

Self-Organizing Maps(SOMs)

Bhargavi

Dr R Bhargavi
Vellore Institute of Technology

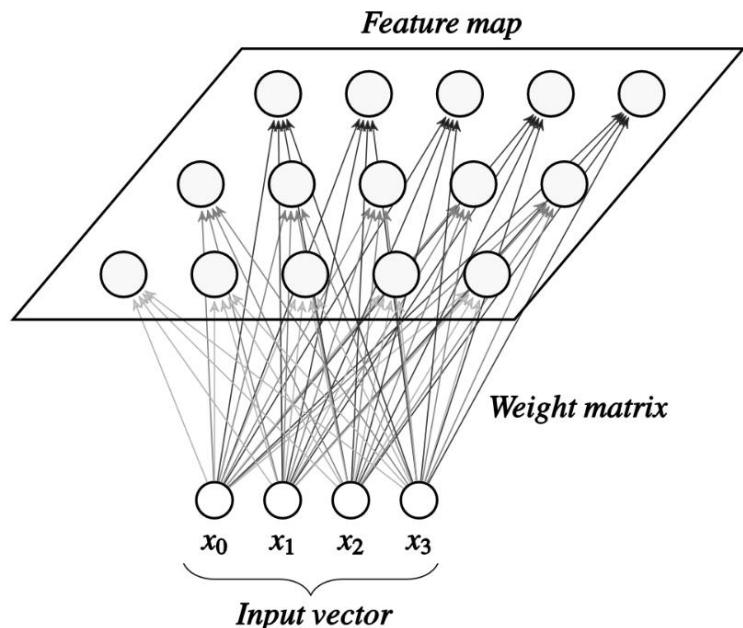
Introduction

- Also known as Kohonen maps after their inventor Teuvo Kohonen.
- SOMs are a type of artificial neural networks used for unsupervised learning.
- Used for dimensionality reduction, data visualization, and clustering.
- SOMs are known for preserving the topological relationships of the input data, meaning that inputs that are close in the original space will also be mapped to neighboring neurons.
- Unlike traditional neural networks that require labeled data, SOMs work in an unsupervised manner, meaning they learn patterns in the data without explicit output labels.
- Unlike other ANN types, SOM doesn't have activation function in neurons, we directly pass weights to output layer without doing anything.
- SOM doesn't use backpropagation with SGD to update weights, this type of unsupervised artificial neural network uses competitive learning to update its weights.

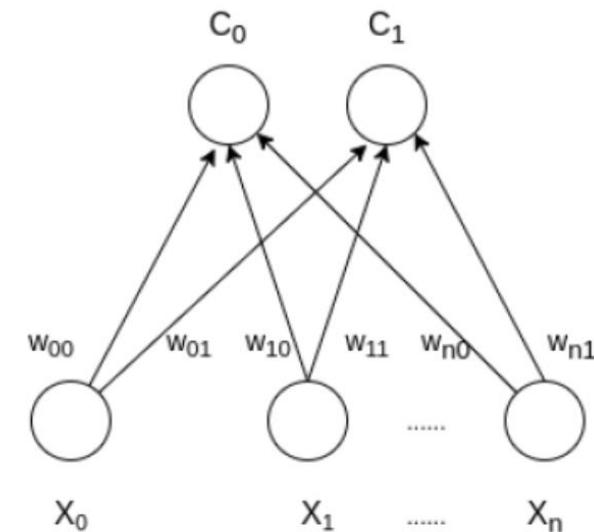
Architecture

- The architecture of a Self-Organizing Map (SOM) consists of two primary components:
 - Input Layer
 - Output Layer (the Map)
- Input layer : Consists of a set of neurons where each input neuron corresponds to a feature or dimension of the data being processed.
- Number of input neurons is equal to the number of features in the dataset (i.e., the dimensionality of the input data).
- The output layer is a grid of neurons, often organized in a 2D grid.
- This layer is the heart of the SOM architecture and is responsible for projecting the high-dimensional input data into a lower-dimensional space preserving the topological relationships of the input data.

Architecture (cont...)



Feature mapping/
Dimensionality reduction



Clustering

Training

- The weight vectors of the neurons are initialized randomly or based on some small range of values.
- An input vector from the dataset is selected.
- Identify the Best Matching Unit (BMU)(The winning node in the output layer). The BMU is the neuron whose weight vector is closest (according to a distance metric like Euclidean distance) to the input vector. $D(x_i, y_i) = \sum_{i=1}^m (x_i - y_i)^2$
- The weights of the BMU are adjusted to move closer to the input vector.
- Weights are updated using the formula $w_j(t+1) = w_j(t) + \eta(t)(x_s - w_j(t))$. Here η is the learning rate
- Repeat until feature map doesn't change.
- After training, the SOM organizes the input data in such a way that similar data points are mapped to nearby neurons, effectively creating a topological map.

