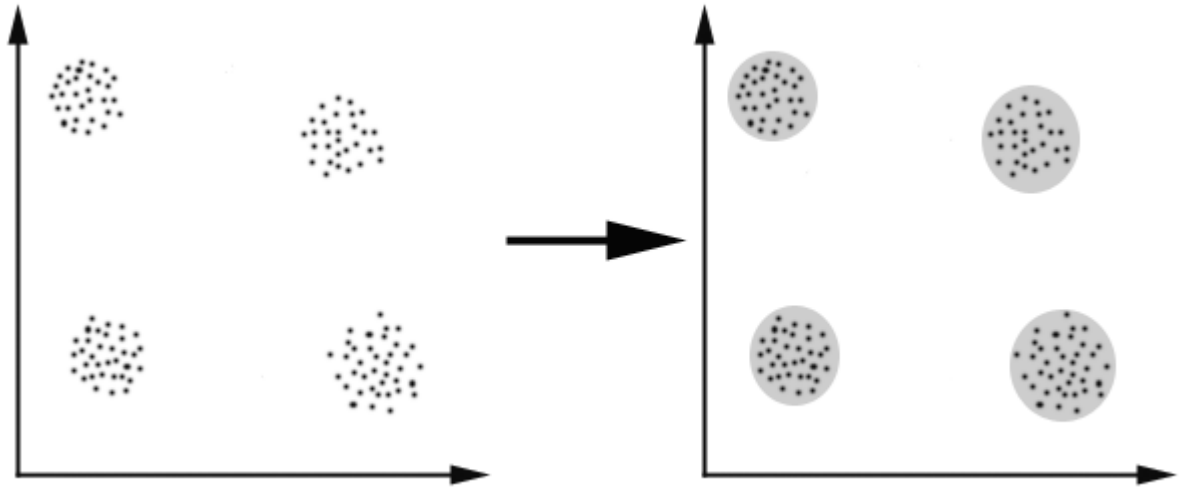
Self-Organizing Maps

Rhee, Je-Keun

What is clustering?



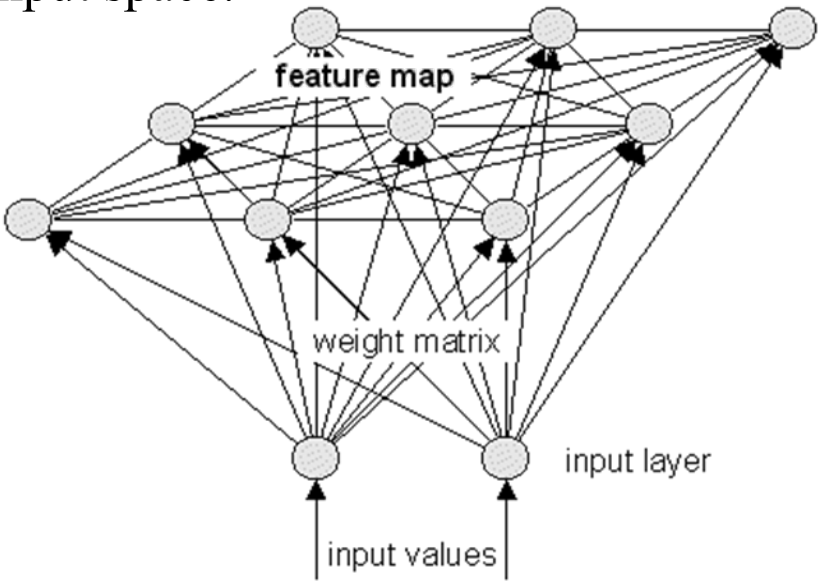
Find K clusters (or a classification that consists of K clusters) so that the objects of one cluster are similar to each other whereas objects of different clusters are dissimilar. (Bacher 1996)

Goal of clustering

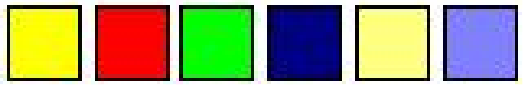
- The goal of clustering is to determine the intrinsic grouping in a set of unlabeled data.
- A *cluster* is a collection of objects which are *similar* between them and are *dissimilar* to the objects belonging to other clusters.
- Possible Applications
  - Marketing: finding groups of customers with similar behavior given a large database of customer data containing their properties and past buying records
  - Libraries: book ordering
  - City-planning: identifying groups of houses according to their house type, value and geographical location;
  - Earthquake studies: clustering observed earthquake epicenters to identify dangerous zones;

Self-organizing map (SOM)

- SOM provides a topology preserving mapping from a multidimensional input space onto neurons.
- Neurons or map units usually form a two-dimensional lattice.
- Self-organizing maps are different from other artificial neural networks in the sense that they use a neighborhood function to preserve the topological properties of the input space.



