Artificial neural networks: Unsupervised learning

Laboratorio Calcolo Matematico
Modulo Reti Neurali

Juan Rojo

INFN, Sezione di Milano

Milano, 20/01/2010

Unsupervised learning

- In previous lectures of the course we have seen always examples of **supervised learning**
- Neural networks can also be used to find patterns in data **without supervision**
- In unsupervised learning, a dataset is presented to the NN, who categorizes this input set into groups
- The mechanism behind unsupervised learning is **competition**
- As opposed to supervised learning, in the unsupervised case data is not organized in **input-output patterns**

The basic competitive neural network

- Competitive neural networks are two layer and fully connected, with connections usually inter-layer and not intra-layer
- As usual with Neural networks, input data is applied to the input layer and the outputs from the output layer nodes are considered
- In supervised learning we have a set of **"known outputs"** to which the net outputs can be compared to through the error function

$$E[\omega, \theta] \equiv \frac{1}{2} \sum_{A=1}^{n_p} \sum_{i=1}^{n_L} (o_i(\mathbf{x}^A) - z_i^A)^2$$

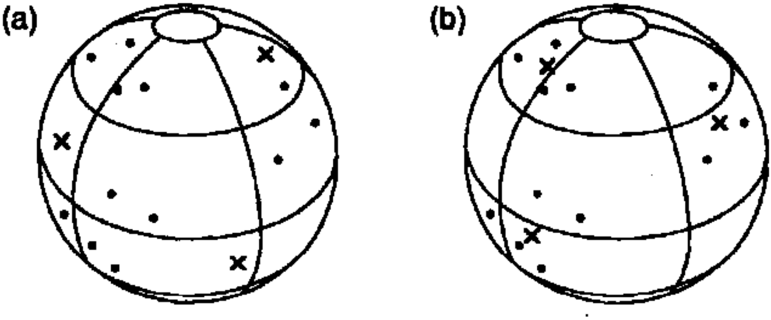
  but this is not available in unsupervised learning
- Here enters **competitions** $\rightarrow$ A node with its weight vector closest to the vector of inputs is declared the winner, and only its weights are adjusted by a **training algorithm**
- The process is repeated for each input vector for a large number of cycles

The basic competitive neural network

- Eventually, different nodes become associated with different groups of input vectors $\longrightarrow$ Nodes become **associated to patterns** in the input set
- The weight vector for a node represents the average of the data vectors for the particular pattern in the data set
- An important concept is how to measure "closeness" between input and weight vectors
- While the number of **input neurons** is fixed by the dimension of the data, the number of **output neurons** neurons is not known a priory: ideally one should have as many output neurons as groups in the input data

The basic competitive neural network - Example

Consider 12 input vectors (normalized to unity) normalized to unity



A 3-3 unsupervised ANN before **(a)** and after (**b**) the training with 12 input data

The basic competitive neural network

- In order to perform the training, let us normalize the input vector **A** and the weight vectors w_i, where i runs though all output nodes,

$$|\mathbf{A}| = 1 \; , \qquad |\mathbf{w_i}| = 1$$

- Then the closeness between the input vector and the weight vectors w_i is defined by the Euclidean distance,

$$D_i \equiv |\mathbf{w_i} - \mathbf{A}|$$

- The node with smaller distance D_i, D_{i*} is declared the winner, and only its weights are adjusted by the training algorithm, given by

$$\Delta w_{i*j} = \eta \left(A_j - w_{i*j} \right)$$

  with η the usual learning rate
- At the end of the unsupervised learning, different nodes in the network should become associated with different hidden patterns in the input data set
- A much more efficient algorithm for unsupervised learning is provided by **Kohonen Self Organizing Map**

Self Organized Maps (SOM)

- The Kohonen SOM not only categorizes the input data (as the competitive network does) but also recognizes which input patterns are close to each other.
- SOMs are different from other types of artificial neural networks in the sense that they use a neighborhood function to preserve the topological properties of the input space
- In SOMs, the weight vectors which are modified by the learning algorithm are not only those of the winning node or Best Matching Unit (BMU), but also those of their neighborhood.
- The learning rate is also chosen to decrease monotonically with training cycles

Self Organized Maps (SOM)

- The SOMs learning rule is

$$\Delta w_{ij}(t+1) = \eta(t) N\left(t, D_{i,i*}\right)\left(A_j - w_{ij}\right)$$

where the neighborhood function depends on the Euclidean distance between the BMU $i*$ and the node i

$$D_{i,i*} \equiv \left|\mathbf{w_i} - \mathbf{w}_{i*}\right|$$

It is clear that for $N\left(D_{i,i*}\right) = \delta_{i,i*}$ we recover the basic competitive neural network learning

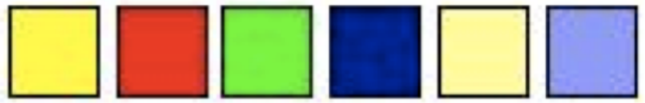
- One can use also a gaussian for $N\left(D_{i,i*}\right) \to$ The important thing is that it shrinks with time
- Self-Organized maps are not only useful at identifying patterns, they are also helpful for multi-dimensional data visualization

Self Organized Maps (SOM)

- The first step is to create an **array of maps**.
- In each position of the array we put an artificial neural network of the for $N_{\mathrm{input}} - 1$, where N_{input} is the dimension of the input data which is used for the learning.
- Each of these ANNs remembers their position in the array
- Initially all weights are initialized at random
- Then each of the data patterns is shown to each of the maps in the array, and the weights of the BMU (and their neighbours) are updated according to the SOMs algorithm.

