Slide 0

Lecture 4.1

Supervised Learning I: Competitive Learning

Jonathan Shapiro

Department of Computer Science, University of Manchester

February 5, 2003

Slide 1

What is unsupervised learning?

- Learning without any feedback about performance from the environment
- Learner receives input stimuli, responds to the stimuli, reinforces the responses (the learning), repeatedly.
- Learns a *self-organized* representation of the world.

Slide 2

How can unsupervised learning work?

Unsupervised learning can only work if there is something in the data to be discovered.

- Looks for features, redundancies, structure, etc. in the data.
- Particular unsupervised learning models have propensities for particular types of structures.
- If the data is uniform or unstructured, or contains the wrong type of structure, the system may produce nonsense.

Slide 3

Why is unsupervised learning important?

1. More appropriate model for some aspects of biological learning. Help explain how organisms (may) discover the primitives of their sensory world.
   - orientation selective cells in visual cortex
   - phoneme sensitization in infants
   - face selective cells in visual cortex in monkeys
   - place cells in rats

Slide 4

2. Useful for similar reasons in artificial systems — as pre-processors for supervised learning systems.
3. Useful for data visualization — help in finding what features exist in high-dimensional data.

Slide 5

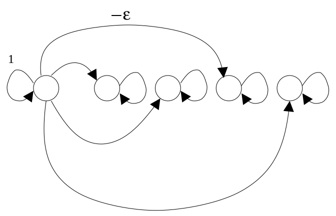
Competitive learning

References: Rosenblatt 1962, von der Marlsburg 1973, Fukushima 1975, Grossberg 1976, Kohonen 1982, Rumelhart and Zipser 1985

This is the simplest and best-know example of unsupervised learning in neural networks.

Slide 6

Competitive filters



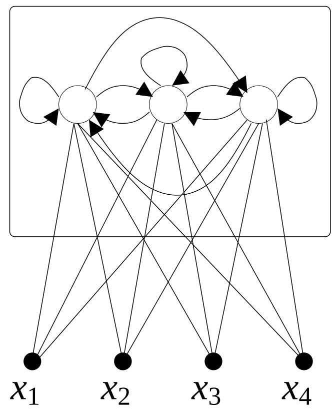


Figure 1: A competitive filter. Each neuron inhibits the activity of the other neurons in the layer, and excites itself.

Slide 7

- Figure 2 shows a competitive filter network.
- Layer of continuous activation neurons, with “on-center, off surround” pattern of connectivity.
- Each node receives external input
- Each node receives small inhibitory input from all other nodes (strength $-\varepsilon$).
- Each node receives excitory input from itself (strength 1).
- $x_i(t) = f\left[x_i(t) - \varepsilon \sum_{j \neq i} x_j(t)\right]$

Result: Node with largest external input fires at a high rate, the other nodes are suppressed.

Winner Takes All!

Slide 8

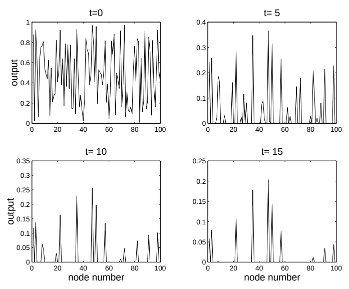


Figure 2: Simulation of the competitive filter. There are 100 neurons. The graphs show the neuron activity against the neuron number for different times t. The network picks out the node with largest activity. Here `satlin` threshold function is used, in each timestep every node is updated once in a random order, and $\varepsilon = 0.01$.

Slide 9

Winner-takes-all networks

- We abstract the above behaviour into a layer of neurons connected to a set of inputs.
- For any input pattern, only one node "fires", the winner.
  - The winner has an output of 1; all other nodes have output 0.
- Which node wins (various ways of doing this):
  - The node with the largest weight-sum-of-inputs, $\sum_j w_{ij} x_j$.
  - The node whose *weights are most like the input pattern*. I.e. that with smallest distance between the pattern and the weights

$$\boxed{d_j = \sum_i (x_i - w_{ij})^2}$$

Slide 10

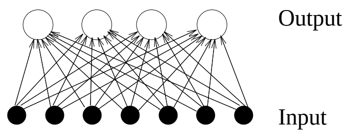


Figure 3: A simple competitive learning network. The outputs are a layer of winner-takes-all neurons. These are connected directly to the inputs by connections with weights

Slide 11

What can winner-takes-all networks compute?