Example

- Consider the following input data
 - Data1 = (0,0,1,1),
 - Data2 = (1,0,0,0),
 - Data3 = (0,1,1,0),
 - Data4 = (0,0,0,1)
- Cluster the data in to 2 clusters. Let the initial weights be as follows
 - [unit 1] = [0.2, 0.4, 0.6, 0.8]
 - [unit 2] = [0.9, 0.7, 0.5, 0.3]
- Let the learning rate be 0.5

Example (cont...)

- For input data (0,0,1,1)
- Calculate the Euclidian distance
- Distance between (0,0,1,1) and (0.2, 0.4, 0.6, 0.8) = 0.4
- Distance between (0,0,1,1) and (0.9,0.7, 0.5, 0.3) = 2.04
- Since Unit 1 wins. So update the weights of Unit 1 using the formula $w_j(t+1) = w_j + \eta(t)(x_s - w_j(t))$
 $[0.2, 0.4, 0.6, 0.8] = [0.2 + 0.5(0 - 0.2), 0.4 + 0.5(0 - 0.4), 0.6 + 0.5(1 - 0.6), 0.8 + 0.5(1 - 0.8)]$
 $= [0.1, 0.2, 0.8, 0.9]$
- The updated weights are:
 - [unit 1] = [0.1, 0.2, 0.8, 0.9]
 - [unit 2] = [0.9,0.7, 0.5, 0.3]

Example (cont...)

- For input data (1,0,0,0)
- Calculate the Euclidian distance
- Distance between (1,0,0,0) and (0.1, 0.2, 0.8, 0.9) = 2.3
- Distance between (1,0,0,0) and (0.9,0.7, 0.5, 0.3) = 0.84
- Since Unit 2 wins. So update the weights of Unit 2 using the formula $w_j(t+1) = w_j + \eta(t)(x_s - w_j(t))$
[0.9, 0.7, 0.5, 0.3] = [0.9 + 0.5(1 - 0.9), 0.7 + 0.5(0 - 0.7), 0.5 + 0.5(0 - 0.5) , 0.3 + 0.5(0 - 0.3)]
= [0.95, 0.35, 0.25, 0.15]
- The updated weights are:
 - [unit 1] = [0.1, 0.2, 0.8, 0.9]
 - [unit 2] = [0.95, 0.35, 0.25, 0.15]

Example (cont...)

- For input data (0,1,1,0,)
- Calculate the Euclidian distance
- Distance between (0,1,1,0,) and (0.1, 0.2, 0.8, 0.9) = 1.5
- Distance between (0,1,1,0,) and (0.95, 0.35, 0.25, 0.15) = 1.91
- Since Unit 1 wins. So update the weights of Unit 1 using the formula $w_j(t+1) = w_j + \eta(t)(x_s - w_j(t))$
 $[0.1, 0.2, 0.8, 0.9] = [0.1 + 0.5(0 - 0.1), 0.2 + 0.5(1 - 0.2), 0.8 + 0.5(1 - 0.8), 0.9 + 0.5(0 - 0.9)]$
 $= [0.05, 0.6, 0.9, 0.45]$
- The updated weights are:
 - [unit 1] = [0.05, 0.6, 0.9, 0.45]
 - [unit 2] = [0.95, 0.35, 0.25, 0.15]

Example (cont...)

- For input data (0,0,0,1)
- Calculate the Euclidian distance
- Distance between (0,0,0,1) and (0.05, 0.6, 0.9, 0.45) = 1.475
- Distance between (0,0,0,1) and (0.95, 0.35, 0.25, 0.15) = 1.81
- Since Unit 1 wins. So update the weights of Unit 1 using the formula $w_j(t+1) = w_j + \eta(t)(x_s - w_j(t))$
[0.05, 0.6, 0.9, 0.45] = [0.05 + 0.5(0 - 0.05), 0.6 + 0.5(0 - 0.6), 0.9 + 0.5(0 - 0.9), 0.45 + 0.5(1 - 0.45)]
= [0.025, 0.3, 0.45, 0.475]
- The updated weights are:
 - [unit 1] = [0.025, 0.3, 0.45, 0.475]
 - [unit 2] = [0.95, 0.35, 0.25, 0.15]
- Now, the learning rate is updated as $\eta(t+1) = 0.5 * \eta(t) = 0.5 * 0.5 = 0.25$
- The process is continued till the cluster assignment doesn't change or till threshold number of epochs is reached.