Artificial Neural Networks
Lecture Notes

Stephen Lucci, PhD

Part 13

About this file:

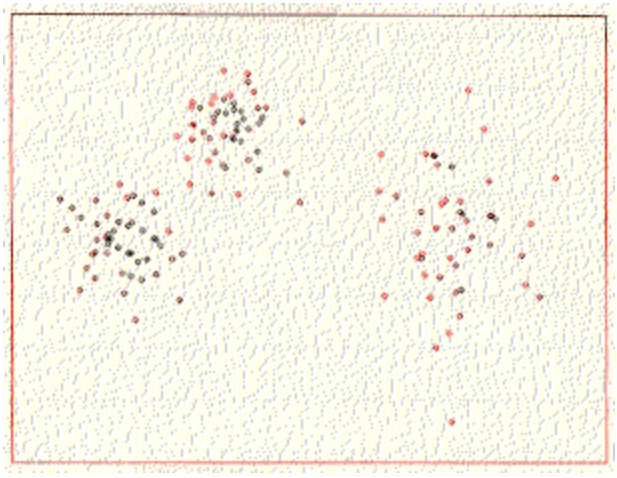
- This is the printer-friendly version of the file "*lecture13.htm*". In case the page is not properly displayed, use IE 5 or higher.
- Since this is an offline page, each file "lecture*nn*.htm" (for instance "lecture13.htm") is accompanied with a "img*nn*" folder (for instance "img13") containing the images which make part of the notes. So, to see the images, Each html file must be kept in the same directory (folder) as its corresponding "img*nn*" folder.
- If you have trouble reading the contents of this file, or in case of transcription errors, email gi0062@bcmail.brooklyn.cuny.edu
- Acknowledgments:
Background image is from http://www.anatomy.usyd.edu.au/online/neuroanatomy/tutorial1/tutorial1.html (edited) at the University of Sydney Neuroanatomy web page. Mathematics symbols images are from metamath.org's GIF images for Math Symbols web page. Other image credits are given where noted, the remainder are native to this file.

Contents

Self-Organization

Introduction

- A neural network discovers clusters of similar patterns in data without supervision (i.e., no target information is provided.)



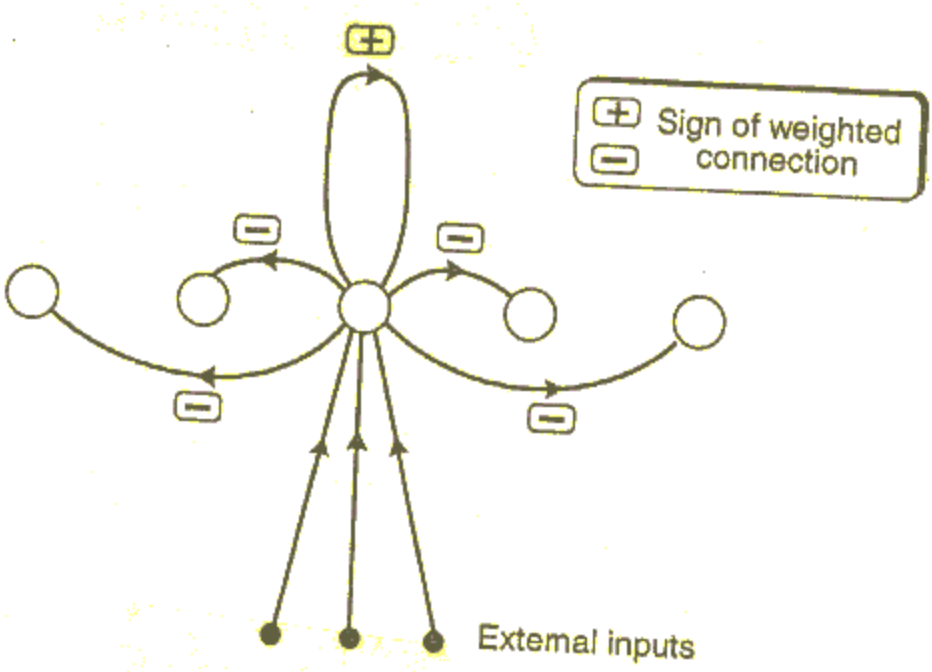
A Typical Dataset

- Self-organization and unsupervised learning -
  The network encodes such data by assigning nodes to clusters in some way.

- Network is supplied with extra resources to search for the location of largest activity.

Competitive Networks

- A network of units as shown below



- Each unit receives the same set of inputs from an external input layer.

- There are intra-layer or lateral connections such that each node j is connected to itself via an excitatory (positive) weight $v_j^{(+)}$

  and inhibits all other nodes in the layer with negative weights $v_{ij}^{(-)}$ - these are symmetric, i.e., $v_{ij}^{(-)} = v_{ji}^{(-)}$.

