

Self-Organizing Maps (SOMs)

- Resources
 - Mehotra, K., Mohan, C. K., & Ranka, S. (1997). Elements of Artificial Neural Networks. MIT Press
 - pp. 187-202
 - Fausett, L. (1994). Fundamentals of Neural Networks. Prentice Hall. pp. 169-187
 - Bibliography of SOM papers
 - <http://citeseer.ist.psu.edu/104693.html>
 - <http://www.cis.hut.fi/research/som-bibl/>
 - Java applet & tutorial information
 - <http://davis.wpi.edu/~matt/courses/soms/>
 - WEBSOM - Self-Organizing Maps for Internet Exploration
 - <http://websom.hut.fi/websom/>

2

Supervised vs. Unsupervised Learning

- An important aspect of an ANN model is whether it needs guidance in learning or not. Based on the way they learn, all artificial neural networks can be divided into two learning categories - supervised and unsupervised.
- In supervised learning, a desired output result for each input vector is required when the network is trained. An ANN of the supervised learning type, such as the multi-layer perceptron, uses the target result to guide the formation of the neural parameters. It is thus possible to make the neural network learn the behavior of the process under study.
- In unsupervised learning, the training of the network is entirely data-driven and no target results for the input data vectors are provided. An ANN of the unsupervised learning type, such as the self-organizing map, can be used for clustering the input data and find features inherent to the problem.

3

Self-Organizing Map (SOM)

- The Self-Organizing Map was developed by professor Kohonen. The SOM has been proven useful in many applications
- One of the most popular neural network models. It belongs to the category of competitive learning networks.
- Based on unsupervised learning, which means that no human intervention is needed during the learning and that little needs to be known about the characteristics of the input data.
- Use the SOM for clustering data without knowing the class memberships of the input data. The SOM can be used to detect features inherent to the problem and thus has also been called SOFM, the Self-Organizing Feature Map.

4

Self-Organizing Map (cont.)

- Provides a topology preserving mapping from the high dimensional space to map units. Map units, or neurons, usually form a two-dimensional lattice and thus the mapping is a mapping from high dimensional space onto a plane.
- The property of topology preserving means that the mapping preserves the relative distance between the points. Points that are near each other in the input space are mapped to nearby map units in the SOM. The SOM can thus serve as a cluster analyzing tool of high-dimensional data. Also, the SOM has the capability to generalize
- Generalization capability means that the network can recognize or characterize inputs it has never encountered before. A new input is assimilated with the map unit it is mapped to.

5

The general problem

- How can an algorithm learn without supervision?
 - I.e., without a “teacher”

6

Self-Organizing Maps (SOM's)

- Categorization method
- A neural network technique
- Unsupervised

7

Input & Output

- Training data: vectors, X
 - Vectors of length n

$$\left. \begin{array}{l} (x_{1,1}, x_{1,2}, \dots, x_{1,i}, \dots, x_{1,n}) \\ (x_{2,1}, x_{2,2}, \dots, x_{2,i}, \dots, x_{2,n}) \\ \dots \\ (x_{j,1}, x_{j,2}, \dots, x_{j,i}, \dots, x_{j,n}) \\ \dots \\ (x_{p,1}, x_{p,2}, \dots, x_{p,i}, \dots, x_{p,n}) \end{array} \right\} p \text{ distinct training vectors}$$
 - Vector components are real numbers
- Outputs
 - A vector, Y , of length m : $(y_1, y_2, \dots, y_i, \dots, y_m)$
 - Sometimes $m < n$, sometimes $m > n$, sometimes $m = n$
 - Each of the p vectors in the training data is classified as falling in one of m clusters or categories
 - That is: Which category does the training vector fall into?
- Generalization
 - For a new vector: $(x_{j,1}, x_{j,2}, \dots, x_{j,i}, \dots, x_{j,n})$
 - Which of the m categories (clusters) does it fall into?

