Self Organizing Maps

- Neural Networks use processing, inspired by the human brain, as a basis to develop algorithms that can be used to model and understand complex patterns and prediction problems.
- There are several types of neural networks and each has its own unique use.
- The Self Organizing Map (SOM) is one such variant of the neural network, also known as Kohonen's Map.

- A self-organizing map is also known as SOM and it was proposed by Kohonen.
- It is an unsupervised neural network that is trained using unsupervised learning techniques to produce a low dimensional, discretized representation from the input space of the training samples, known as a map and is, therefore, a method to reduce data dimensions.

- Self-Organizing Maps are very different from other artificial neural networks as they apply competitive learning techniques unlike others using error-correction learning methods such as backpropagation with gradient descent, and use a neighbourhood function to preserve all the topological properties within the input space.

- Self-Organizing Maps were initially only being used for data visualization, but these days, it has been applied to different problems, including as a solution to the Traveling Salesman Problem aswell.
- Map units or neurons usually form a two-dimensional space and hence a mapping from high dimensional space onto a plane is created.

- The map retains the calculated relative distance between the points. Points closer to each other within the input space are mapped to the nearby map units in Self-Organizing Maps.
- Self-Organizing Maps can thus serve as a cluster analyzing tool for high dimensional data.
- Self-Organizing Maps also have the capability to generalize.
- During generalization, the network can recognize or characterize inputs that it has never seen as data before.
- New input is taken up with the map unit and is therefore mapped.

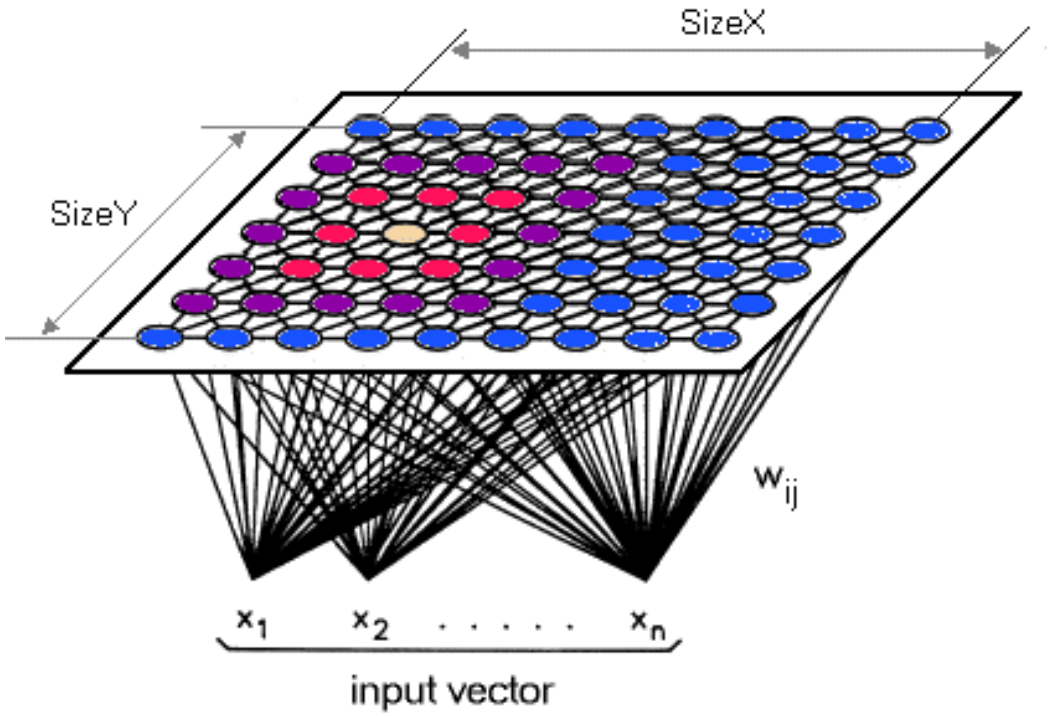
- Self-Organizing Maps provide an advantage in maintaining the structural information from the training data and are not inherently linear.
- Using Principal Component Analysis on high dimensional data may just cause loss of data when the dimension gets reduced into two.
- If the data comprises a lot of dimensions and if every dimension preset is useful, in such cases Self-Organizing Maps can be very useful over PCA for dimensionality reduction.

- Seismic facies analysis generates groups based on the identification of different individual features.
- This method finds feature organizations in the dataset and forms organized relational clusters.

- However, these clusters sometimes may or may not have any physical analogs.
- Therefore a calibration method to relate these clusters to reality is required and Self-Organizing Maps do the job.
- This calibration method defines the mapping between the groups and the measured physical properties.

