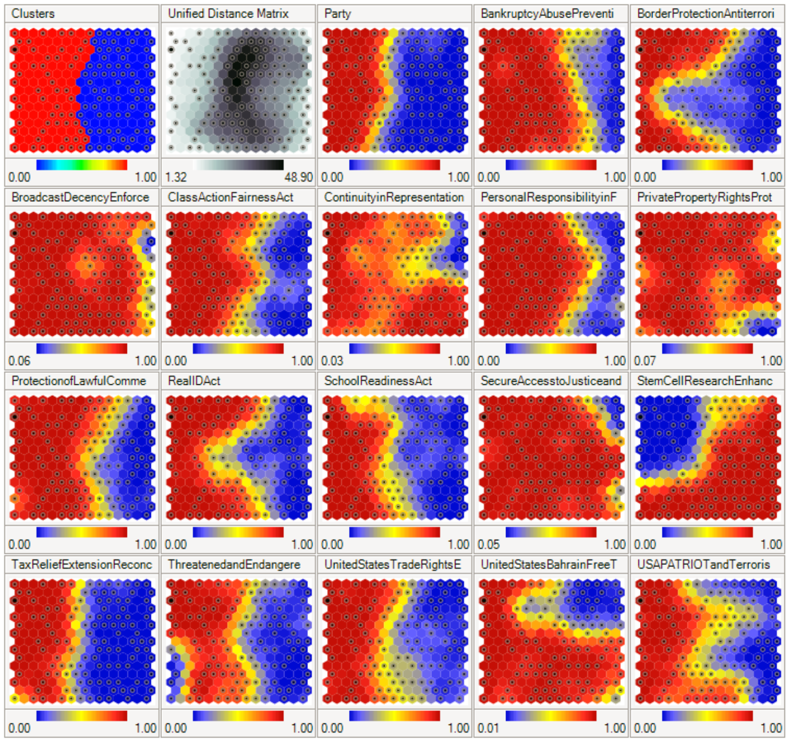
SELF-ORGANIZING FEATURE MAP (SOFM)

Shingchern D. You

Motivation

- Want to visualize high-dimensional data in 2-D
- Can be also used for clustering (Figure from https://en.wikipedia.org/wiki/Self-organizing_map#/media/File:Synapse_Self-Organizing_Map.png)