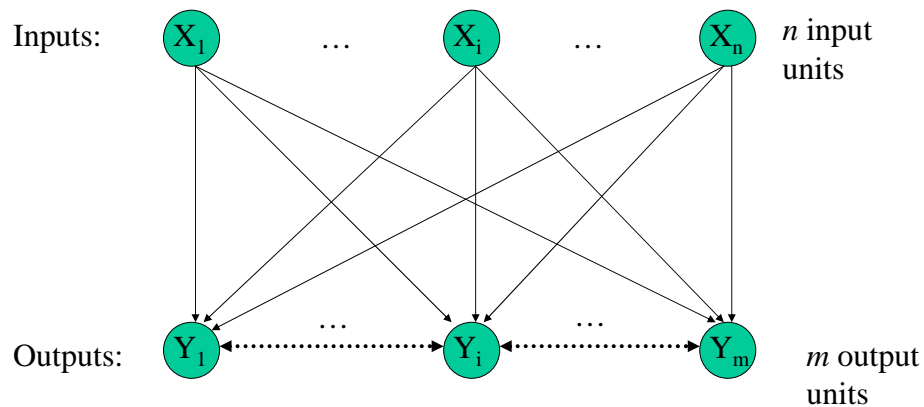
8

Network Architecture

- Two layers of units
 - Input: n units (length of training vectors)
 - Output: m units (number of categories)
- Input units fully connected with weights to output units
- Intra-layer (“lateral”) connections
 - Within output layer
 - Defined according to some topology
 - No weight between these connections, but used in algorithm for updating weights

9

Network Architecture



Note: There is one weight vector of length n associated with each output unit

10

Overall SOM Algorithm

- Training
 - Select output layer topology
 - Train weights connecting inputs to outputs
 - Topology is used, in conjunction with current mapping of inputs to outputs, to define which weights will be updated
 - Distance measure using the topology is reduced over time; reduces the number of weights that get updated per iteration
 - Learning rate is reduced over time
- Testing
 - Use weights from training

11

Output Layer Topology

- Often view output in spatial manner
 - E.g., a 1D or 2D arrangement
- 1D arrangement
 - Topology defines which output layer units are neighbors with which others
 - Have a function, $D(t)$, which gives output unit neighborhood as a function of time (iterations) of the training algorithm
- E.g., 3 output units



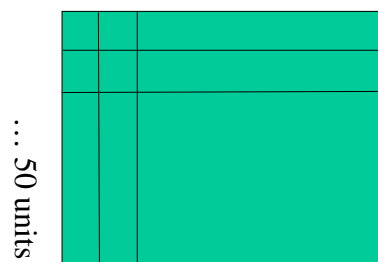
$D(t) = 1$ means update weight B & A if input maps onto B

12

Example: 2D Output Layer Topology



Fully-connected weights



- * Function, $D(t)$, can give output unit radius as a function of time (iterations) when training the weights
- * Usually, initially wide radius, changing to gradually narrower

13

Self Organizing *Maps*

- Often SOM's are used with 2D topographies connecting the output units
- In this way, the final output can be interpreted spatially, i.e., as a *map*

14

SOM Algorithm

- Select output layer network topology
 - Initialize current neighborhood distance, $D(0)$, to a positive value
 - Initialize weights from inputs to outputs to small random values
 - Let $t = 1$
 - While computational bounds are not exceeded do
 - 1) Select an input sample i_l
 - 2) Compute the square of the Euclidean distance of i_l from weight vectors (w_j) associated with each output node

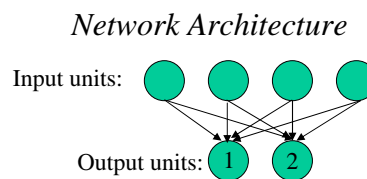
$$\sum_{k=1}^n (i_{l,k} - w_{j,k}(t))^2$$
 - 3) Select output node j^* that has weight vector with minimum value from step 2)
 - 4) Update weights to all nodes within a topological distance given by $D(t)$ from j^* , using the weight update rule:

$$w_j(t+1) = w_j(t) + \eta(t)(i_l - w_j(t))$$
 - 5) Increment t
 - Endwhile
- Learning rate generally decreases with time:
 $0 < \eta(t) \leq \eta(t-1) \leq 1$

From Mehotra et al. (1997), p. 189

Example Self-Organizing Map

- From Fausett (1994)
- $n = 4, m = 2$
- Training samples
 - i1: (1, 1, 0, 0)
 - i2: (0, 0, 0, 1)
 - i3: (1, 0, 0, 0)
 - i4: (0, 0, 1, 1)



What should we expect as outputs?

16

What are the Euclidean Distances Between the Data Samples?