Sample data



- The colors are represented in three dimensions (red, green, blue).
- The input vectors will be:

  R = <255, 0, 0>

  G = <0, 255, 0>

  B = <0, 0, 255>

  .....

- In this case, for example, one would expect the dark blue and the greys to end up near each other on a good map and yellow close to both the red and the green.

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**

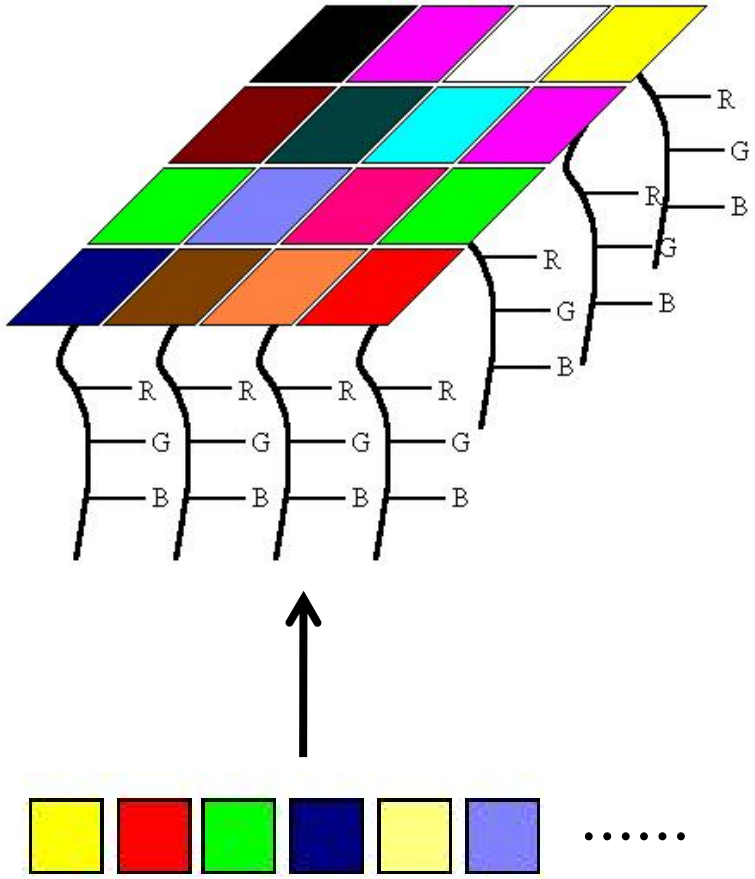
$$w_j(t+1) = w_j(t) + \eta(t)(i_\ell(t) - w_j(t)),$$

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

  4. **Increment t;**

end-while.

Initialize output weight map



Get best matched unit

- The weight with the shortest distance is the winner.
- If we have chosen green which is of the value (0,255,0), The color light green will be closer to green than red.

Light Green = (150, 255, 150)

Red = (255, 0, 0)

$\text{Dist}_{\text{Light Green}} : \sqrt{(150-0)^2 + (255-255)^2 + (150-0)^2}$

$\text{Dist}_{\text{red}} : \sqrt{(255-0)^2 + (0-255)^2 + (255-0)^2}$

Update the weights in the neighborhood of BMU

- Every node, considered as neighbors of a winning node, should adjust their weights, as follows:

$$Current(t+1) = Current(t) + L(t) * (Input\_vector(t) - Current(t))$$

  - t represents time-steps.
  - $L(t)$ is a *learning_rate* which would be decreased with time.

  - Usually the decay of learning rate can be calculated using following equation:

$$L(t) = L_0 \exp(-\frac{t}{\lambda})$$

Update the weights in the neighborhood of BMU

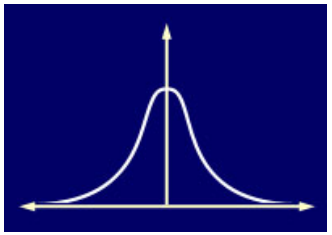
- In addition, the effect of learning might be proportional to the distance from the BMU.
- The amount of learning should fade over, similar to the Gaussian decay.