- Text clustering is another important preprocessing step that can be performed through Self-Organizing Maps.
- It is a method that helps to verify how the present text can be converted into a mathematical expression for further analysis and processing.
- Exploratory data analysis and visualization are also the most important applications of Self-Organizing Maps.

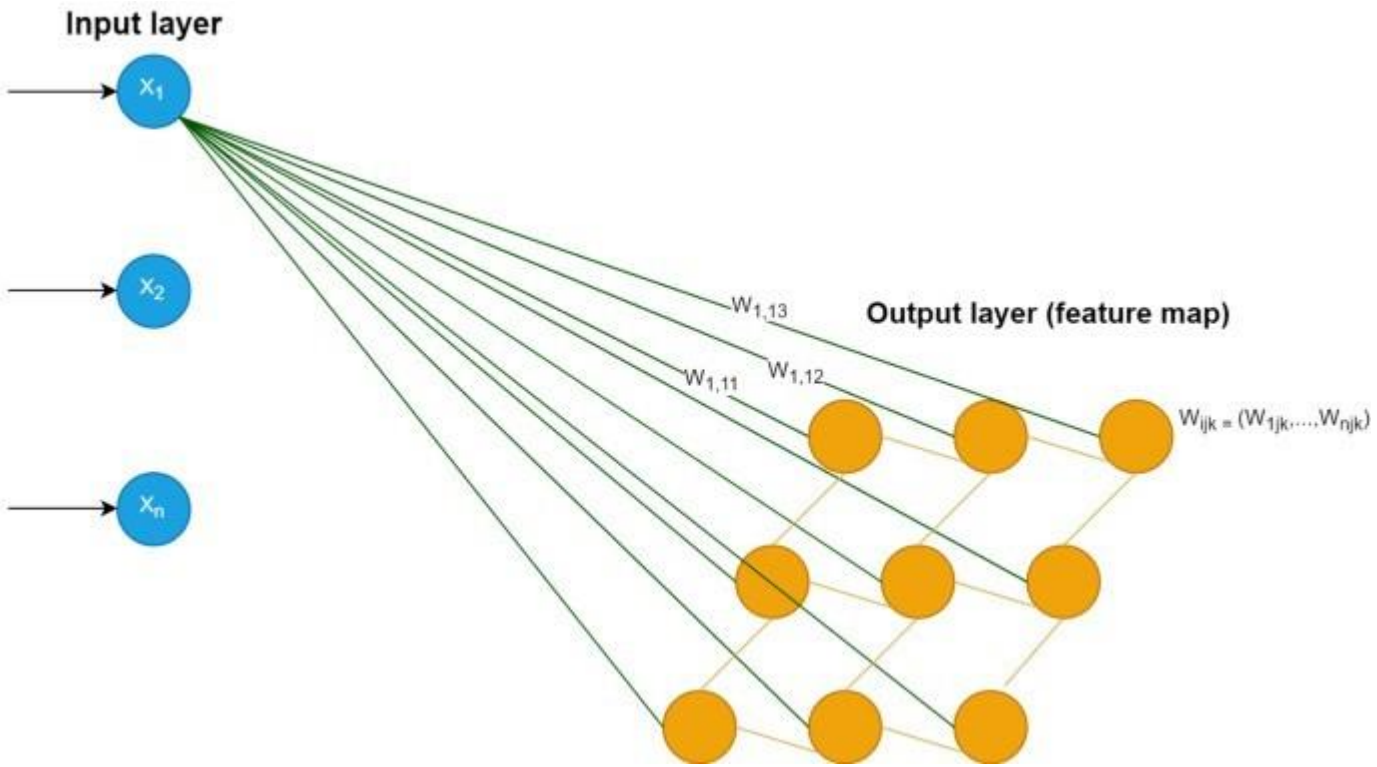
Uses of Self-Organizing Maps



- SOM doesn't use backpropagation with SGD to update weights, this type of unsupervised artificial neural network uses competetive learning to update its weights.
- Competitive learning is based on three processes :
  - Competition
  - Cooperation
  - Adaptation

