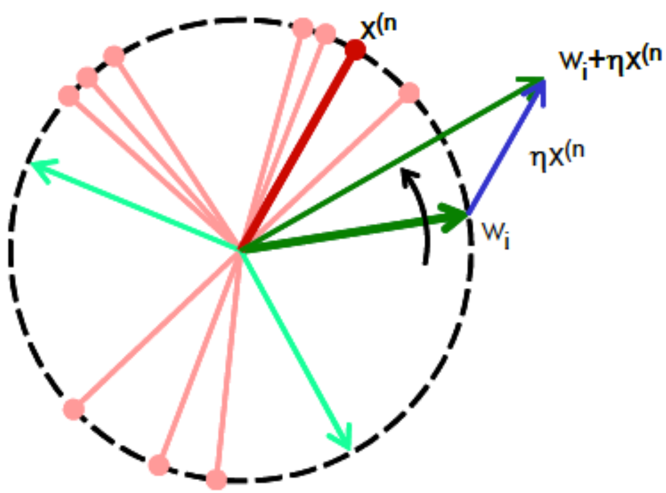
$$g_i(x^{(n}) = w_i^T x^{(n}$$

  - Note: the inner product of two normal vectors is the cosine of the angle between them

- **The neuron with largest activation is then adapted to be more like the input that caused the excitation**

$$w_i(t+1) = w_i(t) + \eta x^{(n}$$

  - Following update, the weight vector is renormalized (||w||=1)



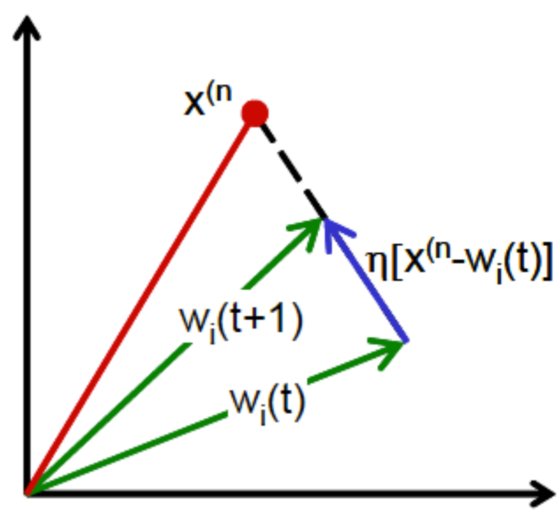
Competitive Learning

- If weights and input patters are un-normalized, the activation function becomes the Euclidean distance

