

## Slide09

# Haykin Chapter 9: Self-Organizing Maps

CPSC 636-600

Instructor: Yoonsuck Choe

Spring 2008

1

## SOM and the Cortical Maps

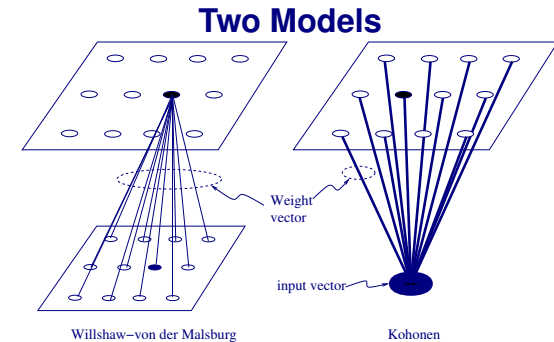
- The development of SOM as a neural model is motivated by the **topographical nature** of cortical maps.
- Visual, tactile, and acoustic inputs are mapped in a topographical manner.
  - **Visual**: retinotopy (position in visual field), orientation, spatial frequency, direction, ocular dominance, etc.
  - **Tactile**: somatotopy (position on skin)
  - **Acoustic**: tonotopy (frequency)

3

## Introduction

- Self-organizing maps (SOM) is based on **competitive learning**, where output neurons compete with each other to be activated (Kohonen, 1982).
- The output neuron that activates is called the **winner-takes-all neuron**.
- **Lateral inhibition** is one way to implement competition for map formation (von der Malsburg 1973).
- In SOM, neurons are placed on a **lattice**, on which a meaningful coordinate system for different **features** is created (**feature map**).
- The lattice thus forms a **topographic map** where the spatial location on the lattice is indicative of the input features.

2



- **Willshaw-von der Malsburg model**: input neurons arranged in 2D lattice, output in 2D lattice. Lateral excitation/inhibition (Mexican hat) gives rise to soft competition. Normalized Hebbian learning. Biological motivation.
- **Kohonen model**: input of any dimension, output neurons in 1D, 2D, or 3D lattice. Relaxed winner-takes-all (neighborhood). Competitive learning rule. Computational motivation.

4

## SOM Overview

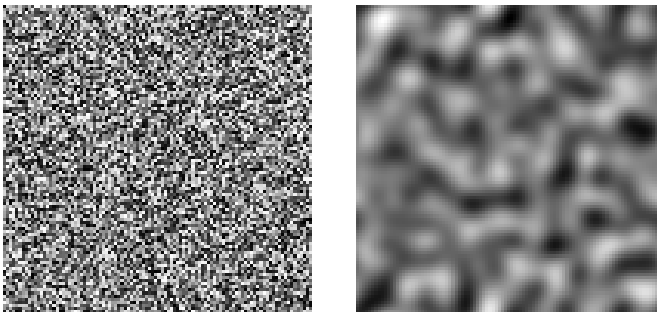
SOM is based on three principles:

- **Competition:** each neuron calculates a discriminant function. The neuron with the highest value is declared the winner.
- **Cooperation:** Neurons near-by the winner on the lattice get a chance to adapt.
- **Adaptation:** The winner and its neighbors increase their discriminant function value relative to the current input. Subsequent presentation of the current input should result in enhanced function value.

**Redundancy** in the input is needed!

5

### Redundancy, etc. (cont'd)



	Left	Right
Structure	No	Yes
Redundancy	No	Yes
Info < Capacity	No	Yes

Consider each pixel as one random variable.

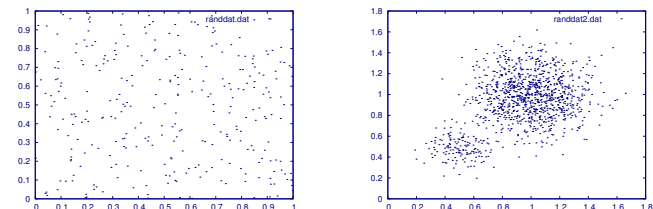
7

## Redundancy, etc.

- Unsupervised learning such as SOM require redundancy in the data.
- The following are intimately related:
  - Redundancy
  - Structure (or organization)
  - Information content relative to channel capacity

6

### Redundancy, etc. (cont'd)

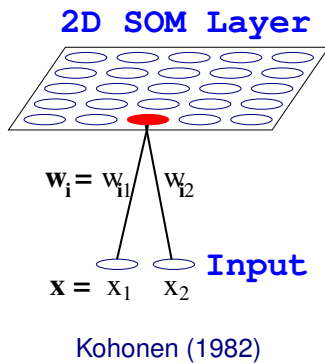


	Left	Right
Structure	No	Yes
Redundancy	No	Yes
Info < Capacity	No	Yes

Consider each axis as one random variable.