- This is said to be an *on-center, off-surround connection scheme*.

- Upon input $\bar{x}$, each unit computes its external input s = $\bar{w} \cdot \bar{x}$.

- One node, say K, will have maximal s.
  Activation a_k will increase, while other activations decrease.

- The total input input to each node j consists of:
  - The external input s_j, and
  - a contribution ℓ_j from other nodes in the layer.

- Node's output $y_i = \sigma(a_i)$ then l_i may be written thus

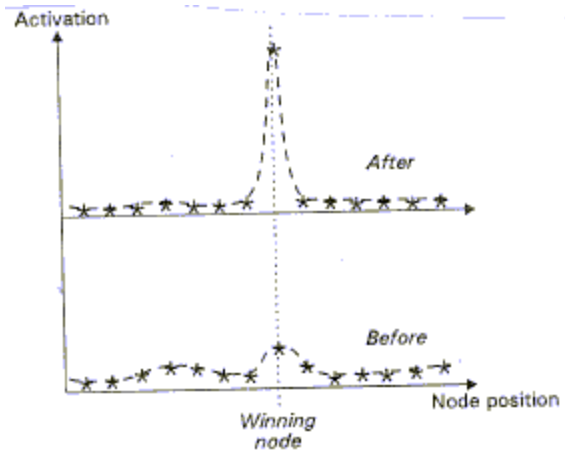
$$\ell_j = v_j^{(+)} y_j + \sum_{i \neq j} v_{ij}^{(-)} y_i$$

- Node K has its activation stimulated directly from the external input more strongly than any other node.

- This is reinforced via the self-excitatory connection.

- As output y_k grows, node k starts to inhibit the other nodes more than they can inhibit node k.

- The layer gradually reaches a point of equilibrium:



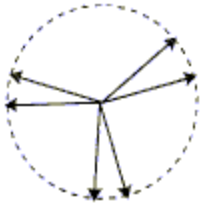
Winner Takes All

Requirements For There To Be A Unique Winning Node

- Only one excitatory weight emanating from any node that is connected to itself (vs. a small excitatory neighborhood.)
- Lateral connections sufficiently strong to ensure a well-defined equilibrium.
- The inhibitory connections from each node must extend to all other nodes.

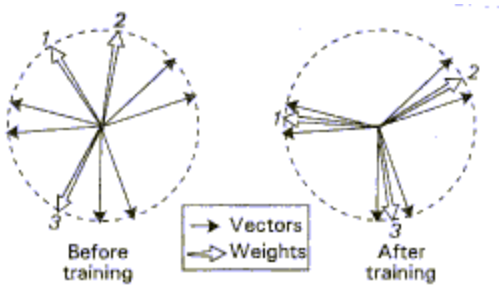
Competitive Learning

- Consider a training set whose vectors all have the same unit length, i.e., $||\bar{x}|| = 1$ for all $\bar{x}$.
- Normalization may be required:
  $\bar{x}' = (1 / ||\bar{x}||)\bar{x}$



In 2-D the vectors all fall on the unit circle.

- **Example**:
  - A competitive neural layer with a set of normalized weight vectors for connection to external input.



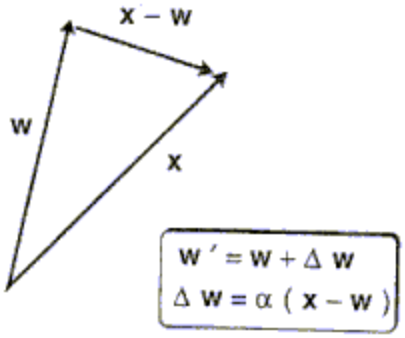
Normalized weights and vectors.

  - There are three nodes and three clusters - we would expect to be able to encode each cluster with a single node. Then, when a vector is presented to the net, there will be a single node that responds maximally to the input.
  - Note, s will be large and positive if the weight and input vectors are well-aligned.
    Recall,
    $s = ||\bar{w}|| \, ||\bar{x}|| \cos\theta$

- The ideal network will have its three weight vectors aligned with the three pattern clusters.
- This may be achieved by "rotating" each weight vector so that it becomes aligned with the cluster it is nearest to.