$$Current(t+1) = Current(t) + \Theta(t)L(t) * (Input\_vector(t) - Current(t))$$

  - Θ *i*s the amount of influence due to a distance from the BMU on its learning

$$\Theta(t) = \exp(-\frac{dist^2}{2\sigma^2(t)})$$

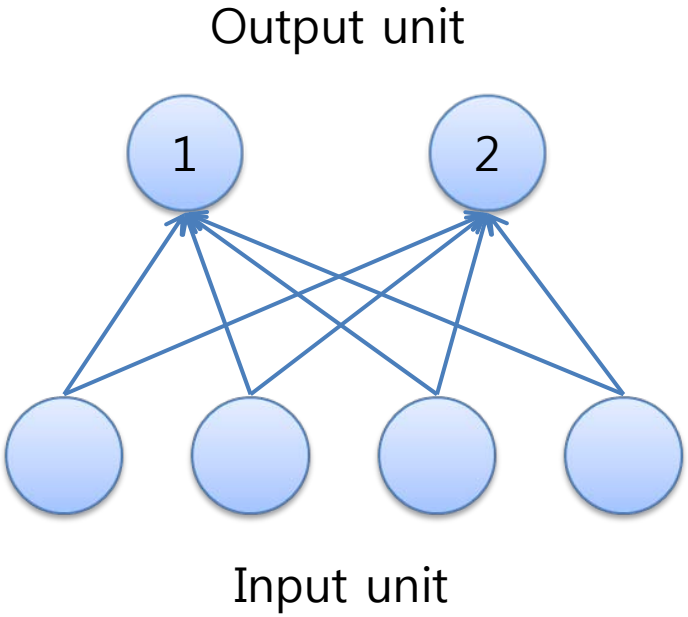
$$\sigma(t) = \sigma_0 \exp(-\frac{t}{\lambda})$$



Example

- Training samples

s1: (1, 1, 0, 0)
s2: (0, 0, 0, 1)
s3: (1, 0, 0, 0)
s4: (0, 0, 1, 1)



Example details

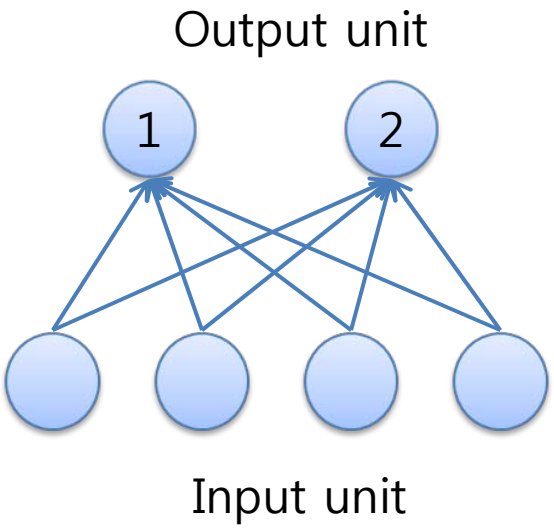
- Let neighborhood = 0
  - Only update associated with winning output unit at each cluster
- Learning rate

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

  $\eta(t) = 0.5;\ 5 <= t <= 8$

  $\eta(t) = 0.4;\ 9 <= t <= 12$
- Initial weights
  - Random values
  - For example, Unit1=(0.2, 0.6, 0.5, 0.9)

    Unit2=(0.8, 0.4, 0.7, 0.3)



For first training sample s1

s1: (1, 1, 0, 0)
s2: (0, 0, 0, 1)
s3: (1, 0, 0, 0)
s4: (0, 0, 1, 1)

Unit1: (0.2, 0.6, 0.5, 0.9)
Unit2: (0.8, 0.4, 0.7, 0.3)

- Unit1
  - $d^2 = (.2 - 1)^2 + (.6 - 1)^2 + (.5 - 0)^2 + (.9 - 0)^2 = 1.86$
- Unit2
  - $d^2 = (.8 - 1)^2 + (.4 - 1)^2 + (.7 - 0)^2 + (.3 - 0)^2 = 0.98$
- Weights on winning unit are updated

For first training sample s1

s1: (1, 1, 0, 0)
s2: (0, 0, 0, 1)
s3: (1, 0, 0, 0)
s4: (0, 0, 1, 1)

Unit1: (0.2, 0.6, 0.5, 0.9)
Unit2: (0.8, 0.4, 0.7, 0.3)

- Weight update

$$(.8 \quad .4 \quad .7 \quad .3) + 0.6 * ((1 \quad 1 \quad 0 \quad 0) - (.8 \quad .4 \quad .7 \quad .3))$$
$$= (.92 \quad .76 \quad .28 \quad .12)$$

  - Unit1: (0.2, 0.6, 0.5, 0.9)
  - Unit2: (0.92, 0.76, 0.28, 0.12)

For second training sample s2

s1: (1, 1, 0, 0)
s2: (0, 0, 0, 1)
s3: (1, 0, 0, 0)
s4: (0, 0, 1, 1)