- Training samples
 - i1: (1, 1, 0, 0)
 - i2: (0, 0, 0, 1)
 - i3: (1, 0, 0, 0)
 - i4: (0, 0, 1, 1)

	i1	i2	i3	i4
i1	0			
i2		0		
i3			0	
i4				0

17

Euclidean Distances Between Data Samples

- Training samples

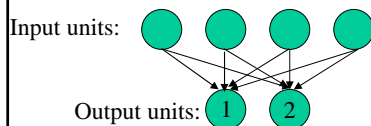
i1: (1, 1, 0, 0)

i2: (0, 0, 0, 1)

i3: (1, 0, 0, 0)

i4: (0, 0, 1, 1)

	i1	i2	i3	i4
i1	0			
i2	3	0		
i3	1	2	0	
i4	4	1	3	0



What might we expect from the SOM?

Example Details

- Training samples

i1: (1, 1, 0, 0)

i2: (0, 0, 0, 1)

i3: (1, 0, 0, 0)

i4: (0, 0, 1, 1)

- Let neighborhood = 0

- Only update weights associated with winning output unit (cluster) at each iteration

- Learning rate

$\eta(t) = 0.6; 1 \leq t \leq 4$

$\eta(t) = 0.5 \eta(1); 5 \leq t \leq 8$

$\eta(t) = 0.5 \eta(5); 9 \leq t \leq 12$

etc.

- Initial weight matrix

(random values between 0 and 1) $\begin{cases} \text{Unit 1:} & \begin{bmatrix} .2 & .6 & .5 & .9 \end{bmatrix} \\ \text{Unit 2:} & \begin{bmatrix} .8 & .4 & .7 & .3 \end{bmatrix} \end{cases}$

$$d^2 = (\text{Euclidean distance})^2 = \sum_{k=1}^n (i_{l,k} - w_{j,k}(t))^2$$

$$\text{Weight update: } w_j(t+1) = w_j(t) + \eta(t)(i_l - w_j(t))$$

19

Problem: Calculate the weight updates for the first four steps

First Weight Update

i1: (1, 1, 0, 0)

i2: (0, 0, 0, 1)

i3: (1, 0, 0, 0)

i4: (0, 0, 1, 1)

- Training sample: i1

- Unit 1 weights

- $d^2 = (.2-1)^2 + (.6-1)^2 + (.5-0)^2 + (.9-0)^2 = 1.86$

- Unit 2 weights

- $d^2 = (.8-1)^2 + (.4-1)^2 + (.7-0)^2 + (.3-0)^2 = .98$

- Unit 2 wins

- Weights on winning unit are updated

$$\begin{array}{l} \text{Unit 1: } \begin{bmatrix} .2 & .6 & .5 & .9 \end{bmatrix} \\ \text{Unit 2: } \begin{bmatrix} .8 & .4 & .7 & .3 \end{bmatrix} \end{array}$$

- Giving an updated weight matrix:

$$\begin{aligned} \text{new-unit2-weights} &= [.8 \ .4 \ .7 \ .3] + 0.6([1 \ 1 \ 0 \ 0] - [.8 \ .4 \ .7 \ .3]) = \\ &= [.92 \ .76 \ .28 \ .12] \end{aligned}$$

$$\begin{array}{l} \text{Unit 1: } \begin{bmatrix} .2 & .6 & .5 & .9 \end{bmatrix} \\ \text{Unit 2: } \begin{bmatrix} .92 & .76 & .28 & .12 \end{bmatrix} \end{array}$$

20

Second Weight Update

i1: (1, 1, 0, 0)

i2: (0, 0, 0, 1)

i3: (1, 0, 0, 0)

i4: (0, 0, 1, 1)

- Training sample: i2

- Unit 1 weights

- $d^2 = (.2-0)^2 + (.6-0)^2 + (.5-0)^2 + (.9-1)^2 = .66$

- Unit 2 weights

- $d^2 = (.92-0)^2 + (.76-0)^2 + (.28-0)^2 + (.12-1)^2 = 2.28$

- Unit 1 wins

- Weights on winning unit are updated

$$\begin{array}{l} \text{Unit 1: } \begin{bmatrix} .2 & .6 & .5 & .9 \end{bmatrix} \\ \text{Unit 2: } \begin{bmatrix} .92 & .76 & .28 & .12 \end{bmatrix} \end{array}$$