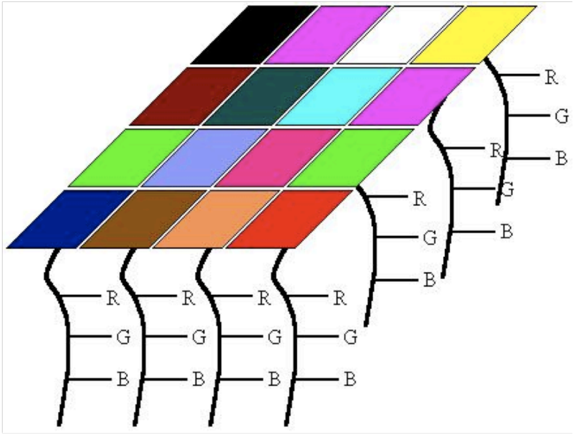
Self-Organized Maps: an example

The starting data set is composed by different colors:



We know the underlying pattern of this data set: each color has a different proportion of BRG basic colors $\longrightarrow$ SOMs should be able to stop these patterns and classify them

Self-Organized Maps: an example



Self-Organized Maps: an example

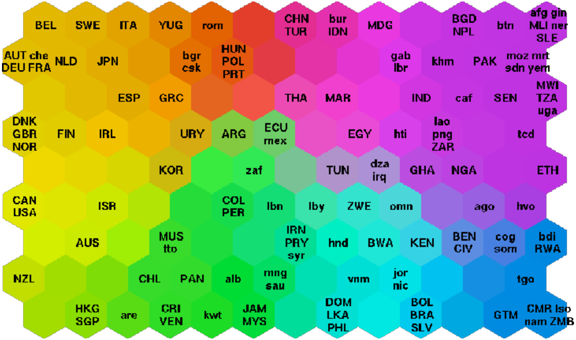
Self-Organized Maps: an example

Self-Organized Maps: Data visualization

- SOMs are also very useful to organize multi-dimensional data visualization
- Consider for example the *wealth* of a country as measured by 39 indicators $\rightarrow$ How to order the countries based in this?
- Generate a SOM and train it: the data set is the set of these 39 estimators for each country (ANN architecture: $39 - 1$)
- After the training, contries are clustered with those of a similar wealth

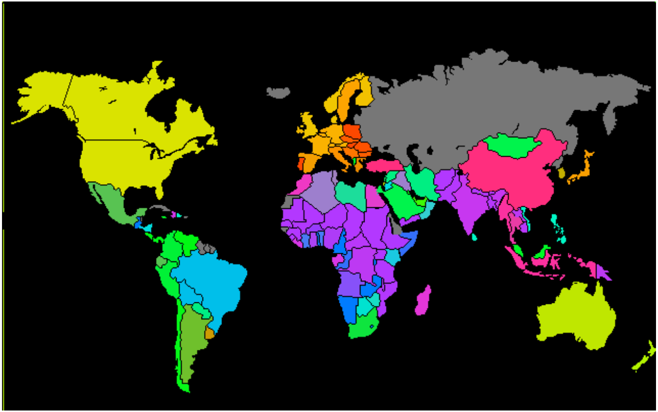
Self-Organized Maps: Data visualization

Some countries are assigned to the same map



In SOMs, also non-local similarties are mantained

Self-Organized Maps: Data visualization



Self-Organized Maps - Summary

Advantages:

- They are very easy to understand and to use them in an effective manner.
- They are very effective for data classification and it is easy to evaluate how strong the similarities between objects are.

Disadvantages:

- A limiting feature to the use of SOMs often referred to as missing data
- Every SOM is different and can find different similarities among the sample vectors of the input dataset
- SOMs organize sample data so that in the final product, the samples are usually surrounded by similar samples, however similar samples are not always near each other. If you have a lot of shades of purple, not always will you get one big group with all the purples
- So a lot of maps need to be constructed in order to get one final good map

SOMs - A Java example

Update the svn working copy and open a Java applet which implements Self-Organized Maps as follows:

appletviewer soms-av.html

Check of SOMs work:

- Sensitivity to initial conditions
- Dependence with the number of iterations
- Differences between SOM based on color patterns and those based on similarity patterns

SOMs - A Java example

The main program in `Screen.java`

- Find in the code where the computation of distances between weights and patterns is performed
- Determine which type of neighborhood function is being implemented
- Determine how the Best Matching Unit can be obtained