8

## Self-Organizing Map (SOM)



- 1-D, 2-D, or 3-D layout of units.
- One weight vector for each unit.
- Unsupervised learning (no target output).

9

## SOM Algorithm

1. Randomly initialize weight vectors  $\mathbf{w}_i$
2. Randomly sample input vector  $\mathbf{x}$
3. Find Best Matching Unit (BMU):

$$\mathbf{i}(\mathbf{x}) = \operatorname{argmin}_j \|\mathbf{x} - \mathbf{w}_j\|$$

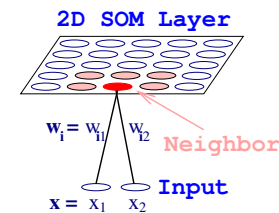
4. Update weight vectors:

$$\mathbf{w}_j \leftarrow \mathbf{w}_j + \eta h(j, \mathbf{i}(\mathbf{x})) (\mathbf{x} - \mathbf{w}_j)$$

$\eta$  : learning rate

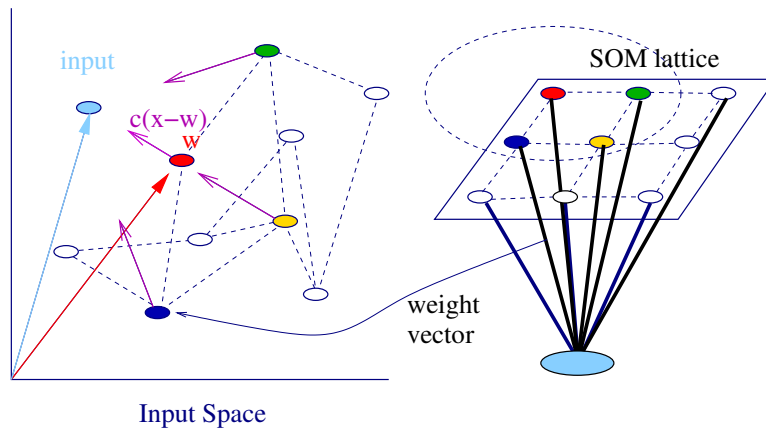
$h(j, \mathbf{i}(\mathbf{x}))$  : neighborhood function of BMU.

5. Repeat steps 2 – 4.



10

## SOM Learning



- Weight vectors can be plotted in the input space.
- Weight vectors move, not according to their proximity to the input in the input space, but according to their proximity in the lattice.

11

## Is This Hebbian Learning?: Sort of

- SOM learning can be viewed as Hebbian learning with a **forgetting term** to check unbounded growth.
- Original Hebb's rule:

$$\Delta \mathbf{w}_j = \eta y_j \mathbf{x},$$

where  $\mathbf{w}_j$  is the weight vector,  $\eta$  the learning rate,  $y_j$  the output response, and  $\mathbf{x}$  the input vector.

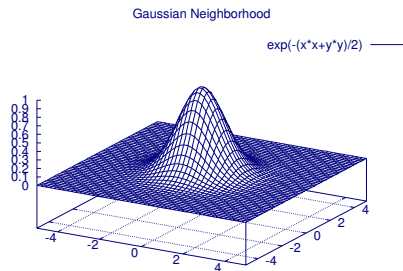
- Hebb's rule plus a **forgetting term**:

$$\begin{aligned} \Delta \mathbf{w}_j &= \eta y_j \mathbf{x} - g(y_j) \mathbf{w}_j \\ &= \eta y_j \mathbf{x} - \eta y_j \mathbf{w}_j \\ &= \eta h_{j, \mathbf{i}(\mathbf{x})} (\mathbf{x} - \mathbf{w}_j), \end{aligned}$$

assuming  $g(y) = \eta y$  and  $y_j = h_{j, \mathbf{i}(\mathbf{x})}$ .

12

## Typical Neighborhood Functions



- Gaussian:  $h(j, i(x)) = \exp(-\|\mathbf{r}_j - \mathbf{r}_{i(x)}\|^2 / 2\sigma^2)$
- Flat:  $h(j, i(x)) = 1$  if  $\|\mathbf{r}_j - \mathbf{r}_{i(x)}\| \leq \sigma$ , and 0 otherwise.
- $\sigma$  is called the **neighborhood radius**.
- $\mathbf{r}_j$  is the location of unit  $j$  on the lattice.