$$g_i(x^{(n)}) = \sqrt{\sum_i \left(w_i - x_i^{(n)}\right)^2}$$

- The learning rule then become

$$w_i(t+1) = w_i(t) + \eta(x^{(n} - w_i(t))$$



Competitive Learning

■ Competitive Learning Algorithm

1. Normalize all input patterns
2. Randomly select a pattern $x^{(n}$
   2a. Find the winner neuron

   $$i = \arg\max_j \left[w_j^T x^{(n} \right]$$

   2.b. Update the winner neuron

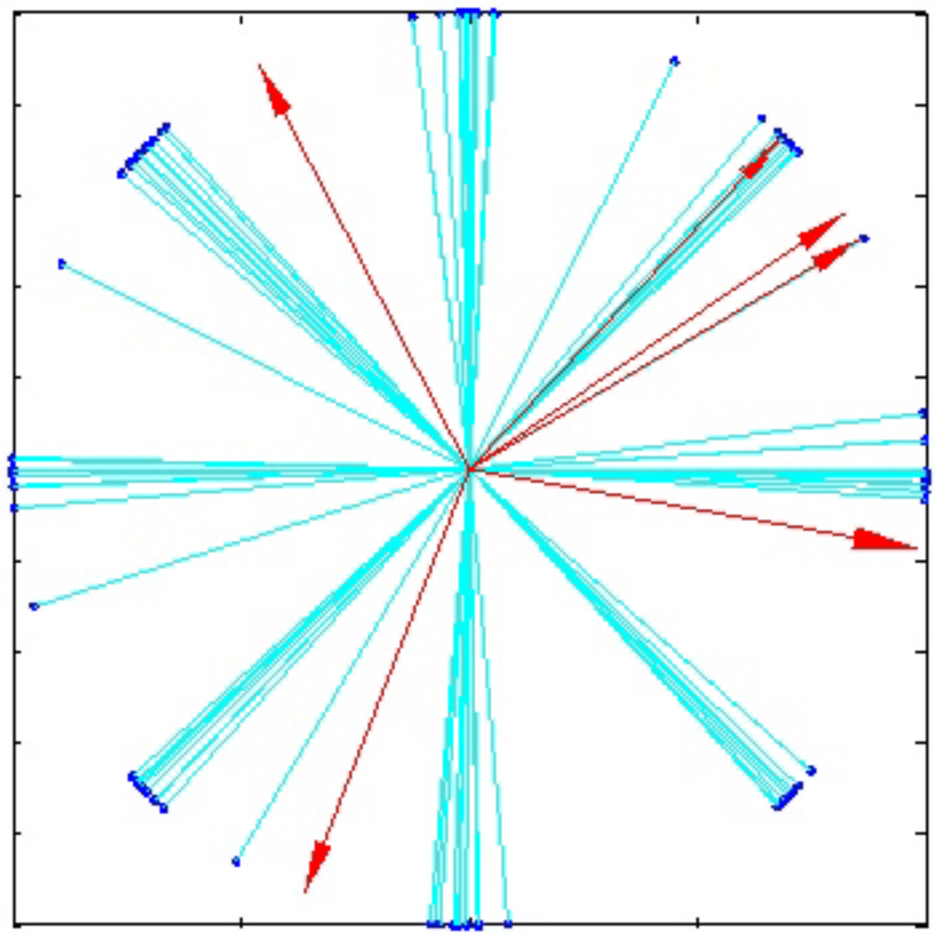
   $$w_i = w_i + \eta x^{(n}$$

   2c. Normalize the winner neuron

   $$w_i = \frac{w_i}{\|w_i\|}$$
3. Go to step 2 until no changes occur in N_{EX} runs