- Giving an updated weight matrix:

$$\begin{aligned} \text{new-unit1-weights} &= [.2 \ .6 \ .5 \ .9] + 0.6([0 \ 0 \ 0 \ 1] - [.2 \ .6 \ .5 \ .9]) = \\ &= [.08 \ .24 \ .20 \ .96] \end{aligned}$$

$$\begin{array}{l} \text{Unit 1: } \begin{bmatrix} .08 & .24 & .20 & .96 \end{bmatrix} \\ \text{Unit 2: } \begin{bmatrix} .92 & .76 & .28 & .12 \end{bmatrix} \end{array}$$

21

Third Weight Update

i1: (1, 1, 0, 0)

i2: (0, 0, 0, 1)

i3: (1, 0, 0, 0)

i4: (0, 0, 1, 1)

- Training sample: i3

- Unit 1 weights

- $d^2 = (.08-1)^2 + (.24-0)^2 + (.2-0)^2 + (.96-0)^2 = 1.87$

- Unit 2 weights

- $d^2 = (.92-1)^2 + (.76-0)^2 + (.28-0)^2 + (.12-0)^2 = 0.68$

- Unit 2 wins

- Weights on winning unit are updated

$$\begin{array}{l} \text{Unit 1: } \begin{bmatrix} .08 & .24 & .20 & .96 \\ .92 & .76 & .28 & .12 \end{bmatrix} \\ \text{Unit 2: } \end{array}$$

- Giving an updated weight matrix:

$$\begin{aligned} \text{new-unit2-weights} &= [.92 \quad .76 \quad .28 \quad .12] + 0.6([1 \ 0 \ 0 \ 0] - [.92 \quad .76 \quad .28 \quad .12]) = \\ &= [.97 \quad .30 \quad .11 \quad .05] \end{aligned}$$

$$\begin{array}{l} \text{Unit 1: } \begin{bmatrix} .08 & .24 & .20 & .96 \\ .97 & .30 & .11 & .05 \end{bmatrix} \\ \text{Unit 2: } \end{array}$$

22

Fourth Weight Update

i1: (1, 1, 0, 0)

i2: (0, 0, 0, 1)

i3: (1, 0, 0, 0)

i4: (0, 0, 1, 1)

- Training sample: i4

- Unit 1 weights

- $d^2 = (.08-0)^2 + (.24-0)^2 + (.2-1)^2 + (.96-1)^2 = .71$

- Unit 2 weights

- $d^2 = (.97-0)^2 + (.30-0)^2 + (.11-1)^2 + (.05-1)^2 = 2.74$

- Unit 1 wins

- Weights on winning unit are updated

$$\begin{array}{l} \text{Unit 1: } \begin{bmatrix} .08 & .24 & .20 & .96 \\ .97 & .30 & .11 & .05 \end{bmatrix} \\ \text{Unit 2: } \end{array}$$

- Giving an updated weight matrix:

$$\begin{aligned} \text{new-unit1-weights} &= [.08 \quad .24 \quad .20 \quad .96] + 0.6([0 \ 0 \ 1 \ 1] - [.08 \quad .24 \quad .20 \quad .96]) = \\ &= [.03 \quad .10 \quad .68 \quad .98] \end{aligned}$$

$$\begin{array}{l} \text{Unit 1: } \begin{bmatrix} .03 & .10 & .68 & .98 \\ .97 & .30 & .11 & .05 \end{bmatrix} \\ \text{Unit 2: } \end{array}$$

23

Applying the SOM Algorithm

Data sample utilized

time (t)	1	2	3	4	D(t)	$\eta(t)$
1	Unit 2				0	0.6
2		Unit 1			0	0.6
3			Unit 2		0	0.6
4				Unit 1	0	0.6

‘winning’ output unit

After many iterations (epochs)
through the data set:

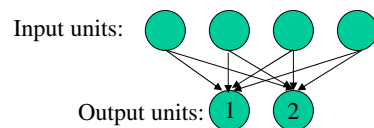
$$\begin{array}{l} \text{Unit 1:} \\ \text{Unit 2:} \end{array} \begin{bmatrix} 0 & 0 & .5 & 1.0 \\ 1.0 & .5 & 0 & 0 \end{bmatrix}$$

Did we get the clustering that we expected?