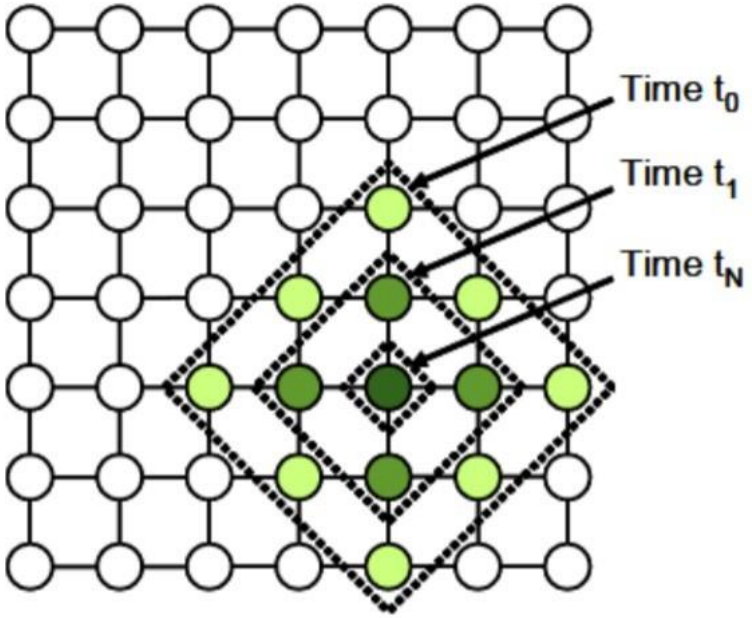
- As we said before each neuron in a SOM is assigned a weight vector with the same dimensionality as the input space.
- In the example below, in each neuron of the output layer we will have a vector with dimension n.
- We compute distance between each neuron (neuron from the output layer) and the input data, and the neuron with the lowest distance will be the winner of the competetion.
- The Euclidean metric is commonly used to measure distance.

Competition



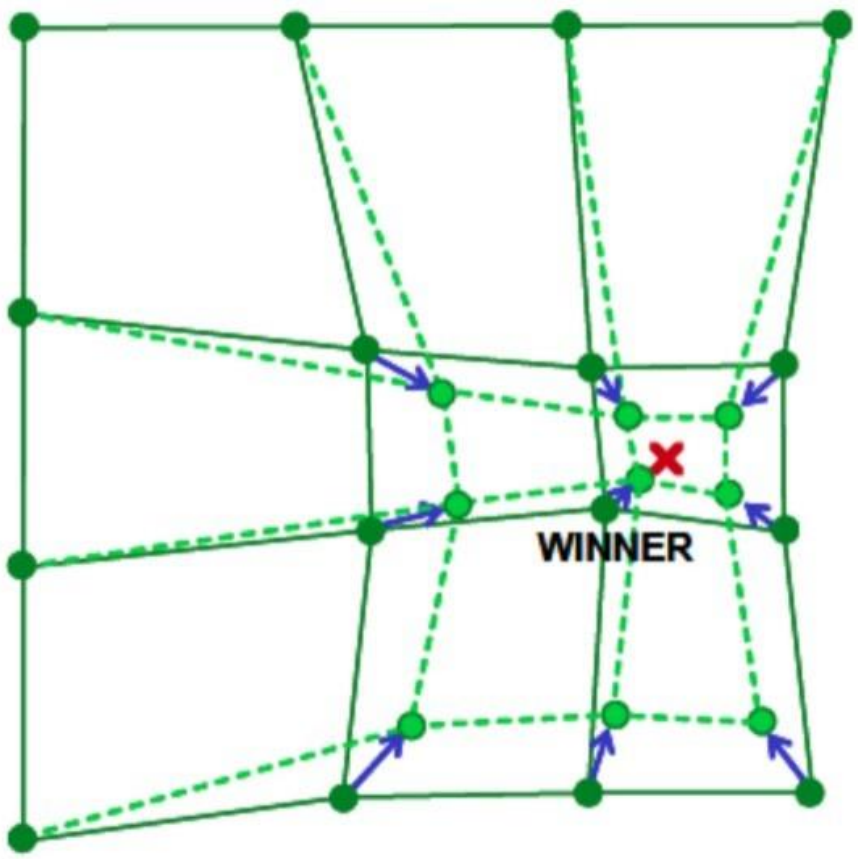
- We will update the vector of the winner neuron in the final process (adaptation) but it is not the only one, also it's neighbor will be updated.
- How do we choose the neighbors ?
- To choose neighbors we use neighborhood kernel function, this function depends on two factor : time (time incremented each new input data) and distance between the winner neuron and the other neuron (How far is the neuron from the winner neuron).

- The image below show us how the winner neuron's (The most green one in the center) neighbors are choosen depending on distance and time factors.