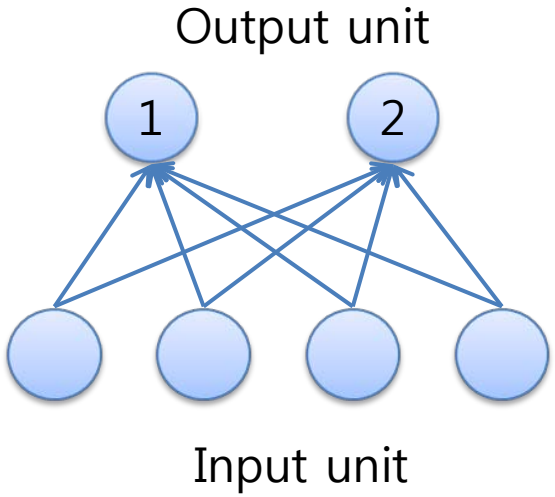
Unit1: (0.2, 0.6, 0.5, 0.9)
Unit2: (0.92, 0.76, 0.28, 0.12)

- Unit1
  - $d^2 = (.2 - 0)^2 + (.6 - 0)^2 + (.5 - 0)^2 + (.9 - 1)^2 = 0.66$
- Unit2
  - $d^2 = (.92 - 0)^2 + (.76 - 0)^2 + (.28 - 0)^2 + (.12 - 1)^2 = 2.28$
- Weight update

$$(.2 \quad .6 \quad .5 \quad .9) + 0.6 * ((0 \quad 0 \quad 0 \quad 1) - (.2 \quad .6 \quad .5 \quad .9)) = (.08 \quad .24 \quad .20 \quad .96)$$

  - Unit1: (0.08, 0.24, 0.20, 0.96)
  - Unit2: (0.92, 0.76, 0.28, 0.12)

- After many iterations (epochs) through the data:
  - Unit1: (0, 0, 0.5, 1.0)
  - Unit2: (1.0, 0.5, 0, 0)



- What clusters do the data samples fall into?

s1: (1, 1, 0, 0)
s2: (0, 0, 0, 1)
s3: (1, 0, 0, 0)
s4: (0, 0, 1, 1)

Clustering

s1: (1, 1, 0, 0)
s2: (0, 0, 0, 1)
s3: (1, 0, 0, 0)
s4: (0, 0, 1, 1)

Unit1: (0, 0, 0.5, 1.0)
Unit2: (1.0, 0.5, 0, 0)

- For sample s1
  - Unit1: $d^2 = (0 - 1)^2 + (0 - 1)^2 + (.5 - 0)^2 + (1.0 - 0)^2 = 3.25$
  - Unit2: $d^2 = (1 - 1)^2 + (.5 - 1)^2 + (0 - 0)^2 + (0 - 0)^2 = 0.25$

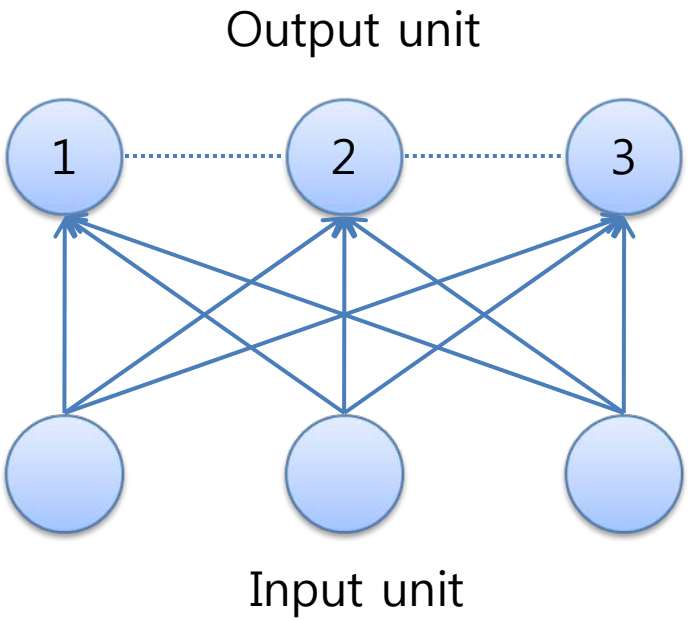
⋮

- s1 and s3 cluster into unit 2
- s2 and s4 cluster into unit 1

Another example

- Training samples

s1: (1.1, 1.7, 1.8)
s2: (0, 0, 0)
s3: (0, 0.5, 1.5)
s4: (1, 0, 0)
s5: (0.5, 0.5, 0.5)
s6: (1, 1, 1)



Example details

- Neighborhood distance
  - D(t) gives output unit neighborhood as a function of time

    $0 <= t <= 6$, $D(t) = 1$

    $t > 6$, $D(t) = 0$

- Learning rate

  $0 <= t <= 5$, $\eta(t) = 0.6$

  $6 <= t <= 12$, $\eta(t) = 0.25$

  $t > 12$, $\eta(t) = 0.1$

- Initial weights

  Unit1: (0.2, 0.7, 0.3)

  Unit2: (0.1, 0.1, 0.9)

  Unit3: (1, 1, 1)