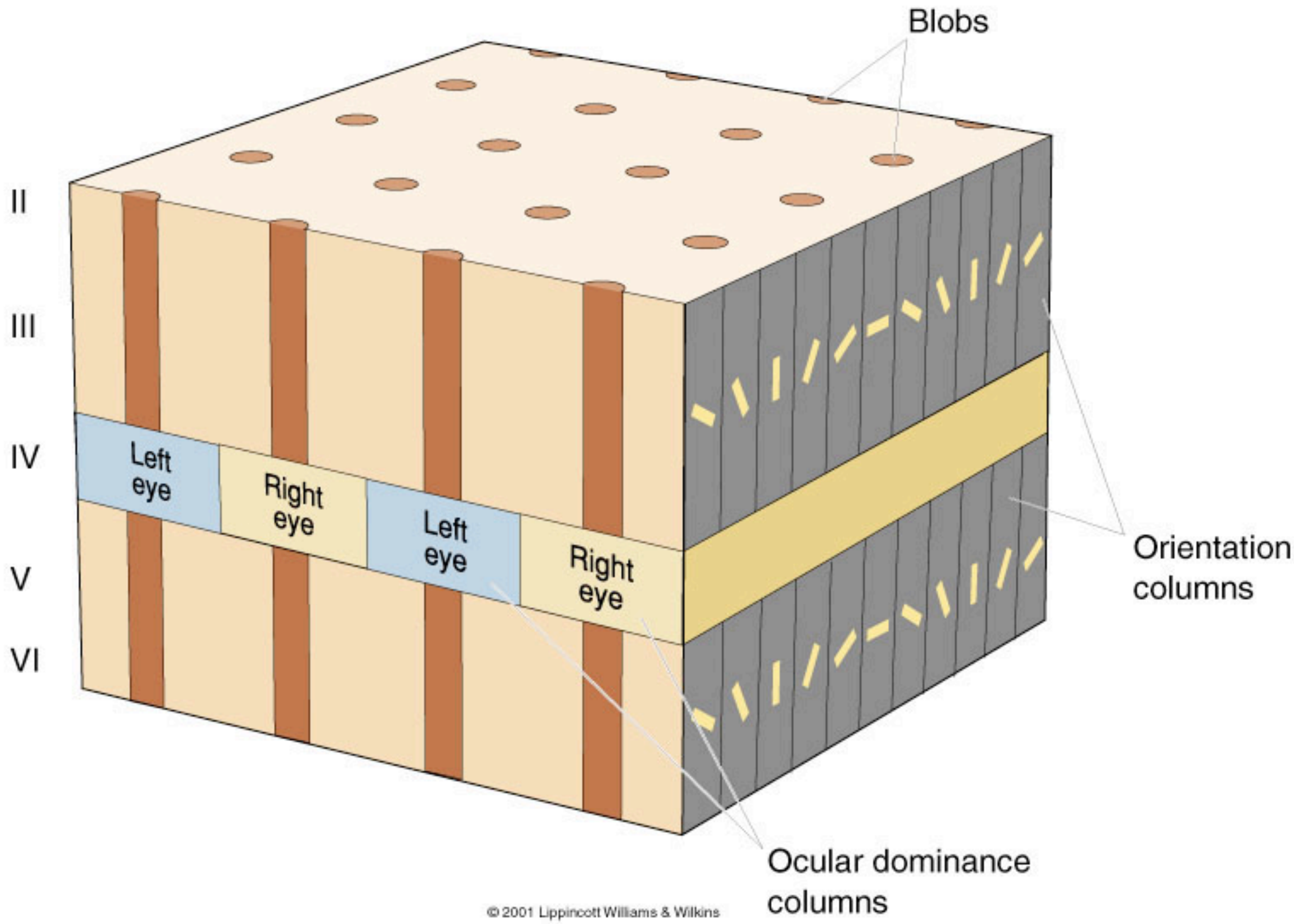
Competitive Learning

- **Demo:**



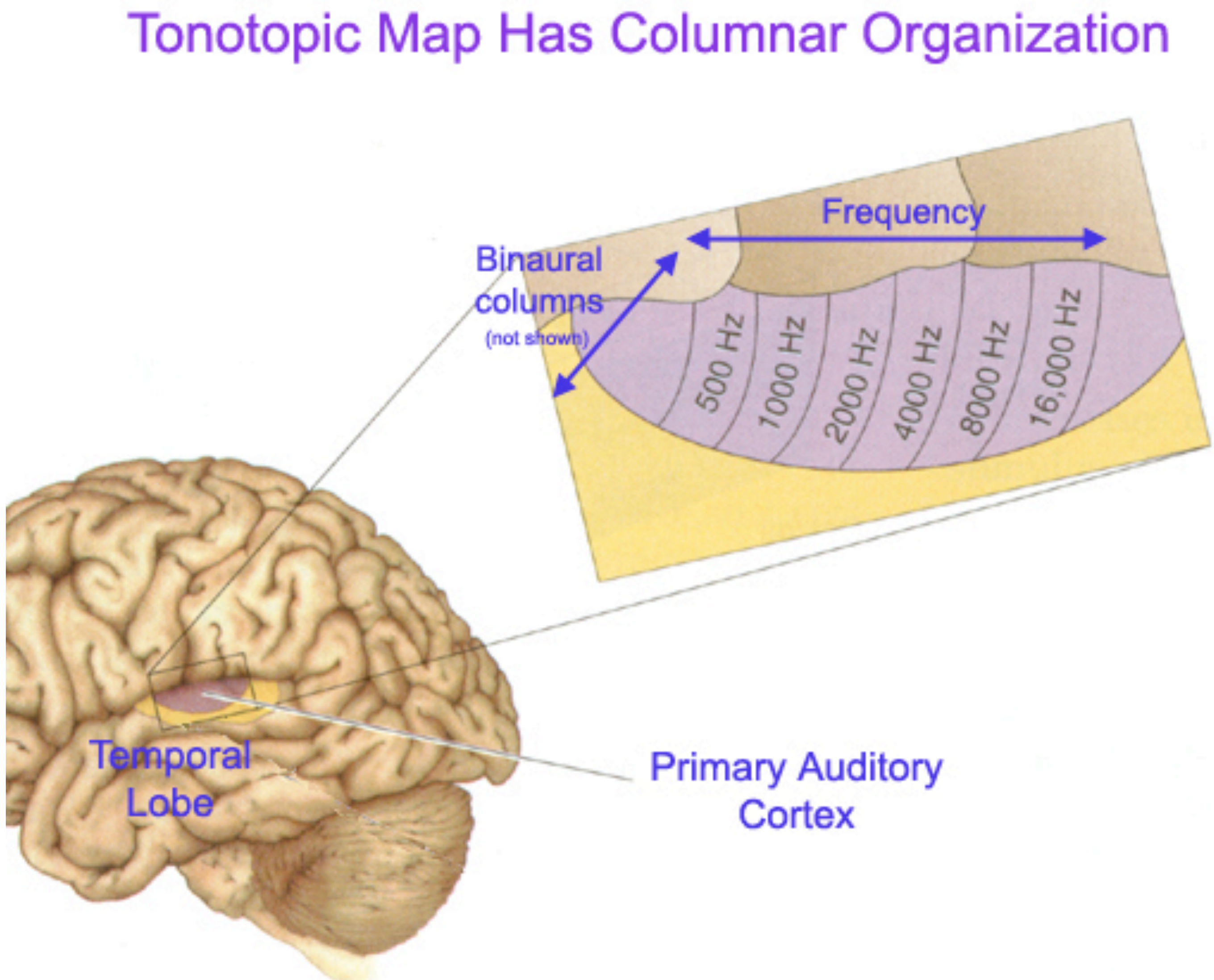
Direction maps

Figure 10.26
A cortical module.