Training Methodology

- Iteratively apply vectors and adjust the weights of the node whose external input is largest.
- Let the "winning node" have index k and weight vector $\overline{w}_k$ (note $\overline{w}_k$ is the k^{th} weight vector, not the k^{th} component of $\overline{w}$ which is denoted w_k.)
- $\overline{w}_k$ should be rotated toward $\overline{x}$.
  Thus, we merely add a fraction of the difference vector $\overline{x}$ - $\overline{w}_k$



- **The Learing Rule**:

$$\Delta\overline{w}_j = \begin{cases} \alpha(\overline{x} - \overline{w}_j)\ ,\ j = k \\ 0\ ,\ j \neq k. \end{cases}$$

- If the network is "winner-takes-all", then node k will have its output close to one while others close to zero.
- After letting network reach equilibrium:
  $\Delta\overline{w}_j = \alpha(\overline{x} - \overline{w}_j)\ y$.

Training Algorithm

1. Apply a vector at the input to the network and evaluate s for each node.
2. Update the net until it reaches equilibrium where

$da_j / dt = c_1 (s_j + \ell_j - c_2 a_j$.

3. Train all nodes using:
$\Delta \overline{w}_j = \alpha(\overline{x} - \overline{w}_j)\, y$.

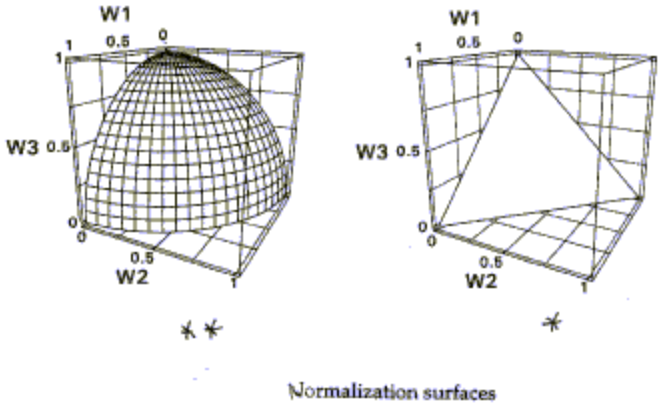
- Note: It is possible that if the patterns are not so well clustered and if there are many more patterns than nodes, then the weight coding may be unstable.

Problem with Normalization

- Weights are initially of unit length - as they adapt, their length will change. It would be impractical to renormalize after each training step.

- If all inputs are positive
$\sum_i w_i = 1$ *
is approximately equal to Euclidean length.
Since the Euclidean length is one, the squared length is also one
$\sum_i w_i^2 = 1$ **

** defines point on a unit sphere.
* defines point on a plane.



Normalization surfaces

- According to how closely the distance of the plane from the origin approximates to one, the two schemes may be said to be equivalent.

- Our learning rule may be rewritten as
$\Delta = \alpha \overline{x} y - \alpha \overline{w} y$.

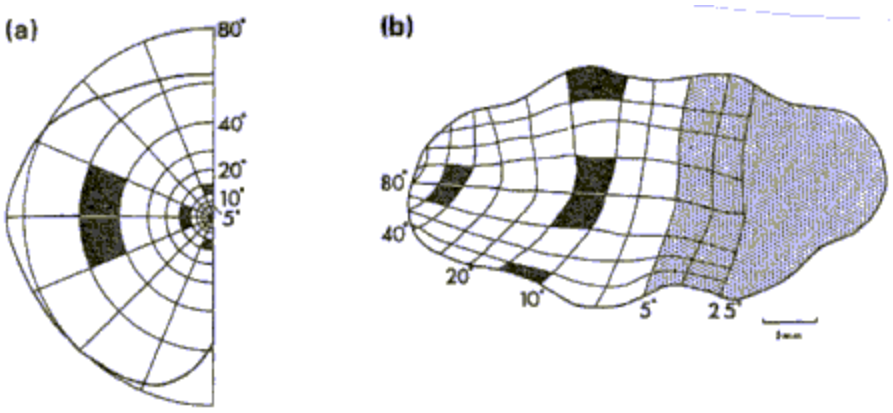
The first term looks like *Hebbian Learning*.
The second is the *weight decay*.
This latter property has a biological interpretation in terms of preservation of metabolic resources - the sum of synaptic strengths may not exceed a certain value.

Kohonen's Self-Organizing Feature Maps

- Competitive Networks:
  - Identify clusters in training data.
  - Also, nodes that are physically adjacent in the network encode patterns that are "adjacent" in some sense, in the pattern space of the input.

- The concept of proximity leads to the idea of a topography or map defined over a neural layer in which these maps represent some feature of the input space.

- The mammalian cortex is the externally visible sheet of neural tissue that is folded and wrapped to enclose more central areas of the brain.
  The cortex is responsible for processing sensory information, e.g., sound and vision.