24

Training samples

i1: (1, 1, 0, 0)
i2: (0, 0, 0, 1)
i3: (1, 0, 0, 0)
i4: (0, 0, 1, 1)

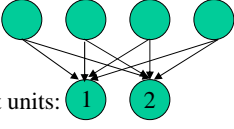


Weights

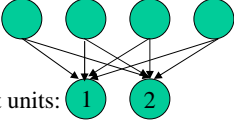
$$\begin{array}{l} \text{Unit 1:} \\ \text{Unit 2:} \end{array} \begin{bmatrix} 0 & 0 & .5 & 1.0 \\ 1.0 & .5 & 0 & 0 \end{bmatrix}$$

What clusters do the
data samples fall into?

25

<i>Training samples</i>	<h2 style="margin: 0;">Solution</h2>		<i>Weights</i>
i1: (1, 1, 0, 0) i2: (0, 0, 0, 1) i3: (1, 0, 0, 0) i4: (0, 0, 1, 1)	Input units:		Unit 1: $\begin{bmatrix} 0 & 0 & .5 & 1.0 \end{bmatrix}$ Unit 2: $\begin{bmatrix} 1.0 & .5 & 0 & 0 \end{bmatrix}$
<div style="display: flex; justify-content: space-between;"> <div style="width: 45%;"> <ul style="list-style-type: none"> • Sample: i1 <ul style="list-style-type: none"> – Distance from unit1 weights <ul style="list-style-type: none"> • $(1-0)^2 + (1-0)^2 + (0-.5)^2 + (0-1.0)^2 = 1+1+.25+1=3.25$ – Distance from unit2 weights <ul style="list-style-type: none"> • $(1-1)^2 + (1-.5)^2 + (0-0)^2 + (0-0)^2 = 0+.25+0+0=.25$ (winner) • Sample: i2 <ul style="list-style-type: none"> – Distance from unit1 weights <ul style="list-style-type: none"> • $(0-0)^2 + (0-0)^2 + (0-.5)^2 + (1-1.0)^2 = 0+0+.25+0$ (winner) – Distance from unit2 weights <ul style="list-style-type: none"> • $(0-1)^2 + (0-.5)^2 + (0-0)^2 + (1-0)^2 = 1+.25+0+1=2.25$ </div> <div style="width: 50%; text-align: right;"> $d^2 = (\text{Euclidean distance})^2 = \sum_{k=1}^n (i_{l,k} - w_{j,k}(t))^2$ </div> </div>			

26

<i>Training samples</i>	<h2 style="margin: 0;">Solution</h2>		<i>Weights</i>
i1: (1, 1, 0, 0) i2: (0, 0, 0, 1) i3: (1, 0, 0, 0) i4: (0, 0, 1, 1)	Input units:		Unit 1: $\begin{bmatrix} 0 & 0 & .5 & 1.0 \end{bmatrix}$ Unit 2: $\begin{bmatrix} 1.0 & .5 & 0 & 0 \end{bmatrix}$
<div style="display: flex; justify-content: space-between;"> <div style="width: 45%;"> <ul style="list-style-type: none"> • Sample: i3 <ul style="list-style-type: none"> – Distance from unit1 weights <ul style="list-style-type: none"> • $(1-0)^2 + (0-0)^2 + (0-.5)^2 + (0-1.0)^2 = 1+0+.25+1=2.25$ – Distance from unit2 weights <ul style="list-style-type: none"> • $(1-1)^2 + (0-.5)^2 + (0-0)^2 + (0-0)^2 = 0+.25+0+0=.25$ (winner) • Sample: i4 <ul style="list-style-type: none"> – Distance from unit1 weights <ul style="list-style-type: none"> • $(0-0)^2 + (0-0)^2 + (1-.5)^2 + (1-1.0)^2 = 0+0+.25+0$ (winner) – Distance from unit2 weights <ul style="list-style-type: none"> • $(0-1)^2 + (0-.5)^2 + (1-0)^2 + (1-0)^2 = 1+.25+1+1=3.25$ </div> <div style="width: 50%; text-align: right;"> $d^2 = (\text{Euclidean distance})^2 = \sum_{k=1}^n (i_{l,k} - w_{j,k}(t))^2$ </div> </div>			

27

Conclusion

- Samples i1, i3 cluster with unit 2
- Samples i2, i4 cluster with unit 1

28

What about generalization?

Training samples

i1: (1, 1, 0, 0)

i2: (0, 0, 0, 1)

i3: (1, 0, 0, 0)

i4: (0, 0, 1, 1)

- New data sample
i5: (1, 1, 1, 0)
- What unit *should* this cluster with?
- What unit *does* this cluster with?

