Introduction to Artificial Neural Networks

Lecture 9:

competitive learning

By: Ali Motie Nasrabadi

Outline

- **Biological background**
- **Competitive learning**
- **Simple competitive learning algorithm**
- **MATLAB Toolbox**

Biological background

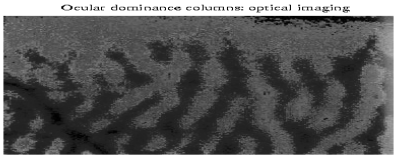
Neurons are wired topographically, nearby neurons connect to nearby neurons. In visual cortex, neurons are organized in functional columns.

Ocular dominance columns: one region responds to one eye input

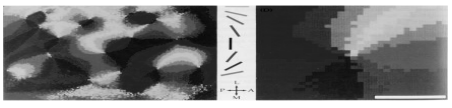
Orientation columns: one region responds to one direction

Self-organization as a principle of neural development

Ocular dominance columns: optical imaging



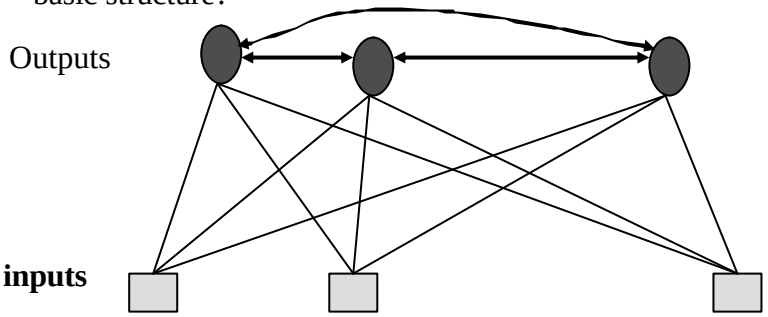
Orientation columns: optical imaging



15

Competitive learning

-finite resources: outputs 'compete' to see which will win via inhibitory connections between them
-aim is to automatically discover statistically salient features of pattern vectors in training data set: feature detectors
-can find clusters in training data pattern space which can be used to classify new patterns
-basic structure:



Lecture 9-5

input layer fully connected to output layer

input to output layer connection feedforward

output layer compares activation's of units following presentation of pattern vector $\underline{x}$ via (sometimes virtual) inhibitory lateral connections

winner selected based on largest activation
winner- takes-all (WTA)

linear or binary activation functions of output units.

Very different from previous (supervised) learning where we pay our attention to input-output relationship, here we will look at the pattern of connections (weights)

Lecture 9-6

More formally, if we have a d dimensional input vector $\underline{x}$ and M outputs y_i we have :

$$y_i = \sum_{j=1}^{d} w_{ji} x_j$$

where :

$$\|\underline{w}_i\| = \sum_{j=1}^{d} w_{ji}^{\ 2} = 1$$

for all i = 1, ..., M. We say that neuron i* is the winner if :

$$y_{i^*} > y_i \quad \textit{for all } i, i \neq i^*$$

and could make this a binary function by defining :

$$v_j = \begin{cases} 1 \textit{ if } & y_j > y_i \quad \textit{for all } i, i \neq j \\ & 0 \quad \textit{else} \end{cases}$$

Lecture 9-7

Simple competitive learning algorithm

Initialise all weights to random values and normalise (so that $\| \underline{w} \| = 1$)

loop until stopping criteria satisfied .

choose pattern vector x from training set
Compute distance between pattern and weight vectors
$\| \underline{X}_i - \underline{W} \|$

find output unit with largest activation ie 'winner'
i* with the property that
$\| \underline{X}_{i^*} - \underline{W} \| < \| \underline{X}_i - \underline{W} \|$

update the weight vector of winning unit only with
$\underline{W}(t+1) = \underline{W}(t) + \eta(t) (\underline{X}_i - \underline{W}(t))$

end loop

Lecture 9-8

NB choosing the largest output is the same as choosing the vector $\underline{w}$ that is nearest to $\underline{x}$ since:

a) $\underline{w}.\underline{x} = \underline{w}^T\underline{x} = ||\underline{w}||\ ||\underline{x}||$ cos(angle between x and w)

$= ||w||\ ||x||$ if the angle is 0

b) $||\underline{w} - \underline{x}||^2 = (w1-x1)^2 + (w2-x2)^2$

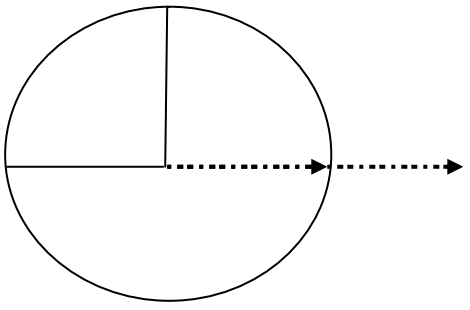
$= w1^2 + w2^2 + x1^2 + x2^2 - 2(x1w1 + x2w2)$

$= ||\underline{w}||^2 + ||\underline{x}||^2 - 2\ \underline{w}^T\underline{x}$

Since $||\underline{w}|| = 1$ and $||\underline{x}||$ is fixed, minimising $||\underline{w} - \underline{x}||^2$ is equivalent to maximising $\underline{w}^T\underline{x}$

Therefore, as we only really want the angle, WLOG only consider inputs with $||\underline{x}|| = 1$

Lecture 9-9



EG $h = 0.5$

W1 = (-1, 0)