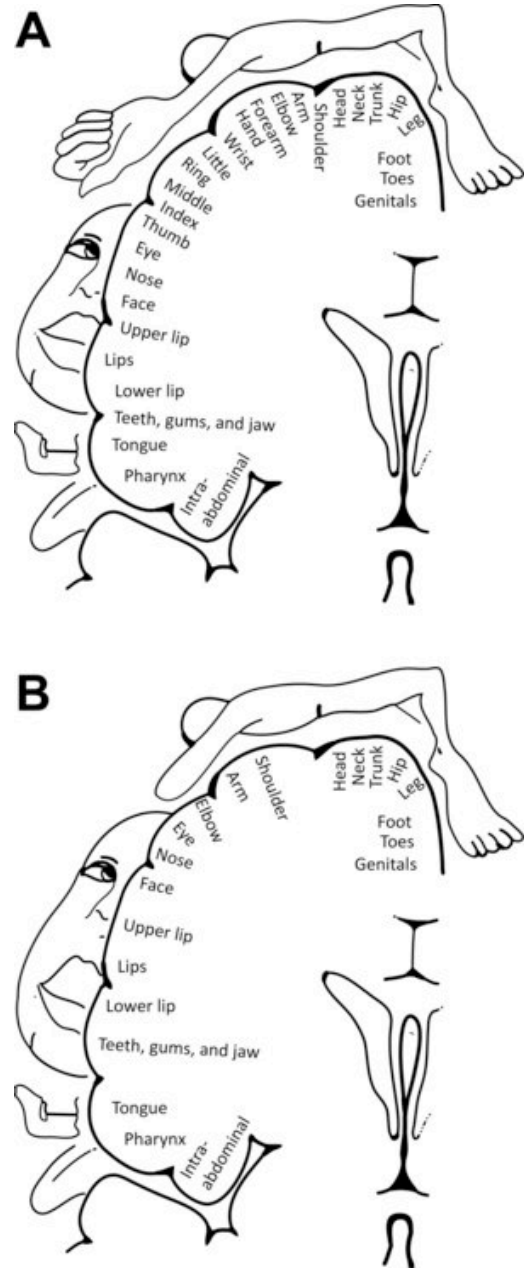
mcb.berkeley,edu/

Tonotopic maps



http://www.d.umn.edu/~jfitzake/Lectures/UndergradPharmacy/SensoryPhysiology/Audition/TonotopicMaps.html