These networks are a multiple-output analogues of perceptrons. A winner-takes-all network with N nodes will classify the input patterns into N classes with a *one-of-N* representation. These classes are seperated by hyperplanes.

Slide 12

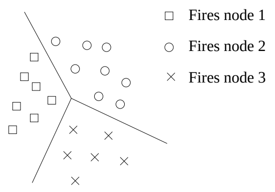


Figure 4: The decision boundary for a winner-takes-all network. In this there are three neurons. The squares, circles, and x's represent patterns which cause the first, second, and third neuron to fire respectively. The decision boundary is a combination of lines, these would be hyper-planes in higher dimensions.

Slide 13

Simple competitive learning

- An unsupervised learning algorithm for winner-takes-all networks
- Initialize the weights (e.g. small random values)
- For each pattern, *only the weights going into the winning node are updated*.
- Update makes the weights of the winning node closer to the current input pattern (so more likely to fire for that pattern).

Slide 14

- Repeat forever,
  - Apply a pattern x_j to the network
  - Find the winning node i^*.
  - Update weights of winning node (only) according to the rule,

$$\boxed{w_{i^* j} = w_{i^* j} + s\left[x_j - w_{i^* j}\right].}$$

The stepsize, s, is, as usual, small and arbitrary.

Slide 15

What does the competitive learning network learn?

- Learns to find *clusters* in data
- All patterns which cause the same node to fire are clustered together
- The weights of that node are the typical pattern in the cluster.

Slide 16

How well does competitive learning perform?

- For batch learning, convergence can be proven, although there will be many possible solutions, in general.
- For incremental (on-line) learning, convergence and stability are not insured.
- **Dead Units** — Nodes which never win a competition cannot learn. (Less likely with more sophisicated algorithms.)
- There is no guarantee that the number of clusters found by the network will match the number in the data in any sense.
- Similar to k-means clustering.

Slide 17

Examples of Competitive Learning

Example 1 - A two-dimensional data set —The first example consists of a set of 960 two-dimensional patterns of real numbers. For example, the first three patterns are:

```
1.3335399e-01    2.4407123e-01   -1.4771282e-01
1.0001405e+00    1.0014015e+00    9.8681831e-01
```

Slide 18

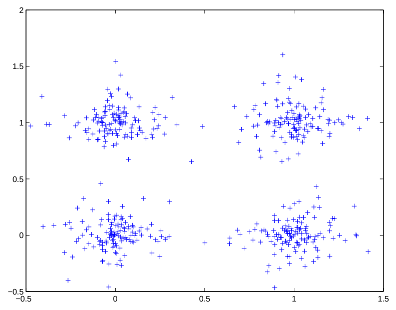


Figure 5: A set of clustered data in two-dimensions. The x-axis is the first component of the pattern, the y-axis is the second.

Slide 19

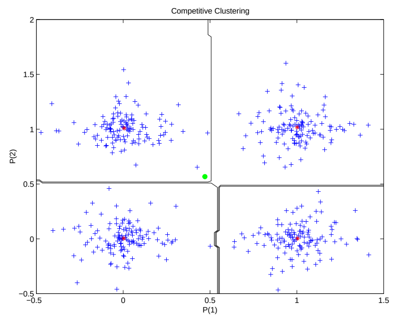


Figure 6: A competitive network learns to find the clusters in the data. The datapoints are in blue; the initial weights are in gree; the final weights are in red. You can see that the weights are at the centers of the clusters. The line is the decision boundary; any pattern in an area bounded by the lines will cause the same neuron to fire.

Slide 20

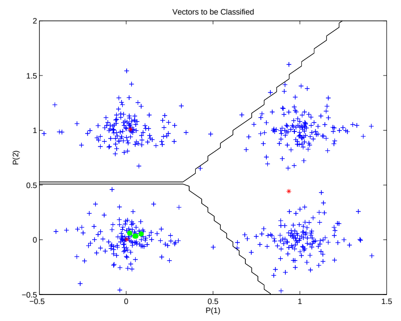


Figure 7: This figure shows a less successful attempt at finding clusters in the data. All the initial weights are near the lower left-hand cluster. Consequently, the network has two neurons which have learned that cluster, while one neuron covers the two clusters on the right-hand side of the figure.

Slide 21

Example 2 - Visual Edges This next example consists of high-dimensional data. The patterns are 256 bit strings of 1's and 0's. They represent 16 x 16 bit images which are shown in figure 8. They are edges of different orientation.