13

## Two Phases of Adaptation

- **Self-organization or ordering phase:** High learning rate, large neighborhood radius (entire map).
- **Convergence phase:** Low learning rate, small neighborhood radius (one or zero).

15

## Training Tips

- Start with large neighborhood radius.  
Gradually decrease radius to a small value.

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

- Start with high learning rate  $\eta$ .  
Gradually decrease  $\eta$  to a small value.

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

14

## Performance Measures

- **Quantization Error**  
Average distance between each data vector and its BMU.

$$\epsilon_Q = \frac{1}{N} \sum_{j=1}^N \|\mathbf{x}_j - \mathbf{w}_{i(x_j)}\|$$

- **Topographic Error**  
The proportion of all data vectors for which first and second BMUs are not adjacent units.

$$\epsilon_T = \frac{1}{N} \sum_{j=1}^N u(\mathbf{x}_j),$$

$u(\mathbf{x}) = 1$  if the 1st and 2nd BMUs are not adjacent  
 $u(\mathbf{x}) = 0$  otherwise.

16

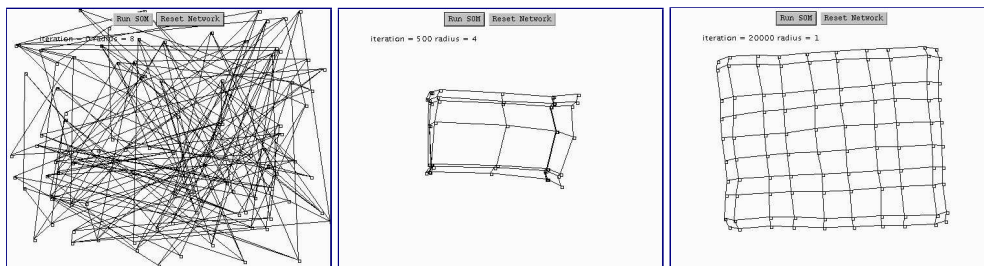
## SOM Summary

Essential ingredients of SOM: Hebbian learning rule (with forgetting term)

- Input generated according to a certain probability distribution on a continuous input space.
- Topology of network form on the discrete lattice.
- Time-varying neighborhood function around the winner.
- Time-varying learning rate.

17

### Example: 2D Input / 2D Output



- Train with uniformly random 2D inputs. Each input is a point in Cartesian plane.
- Nodes: weight vectors ( $x$  and  $y$  coordinate).
- Edges: connect immediate neighbors on the map.

19

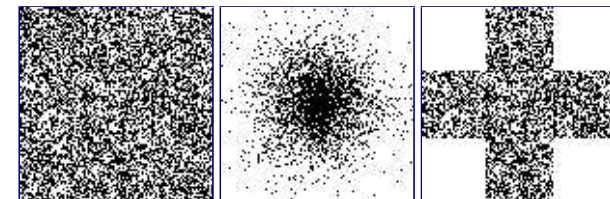
## SOM Summary (cont'd)

Properties of SOM

- **Approximation of the input space:** The collection of weight vectors provides a good approximation of the input space.
- **Topological ordering:** Spatial location on the lattice correspond to a certain feature of input patterns. Near-by neurons on the lattice represent similar input features.
- **Density matching:** More neurons are recruited to represent dense area in the input space.
- **Feature selection:** Select best features to approximate the underlying distribution.

18

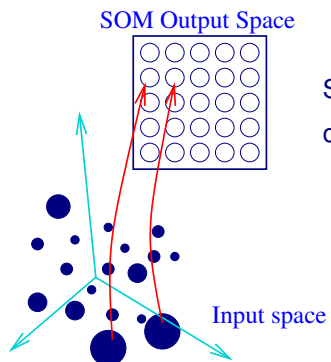
### Different 2D Input Distributions



- What would the resulting SOM map look like?
- Why would it look like that?

20

## High-Dimensional Inputs



SOM can be trained with inputs of arbitrary dimension.

- Dimensionality reduction: **N-D to 2-D.**
- Extracts topological features.
- Used for visualization of data.

21

## Exercise

1. What happens when  $h_{j,i}(x)$  and  $\eta$  was reduced quickly vs. slowly?
2. How would the map organize if different input distributions are given?
3. For a fixed number of input vectors from real-world data, a different visualization scheme is required. How would you use the number of input vectors that best match each unit to visualize the property of the map?