Phantom Digits



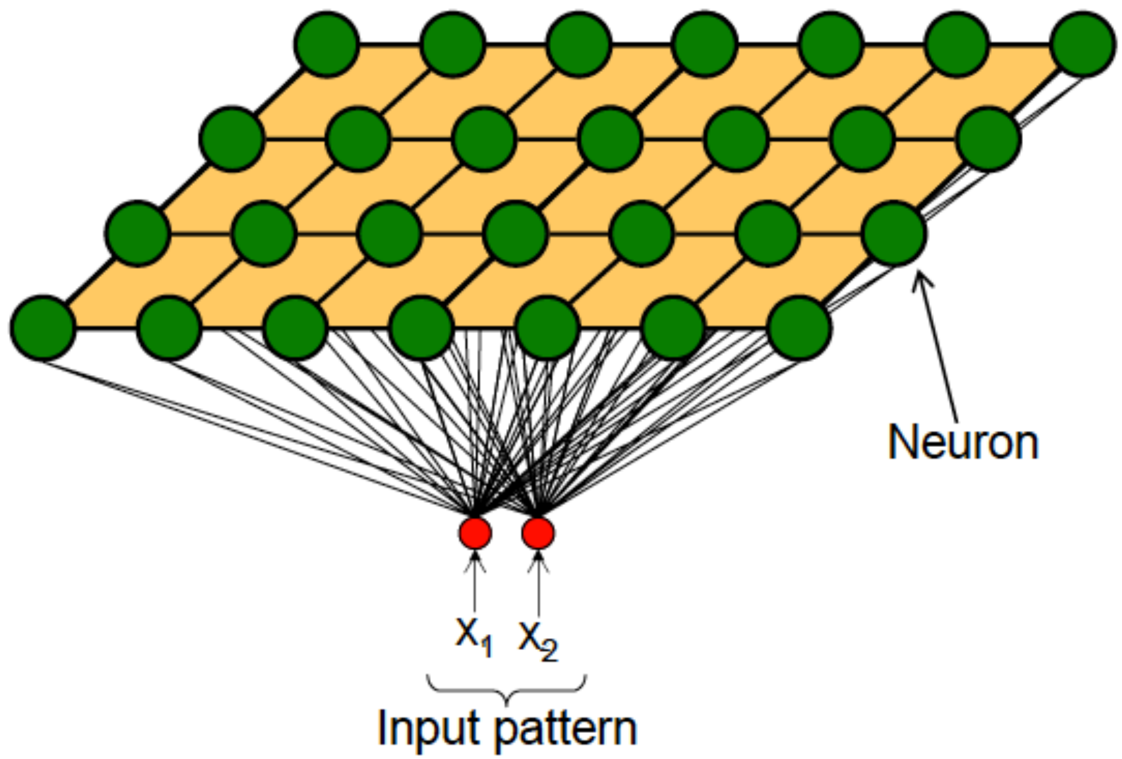
http://www.scholarpedia.org/article/Phantom_touch

Kohonen Self Organizing Maps

- **Kohonen Self-Organizing Maps (SOMs) produce a mapping from a multidimensional input space onto a lattice of clusters (or neurons)**
  - The key feature in SOMs is that the mapping is topology-preserving, in that neighboring neurons respond to “similar” input patterns
  - SOMs are typically organized as one- or two- dimensional lattices (i.e., a string or a mesh) for the purpose of visualization and dimensionality reduction
- **Unlike MLPs trained with the back-propagation algorithm, SOMs have a strong neurobiological basis**
  - On the mammalian brain, visual, auditory and tactile inputs are mapped into a number of “sheets” (folded planes) of cells [Gallant, 1993]
  - Topology is preserved in these sheets; for example, if we touch parts of the body that are close together, groups of cells will fire that are also close together
- **Kohonen SOMs result from the synergy of three basic processes**
  - Competition
  - Cooperation
  - Adaptation

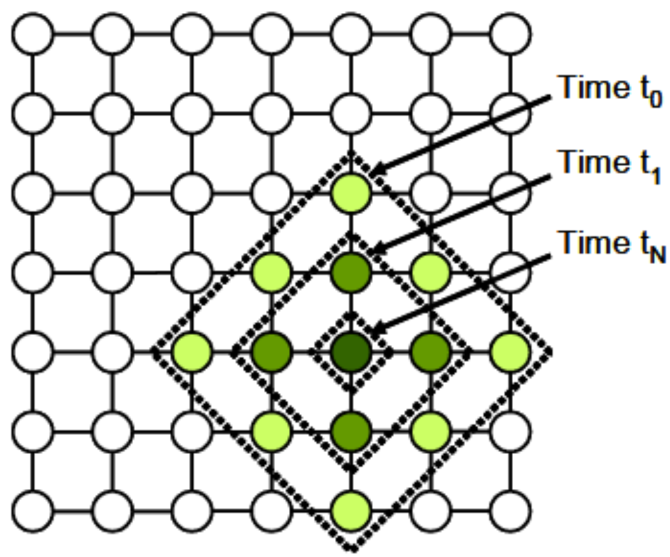
Competition

- **Each neuron in a SOM is assigned a weight vector with the same dimensionality d as the input space**
- **Any given input pattern is compared to the weight vector of each neuron and the closest neuron is declared the winner**
    - The Euclidean metric is commonly used to measure distance