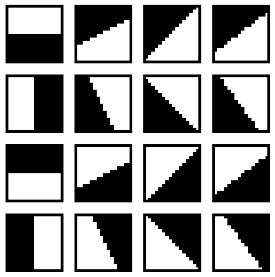


Figure 8: A set of 16 patterns of 256 1's and 0's arranged in a 16 x 16 grid. In the figures, a black square represents a 0 and a white square represents a 1.

Slide 22

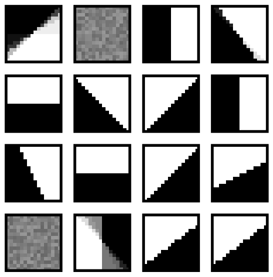


Figure 9: The weights of the 16 neurons in the competitive network after learning. There are 256 weights. These are shown as a 16 x 16 grid to show their relationship with the input. The weights are shown as a gray scale, with black representing a weight of 0, white representing a weight of 1, and gradations of gray representing the numbers in between. There are two dead units (the uniformly gray squares). A few of the neurons learned mixtures of patterns, but most of the neurons learned a single pattern.

Slide 23

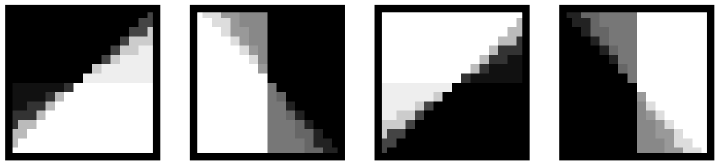


Figure 10: Same as figure 9 for a 4 neuron competitive network. Each neuron has learned to respond to a range of angles.

Slide 24

K-means clustering

This is a statistical approach for clustering data. The goal:

- partition the data into K subsets S_i, $i = 1, \dots, K$,
- with the mean of the data in the i subset as $\mathbf{m_i}$
- which minimizes $E = \sum_i^K \sum_{n \in S_i} ||x^n - m_i||^2$.

Competitive learning implements K-means clustering, with the weights as the means.

Slide 25

How good a clustering? Davies-Bouldin index — a heuristic measure of the quality of the clustering

$$I_{db} = \frac{1}{C} \sum_{i=1}^{C} \max j \neq i \left[\frac{\Delta(x_i) + \delta(x_j)}{\delta(x_i, x_j)} \right]$$

where

- C is the number of clusters;
- $\Delta(x_i)$ is the mean-squared distance from points in cluster i to the center;
- $\delta(x_i, x_j)$ is the distance between the centers of cluster i and cluster j.

A good clustering has a small value of this index.

Slide 26

Using competitive learning network to preprocess data for a supervised learning system

The architecture:

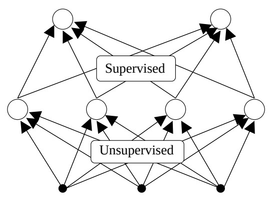


Figure 11: Architecture for using unsupervised clustering to preprocess data for classification or other supervised learning tasks. The unsupervised learning weights are trained first. Then, with these weights held fixed, the other weights are trained. Thus, the supervised learning task is to classify the clusters.

Slide 27

Advantages:

- Computational speed — unsupervised algorithms are much faster than backpropagation, and learning time grows with the number of layers for a supervised learning system.
- Improved generalization performance *if* the the data is clustered, and that clustering is reflected in the input-output mapping.

Disadvantages:

- Not general (unlike backpropagation). Only works if the data is clustered and patterns in a cluster have the same output mapping.
- Pattern within a cluster have the same internal representation, so the same output. This is fixed by using radial basis networks (see below).

Slide 28

Radial Basis Function Networks

- A modification of winner-takes-all networks
- Instead of an output of one 1 and all the rest 0's, the network interpolates between those outputs.
  - If the pattern is in the center of a cluster, the output of the node associated with that cluster is nearly 1, and the others are small.
  - If the pattern is between two clusters, the output of the nodes associated with those clusters would be near 0.5 and the rest small.
  - etc.

Slide 29

- How this is done mathematically, is to set the output of a hidden unit j to be

$$h_i = \frac{f\left[\sum_j (w_{ij} - x_j)^2\right]}{\sum_j f\left[\sum_j (w_{ij} - x_j)^2\right]}.$$

Here, $f(y)$ is a function which is peaked at zero, for example $f(y) = \exp(-y)$.

Slide 30

Application — Time-series prediction

Moody and Darken (1989) looked at a time-series prediction problem previously studied by Lapedes and Faber (1988) using an MLP with two hidden layers.

Results:

- Several orders of magnitude improvement in computation time
- Small degradation in performance