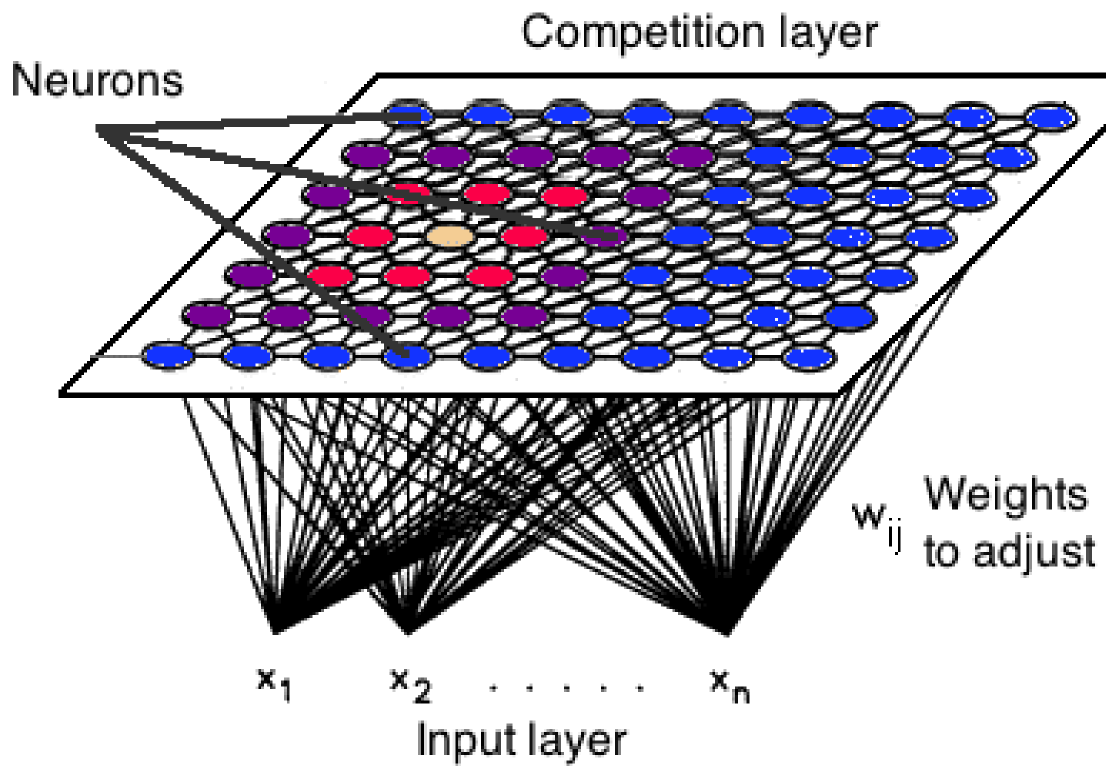
Operating modes

- Training mode
- Mapping mode

Basic network architecture

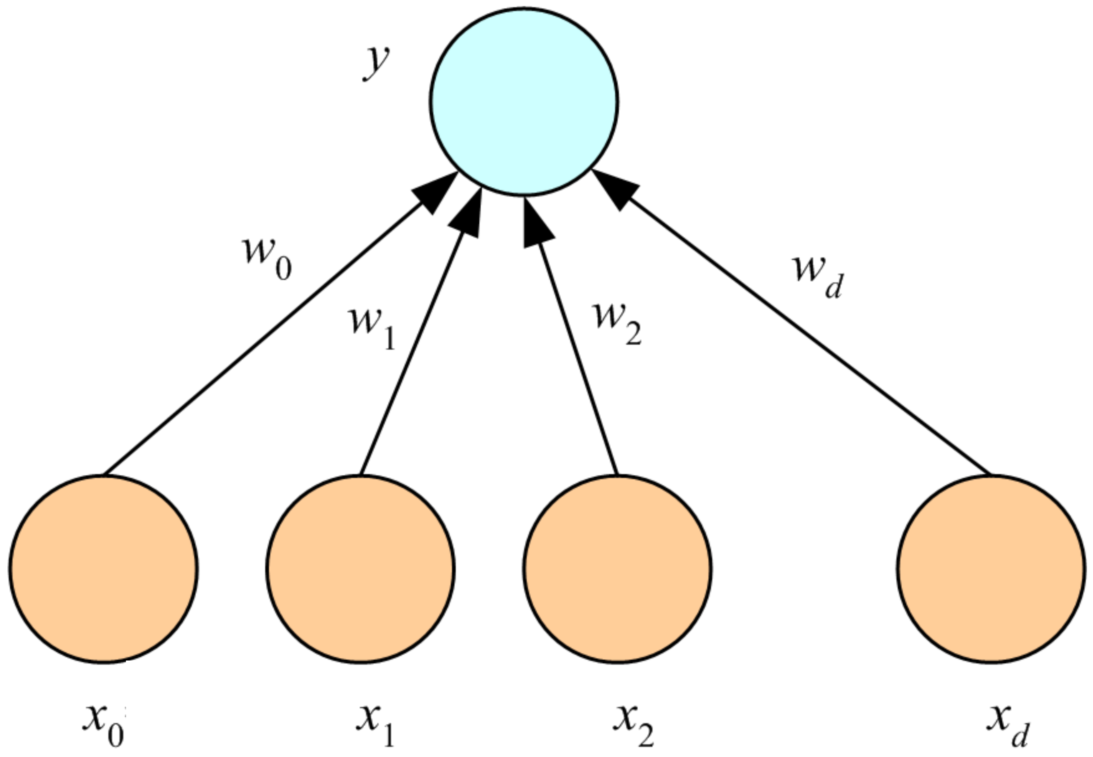
- Input nodes connect to all nodes in competition layer

Source: https://analyticsindiamag.com/6-types-of-artificial-neural-networks-currently-being-used-in-todays-technology



Basic network architecture

- Use a vector to represent the weights from input nodes to a competition node



Basic network architecture

- In a typical neural network, $y = f(\sum x_k w_k)$
- In SOFM, we use $y = \|\boldsymbol{x} - \boldsymbol{w}\|^2$
- The neuron with largest y is called BMU (Best Match Unit)
- The weights around BMU are updated with different rates (closer BMU gets larger change)
- The learning rate is a function of time (iteration)

Training SOFM

- The training procedure
- (1) Initialize the weights (all weights are random numbers between 0 and 1)
- (2) Let (i_0, j_0) be the location of BMU, $\boldsymbol{x}(t)$ be the input vector at time t , and $\boldsymbol{w}_{(\boldsymbol{i},\boldsymbol{j})}(t)$ be the weight vector for neuron at $(i, j\ )$ at time t
- Note that typically we have many input vectors, one vector corresponding to one data point

Training SOFM

- (3) Update the weights of all neuros by $\boldsymbol{w}_{(\boldsymbol{i},\boldsymbol{j})}(t+1) = \boldsymbol{w}_{(\boldsymbol{i},\boldsymbol{j})}(t) + \eta(t) \cdot \gamma_{(i,j)}(t) \cdot [\boldsymbol{x}(t) - \boldsymbol{w}_{(\boldsymbol{i},\boldsymbol{j})}(t)]$

where $\eta(t) = \eta_0 \exp\left(\frac{t}{\lambda}\right)$

$$\gamma_{(i,j)}(t) = \exp[\frac{-\|(i,j) - (i_0, j_0)\|^2}{2\sigma^2(t)}]$$

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

$\lambda = NumIteration/MapRadius$ (or other Eq.)

Training SOFM

- Repeat until required iterations reached
- It is observed that the range of “neighboring” neurons (received some weights updates) gets smaller over time
- Need to properly set the parameters η_0 and σ_0 (and maybe λ)
- Typically, over 1000 iterations are necessary

Training SOFM

- Note that there are several variations on training SOFM
- The above algorithm is just for illustration purpose
- We may split the training into two phases: Ordering or self-organizing phase and Convergence phase

Training SOFM

- How to initialize the SOFM is a trick problem
- Refer to wiki for more details

Using SOFM

- Several methods to show the map
- (1) Display the weights (obvious)
- (2) BMU window
- (3) Error map

Using SOFM

- BMU window: Once training is done, plot BMU for each input
- Error map: To calculate an error map, loop through every map node of the network. Add up the distance (not the physical distance, but the weight distance. This is exactly the same as how the BMU is calculated) from the node we're currently evaluating, to each of it's neighbors (also called U-Matrix)

Using SOFM

- Iris data (from: https://en.wikipedia.org/wiki/Self-organizing_map)

First 3 features in RGB

Weights

Error map

BMU + Error Map