- After choosing the winner neuron and it’s neighbors we compute neurons update.
- Those choosen neurons will be updated but not the same update, more the distance between neuron and the input data grow less we adjust it like shown in the image below :

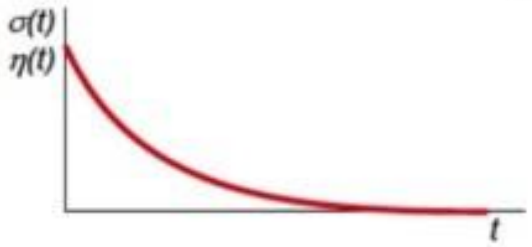
Adaptation



- The winner neuron and it's neighbors will be updated using this formula:

$$w_k = w_k + \eta(t) \cdot h_{ik}(t) \cdot (x^{(n} - w_k)$$

A learning rate decay rule $\eta(t) = \eta_0 \exp\left(-\frac{t}{\tau_1}\right)$

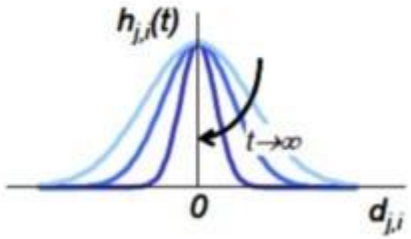


- This learning rate indicates how much we want to adjust our weights.
- After time t (positive infinite), this learning rate will converge to zero so we will have no update even for the neuron winner .

A neighborhood kernel function $h_{ik}(t) = \exp\left(-\frac{d_{ik}^2}{2\sigma^2(t)}\right)$

- where d_{ik} is the lattice distance between w_i and w_k

A neighborhood size decay rule $\sigma(t) = \sigma_0 \exp\left(-\frac{t}{\tau_2}\right)$



- The neighborhood kernel depends on the distance between winner neuron and the other neuron (they are proportionally reversed : d increase make h(t) decrease) and the neighborhood size wich itself depends on time (decrease while time incrementing) and this make neighborhood kernel function decrease also.

Full SOM Algorithm

1. Initialize weights to some small, random values
2. Repeat until convergence
   - 2a. Select the next input pattern $x^{(n}$ from the database
     - 2a1. Find the unit w_i that best matches the input pattern $x^{(n}$

$$i(x^{(n)}) = \arg\min_j \left\| x^{(n} - w_j \right\|$$

     - 2a2. Update the weights of the winner w_i and all its neighbors w_k

$$w_k = w_k + \eta(t) \cdot h_{ik}(t) \cdot (x^{(n} - w_k)$$

   - 2b. Decrease the learning rate $\eta(t)$
   - 2c. Decrease neighborhood size $\sigma(t)$

- Self-Organizing Maps consist of two important layers, the first one is the input layer, and the second one is the output layer, which is also known as a feature map.
- Each data point in the dataset recognizes itself by competing for a representation.
- The Self-Organizing Maps' mapping steps start from initializing the weight to vectors.

- After this, a random vector as the sample is selected and the mapped vectors are searched to find which weight best represents the chosen sample.
- Each weighted vector has neighboring weights present that are close to it. The chosen weight is then rewarded by being able to become a random sample vector.
- This helps the map to grow and form different shapes. Most generally, they form square or hexagonal shapes in a 2D feature space.
- This whole process is repeatedly performed a large number of times and more than 1000 times.

- Self-Organizing Maps do not use backpropagation with SGD to update weights, this unsupervised ANN uses competitive learning to update its weights i.e Competition, Cooperation and Adaptation.
- Each neuron of the output layer is present with a vector with dimension n.
- The distance between each neuron present at the output layer and the input data is computed.

- The neuron with the lowest distance is termed as the most suitable fit.
- Updating the vector of the suitable neuron in the final process is known as adaptation, along with its neighbour in cooperation.
- After selecting the suitable neuron and its neighbours, we process the neuron to update.
- The more the distance between the neuron and the input, the more the data grows.

Hope it Helps

- The suitable weight is further rewarded with transitioning into more like the sample vector. The neighbours transition like the sample vector chosen.
  - The closer a node is to the Best Matching Unit, the more its weights get altered and the farther away the neighbour is from the node, the less it learns.
- Repeat the second step for N iterations.

- Data can be easily interpreted and understood with the help of techniques like reduction of dimensionality and grid clustering.
- Self-Organizing Maps are capable of handling several types of classification problems while providing a useful, and intelligent summary from the data at the same time.

- It does not create a generative model for the data and therefore the model does not understand how data is being created.
- Self-Organizing Maps do not perform well while working with categorical data and even worse for mixed types of data.
- The model preparation time is comparatively very slow and hard to train against the slowly evolving data.

@mitu_skillologies

/mITuSkillologies

@mitu_group

/company/mitu-skillologies

MITUSkillologies

Web Resources
https://mitu.co.in
http://tusharkute.com

contact@mitu.co.in

tushar@tusharkute.com