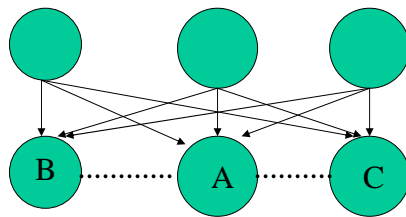
29

Example 5.7

- pp. 191-194 of Mehotra et al. (1997)

$n = m = 3$

Input units:



Output units

Input Data Samples

I1	1.1	1.7	1.8
I2	0	0	0
I3	0	0.5	1.5
I4	1	0	0
I5	0.5	0.5	0.5
I6	1	1	1

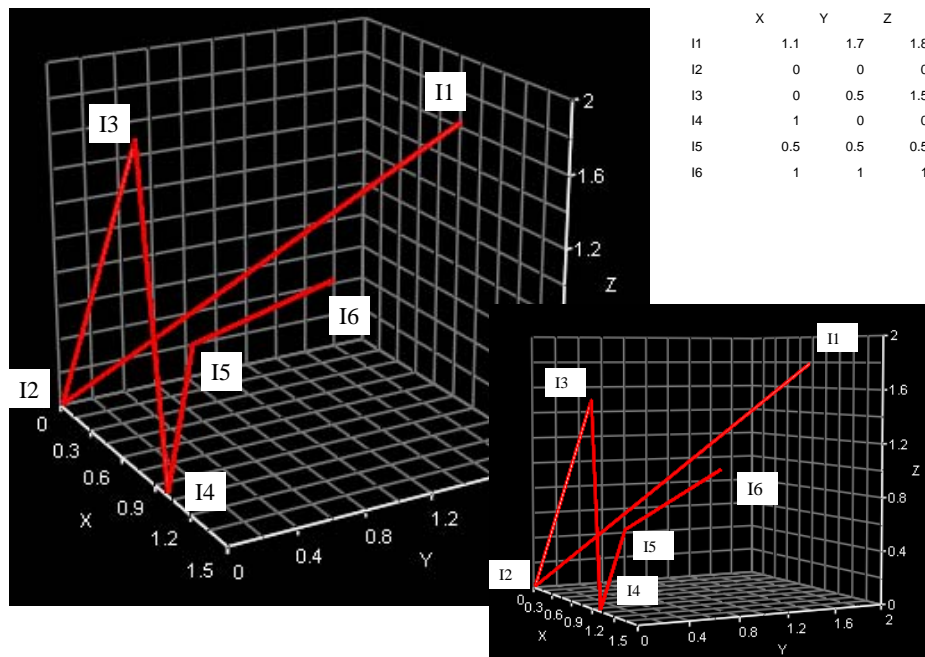
- What do we expect as outputs from this example?

Squared Euclidean Distances of One Input Value to Another

	I1	I2	I3	I4	I5	I6
I1	0					
I2	7.34	0				
I3	2.74	2.5	0			
I4	6.14	1	3.5	0		
I5	3.49	0.75	1.25	0.75	0	
I6	1.14	3	1.5	2	0.75	0

31

Data Samples Plotted as X,Y,Z points in 3D Space



Example Details: Neighborhood distance & Learning Rate

- Neighborhood distance
 $D(t)$ gives output unit neighborhood as a function of time
 $0 \leq t \leq 6, D(t) = 1$
 $t > 6, D(t) = 0$
- Learning rate also varies with time
 $0 \leq t \leq 5, \eta(t) = 0.6$
 $6 \leq t \leq 12, \eta(t) = .25$
 $t > 12, \eta(t) = 0.1$

Initial weights

W _a	0.2	0.7	0.3
W _b	0.1	0.1	0.9
W _c	1	1	1

33

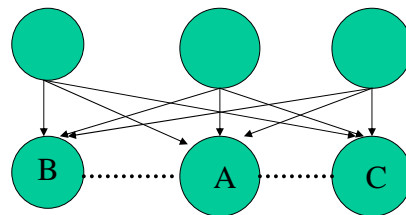
<http://www.cprince.com/courses/cs5541/lectures/SOM/SOM.xls>

First Iteration

- Use input data in order I_1, I_2, \dots, I_6
 - Start with I_1 : 1.1 1.7 1.8
- 1) Compute Euclidean distance of data from current weight vectors for output units