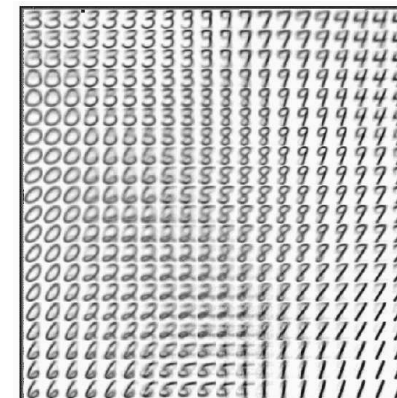
23

## Applications

- **Data clustering and visualization.**
- **Optimization problems:**  
Traveling salesman problem.
- **Semantic maps:**  
Natural language processing.
- **Preprocessing for signal and image-processing.**
  1. Hand-written character recognition.
  2. Phonetic map for speech recognition.

22

## SOM Example: Handwritten Digit Recognition



- Preprocessing for feedforward networks (supervised learning).
- Better representation for training.
- Better generalization.

24

## SOM Demo

Jochen Fröhlich's *Neural Networks with JAVA* page:

<http://fbim.fh-regensburg.de/~saj39122/jfroehl/diplom/e-index.html>

Check out the `Sample Applet` link.

25

## SOM Demo: Traveling Salesman Problem

Using Fröhlich's SOM applet:

- 1D SOM map ( $1 \times n$ , where  $n$  is the number of nodes).
- 2D input space.
- Initial neighborhood radius of 8.
- Stop when radius  $< 0.001$ .
- Try 50 nodes, 20 input points.

Click on [Parameters] to bring up the config panel. After the parameters are set, click on [Reset] in the main applet, and then [Start learning].

26

## SOM Demo: Space Filling in 2D

Using Fröhlich's SOM applet:

- 1D SOM map ( $1 \times n$ , where  $n$  is the number of nodes).
- 2D input space.
- Initial neighborhood radius of 100.
- Stop when radius  $< 0.001$ .
- Try 1000 nodes, and 1000 input points.

27

## SOM Demo: Space Filling in 3D

Using Fröhlich's SOM applet:

- 2D SOM map ( $n \times n$ , where  $n$  is the number of nodes).
- 2D input space.
- Initial neighborhood radius of 10.
- Stop when radius  $< 0.001$ .
- Try  $30 \times 30$  nodes, and 500 input points. Limit the  $y$  range to 15.

Also try  $50 \times 50$ , 1000 input points, and 16 initial radius.

28

## Vector Quantization

- **Vector quantization** exploits the structure in the input distribution for the purpose of data compression.
- In vector quantization, the input space is partitioned into a number of distinct regions and for each region a **reconstruction vector** is defined.
- A new input is then represented by the reconstruction vector representing the region it falls into.
- Since only the **index** of the reconstruction vector need to be stored or transmitted, significant saving is possible in terms of storage space and bandwidth.
- The collection of reconstruction vectors is called the **code book**.

29

## Learning Vector Quantization

- Train with SOM in unsupervised mode.
- Then, tune the weight vectors in a supervised mode:
  - If class of the input vector and the class of the best matching weight vector **match**,

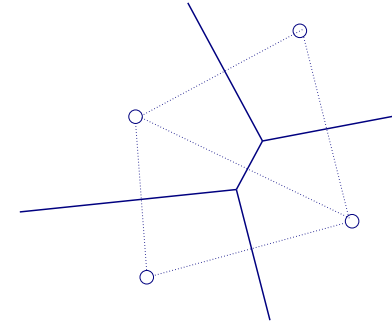
$$\mathbf{w}_c(n+1) = \mathbf{w}_c(n) + \alpha_n [\mathbf{x}_i - \mathbf{w}_c(n)]$$

- If class of the input vector and the class of the best matching weight vector **do not match**,

$$\mathbf{w}_c(n+1) = \mathbf{w}_c(n) - \alpha_n [\mathbf{x}_i - \mathbf{w}_c(n)]$$

31

## Vector Quantization (cont'd)



- A vector quantizer that minimizes encoding distortion is called a **Voronoi** or **nearest-neighbor quantizer**.
- SOM provides an approximate method for calculating the Voronoi tessellation.

30

## Other Topics

- Different ways of visualization using SOM.
- Contextual map (or semantics map).
- SOM viewed as
  - Abstract neuroscientific model of the cortex
  - Vector quantizer
- Difficulty of analysis (convergence, etc.)
- Use in modeling cortical map formation.

32