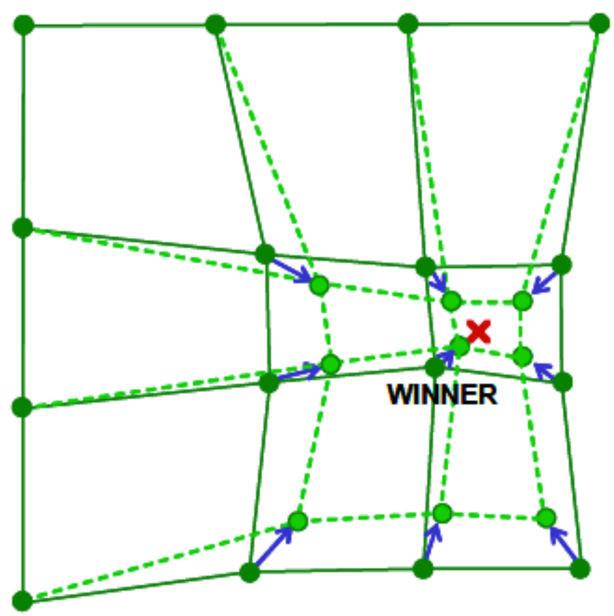
Cooperation

- **The activation of the winning neuron is spread to neurons in its immediate neighborhood**
  - This allows topologically close neurons to become sensitive to similar patterns
- **The winner's neighborhood is determined on the lattice topology**
  - Distance in the lattice is a function of the number of lateral connections to the winner (as in city-block distance)
- **The size of the neighborhood is initially large, but shrinks over time**
  - An initially large neighborhood promotes a topology-preserving mapping
  - Smaller neighborhoods allows neurons to specialize in the latter stages of training



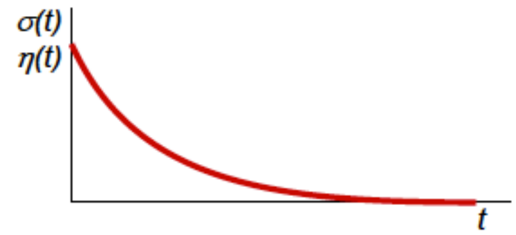
Adaptation

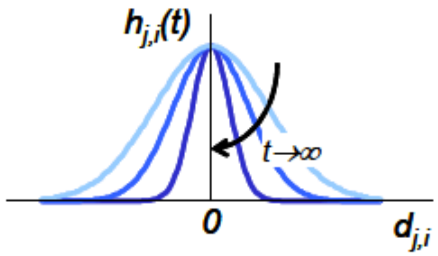
- **During training, the winner neuron and its topological neighbors are adapted to make their weight vectors more similar to the input pattern that caused the activation**
  - The adaptation rule is similar to the one presented in slide 4
  - Neurons that are closer to the winner will adapt more heavily than neurons that are further away
  - The magnitude of the adaptation is controlled with a learning rate, which decays over time to ensure convergence of the SOM



SOM Algorithm

- **Define**
  - A learning rate decay rule $\eta(t)=\eta_0 \exp\left(-\frac{t}{\tau_1}\right)$
  - A neighborhood kernel function $h_{ik}(t)=\exp\left(-\frac{d_{ik}^2}{2\sigma^2(t)}\right)$
    - where d_{ik} is the lattice distance between w_i and w_k
  - A neighborhood size decay rule $\sigma(t)=\sigma_0 \exp\left(-\frac{t}{\tau_2}\right)$





1. Initialize weights to some small, random values
2. Repeat until convergence
   - 2a. Select the next input pattern $x^{(n}$ from the database
     - 2a1. Find the unit w_i that best matches the input pattern $x^{(n}$