Initial weights

W_a	0.2	0.7	0.3
W_b	0.1	0.1	0.9
W_c	1	1	1

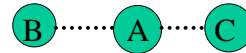


- 2) Compute weight updates

34

Applying the SOM Algorithm

Data sample utilized



time (t)	1	2	3	4	5	6	D(t)	$\eta(t)$	Weights Updated
1	C						1	0.5	C, A
2		B					1	0.5	B, A
3			A				1	0.5	A, B, C
4				B			1	0.5	B, A
5					A		1	0.5	A, B, C
6						C	1	0.5	C, A
7	C						0	0.25	C
8		B					0	0.25	B
9			C				0	0.25	C
10				B			0	0.25	B
11					B		0	0.25	B
12						A	0	0.25	A
13	C						0	0.1	C
14		B					0	0.1	B
15			C				0	0.1	C
16				B			0	0.1	B
17					B		0	0.1	B
18						A	0	0.1	A

35

'winning' output node

Results: Classification & Weights

Classification

Output node	Data sample
A	6
B	2, 4, 5
C	1, 3

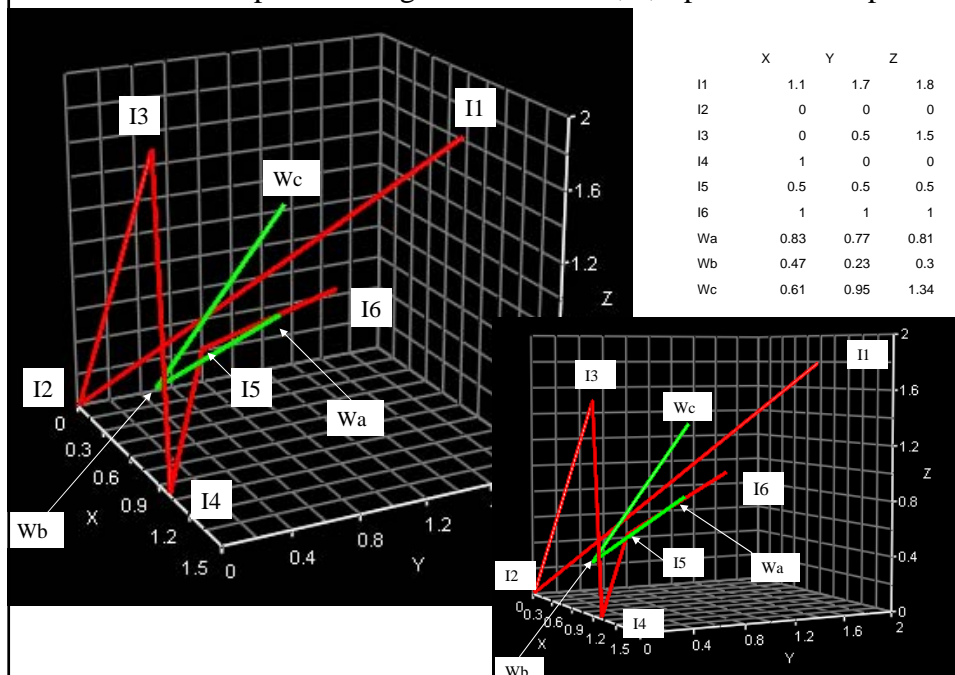
Weights after 15 time steps

Wa	0.83	0.77	0.81
Wb	0.47	0.23	0.3
Wc	0.61	0.95	1.34

Weights after 21 time steps

Wa	0.847	0.793	0.829
Wb	0.46863	0.21267	0.2637
Wc	0.659	1.025	1.386

Data Samples & Weights Plotted as X,Y,Z points in 3D Space



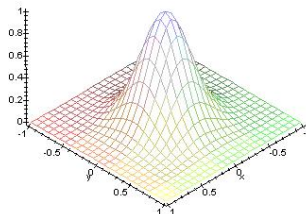
Another SOM example

- More typically, SOM's are used with 2D topographies connecting the output units
- In this way, the final output can be interpreted spatially, i.e., as a map