Retinotopic Map for Macaque Monkey V1

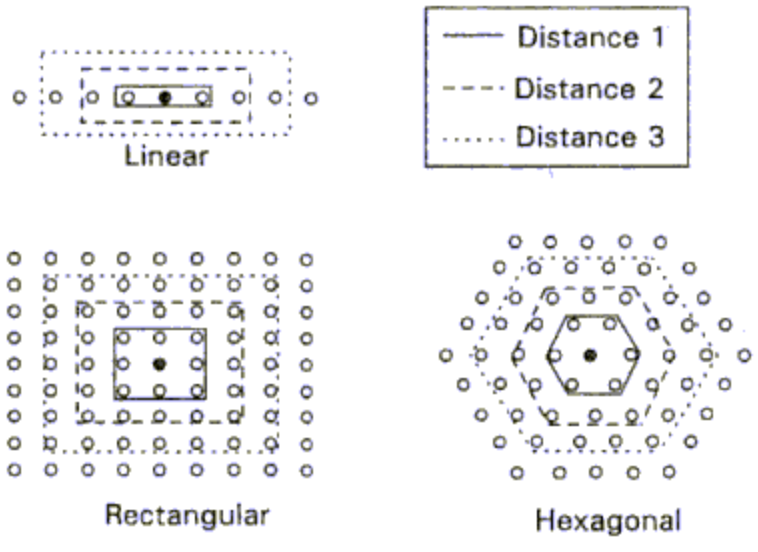
SOM Algorithm

- The network architecture consists of a set of inputs that are fully connected to the self-organizing layer, but now there are no lateral connections.

- The key principle for map formation is that training should take place over an extended region of the network centered on the maximally active node.

- Neighborhood Schemes:
  - Linear array of nodes
  - rectangular grid
  - hexagonal grid

- Three neighborhoods are shown delimited with respect to a shaded unit at distances of 1, 2 and 3 away from this node.



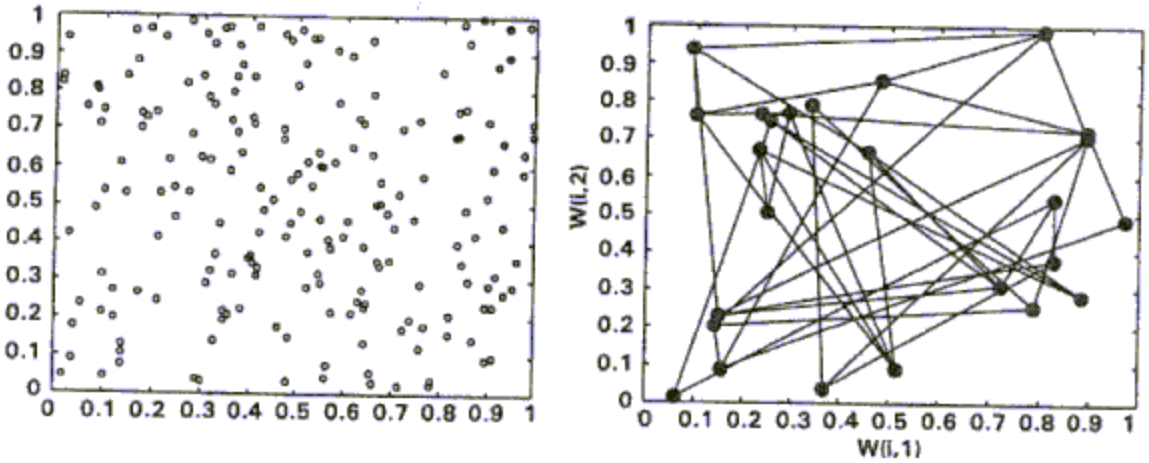
Neighbourhood schemes for SOMs.

- The linear, rectangular and hexagonal arrays have 5, 25 and 19 nodes respectively in their distance-2 neighborhoods (including the center node.)
- Weights are initialized to small random values.
- The neighborhood distance d_N is set to cover over half the network.
- Vectors are drawn randomly from the training set.
- The following operations performed at each selection:
  1. Find the best matching or "winning" node k whose weight vector $\overline{w}_k$ is closest to the current input vector $\overline{x}$ using the vector difference as criterion:
     $$||\overline{w}_k - \overline{x}|| = \min\{ ||\overline{w}_i - \overline{x}|| \}$$
  2. Train node k and all nodes in the neighborhood N_k of k

$$\Delta\overline{w}_j = \begin{cases} \alpha(\overline{x} - \overline{w}_j) \text{ , if j is in } N_k \\ 0 \text{ , if j is not in } N_k \end{cases}$$

  3. Decrease the learning rate α slightly.
  4. After a certain number M of cycles, decrease the size of the neighborhood d_N.
- An Example:

  200 vectors chosen at random from the unit square in pattern space and used to train a net of 25 nodes on a 5 * 5 rectangular grid -



A 25-node net: training set and initial weight space.