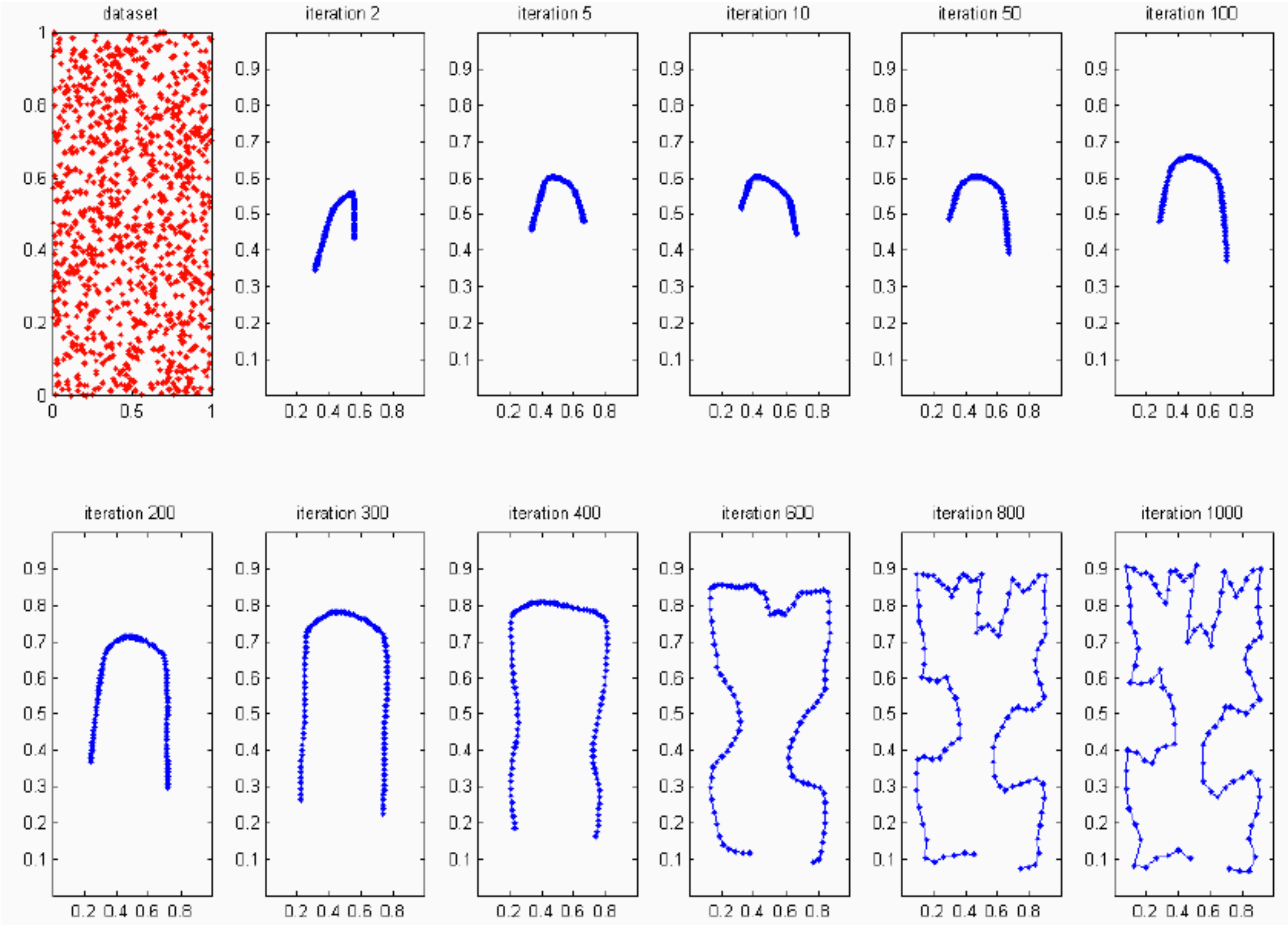
       $$i\left(x^{(n}\right)=\underset{j}{\arg\min}\left\|x^{(n}-w_j\right\|$$