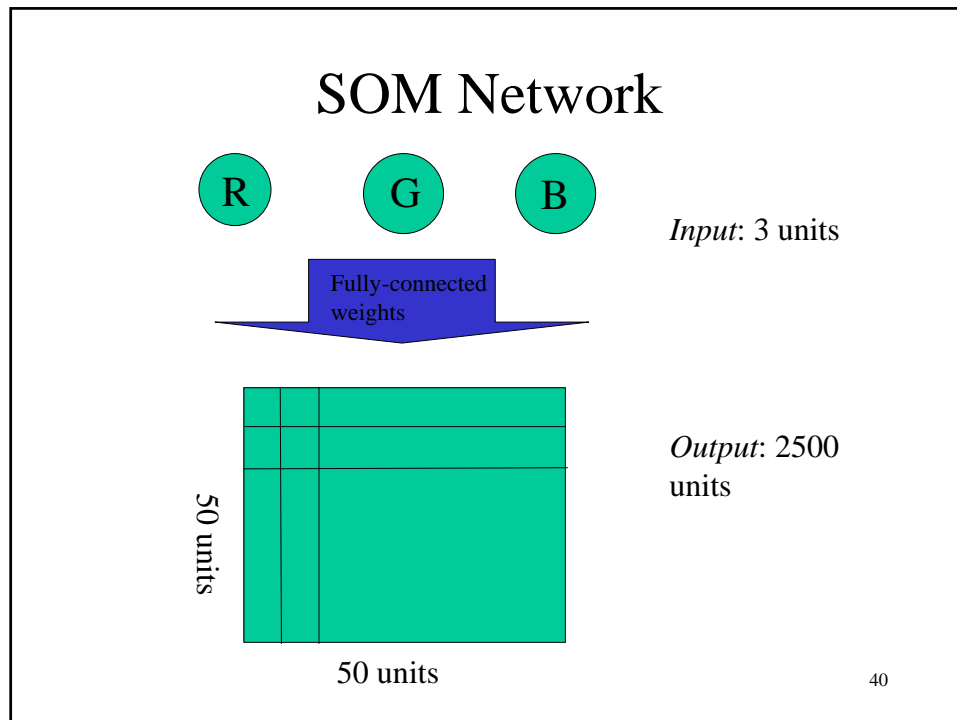
38

Self-Organizing Colors

- Inputs
 - 3 input units: R, G, B
 - Randomly selected RGB colors
- Outputs
 - 2500 units
 - Connected in a 2D matrix
 - *Amount* neighbor weights changed specified by a 2D Gaussian
 - Width of Gaussian decreases with iterations of training



<http://davis.wpi.edu/~matt/courses/soms/applet.html>



Algorithm modifications

- Random selection of winning output if multiple winning outputs
- Weight update modified to include a contribution factor

$$w_j(t+1) = w_j(t) + \eta(t)c(t, i_l, w_j(t))(i_l - w_j(t))$$

Visualization

- Map weights onto colors

<http://davis.wpi.edu/~matt/courses/soms/applet.html>

42

U-matrix (Unified distance matrix)

- U-matrix representation of the Self-Organizing Map visualizes the distances between the neurons. The distance between the adjacent neurons is calculated and presented with different colorings between the adjacent nodes. A dark coloring between the neurons corresponds to a large distance and thus a gap between the codebook values in the input space. A light coloring between the neurons signifies that the codebook vectors are close to each other in the input space. Light areas can be thought as clusters and dark areas as cluster separators. This can be a helpful presentation when one tries to find clusters in the input data without having any a priori information about the clusters.

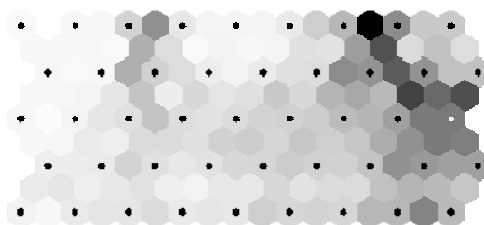


Figure: U-matrix representation of the Self-Organizing Map

43

U-Matrix Visualization

- Provides a simple way to visualize cluster boundaries on the map
- Simple algorithm:
 - for each node in the map, compute the average of the distances between its weight vector and those of its immediate neighbors
- Average distance is a measure of a node's similarity between it and its neighbors

44

U-Matrix Visualization

- Interpretation
 - one can encode the U-Matrix measurements as grayscale values in an image, or as altitudes on a terrain
 - landscape that represents the document space: the valleys, or dark areas are the clusters of data, and the mountains, or light areas are the boundaries between the clusters

45