     - 2a2. Update the weights of the winner w_i and all its neighbors w_k

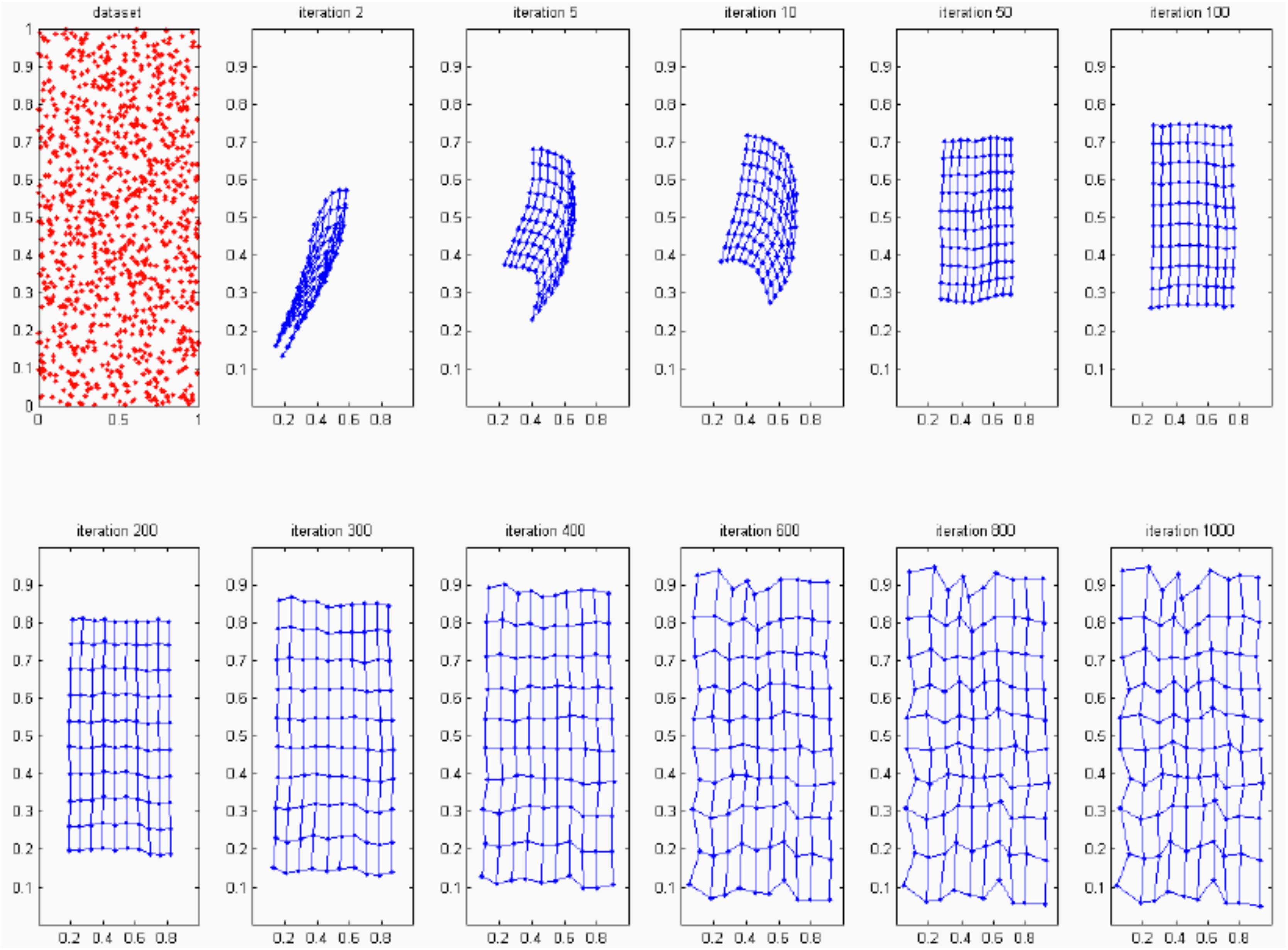
       $$w_k = w_k + \eta(t)\cdot h_{ik}(t)\cdot\left(x^{(n}-w_k\right)$$

   - 2b. Decrease the learning rate $\eta(t)$
   - 2c. Decrease neighborhood size $\sigma(t)$

SOM Example(1d)



SOM Example(2d)



SOM Demo