W2 = (0,1)

X = (1, 0)

y1= -1, y2 = 0 so y2 wins

W2-> (0.5, 0.5)

y1= -1, y2 = 0.5 so y2 wins

W2-> (0.75, 0.25)

y1= -1, y2 = 0.75 so y2 wins

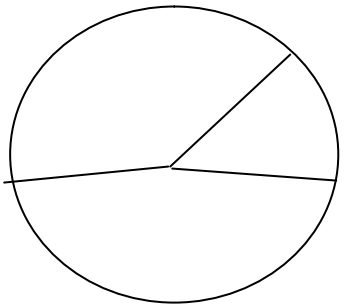
W2-> (0.875, 0.125)

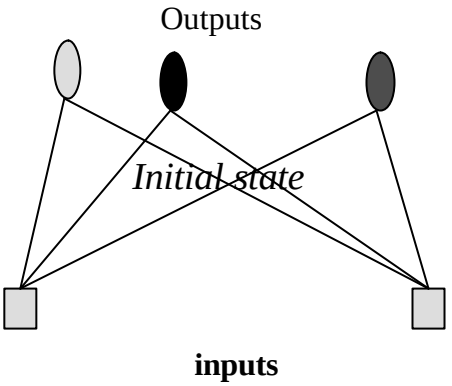
Etc etc

Lecture 9-10

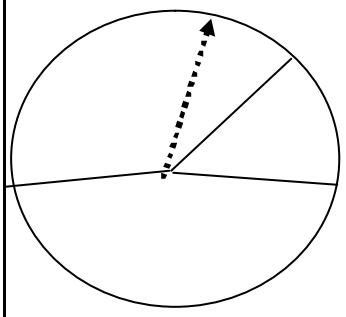
How does competitive learning work?
can view the points of all input vectors as in contact with surface of hypersphere (in 2D: a circle)

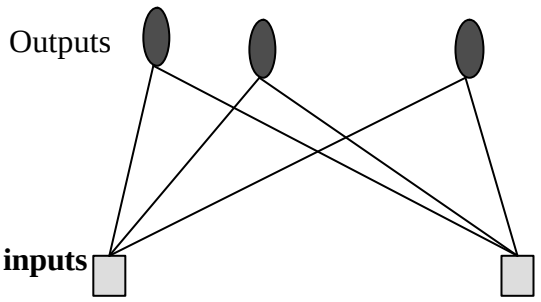
distance between points on surface of hypersphere = degree of similarity between patterns





Lecture 9-11





with respect to the incoming signal, the weight of the yellow line is updated so that the vector is rotated towards the incoming signal

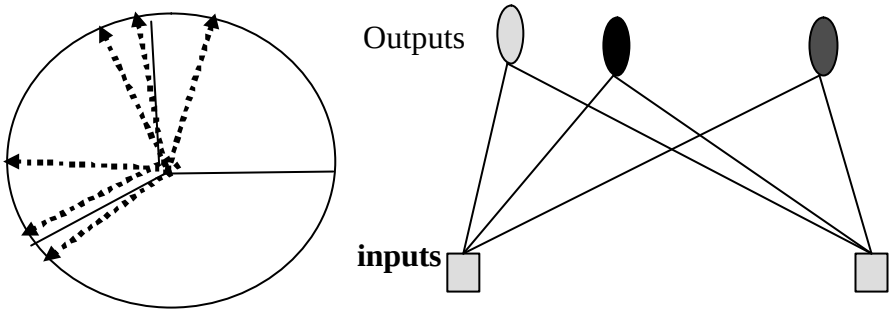
$$\underline{W}(t+1) = \underline{W}(t) + h\,(t)\,(\underline{X}_i - \underline{W}(t))$$

Thus the weight vector becomes more and more similar to the input ie a feature detector for that input

Lecture 9-12

When there are more input patterns, can see each weight vector migrates from initial position (determined randomly) to centre of gravity of a cluster of the input vectors.

Thus: Discover Clusters



Eg on-line k-means. Nearest centre updated by:

$$\underline{\text{centre}}_i(n+1) = \underline{\text{centre}}_i + \eta(t)\,(\underline{X}_i - \underline{\text{centre}}_i(t))$$

Lecture 9-13

Error minimization viewpoint: Consider error minimization across all patterns N in training set. Aim is to decrease error E

$$E = \Sigma\ ||\underline{X}_i - \underline{W}(t)||^2$$

For winning unit k when pattern is X_i the direction the weights need to change in a direction (so as to perform gradient descent) determined by (from previous lectures):

$$\underline{W}(t+1) = \underline{W}(t) + \eta(t)\,(\underline{X}_i - \underline{W}(t))$$

Which is the update rule for supervised learning (remembering that in supervised learning, W is O, the output of the neurons)

ie replace W by O and we recover the adaline/simple gradient descent learning rule

Lecture 9-14

Enforcing fairer competition

Initial position of weight vector of an output unit may be in region with few, if any, patterns (cf problems of k-means)

Many never or rarely become a winner and so weight vector may not be updated preventing it finding richer part of pattern space
DEAD UNIT

Or, initial position of a weight vector may be close to a large number of patterns while most other unit's weights are more distant
CONTINUAL WINNER, but weights will change little over time and prevent other units competing

More efficient to ensure a fairer competition where each unit has an equal chance of representing some part of training data

Leaky learning

modify weights of both winning and losing units but at different learning rates

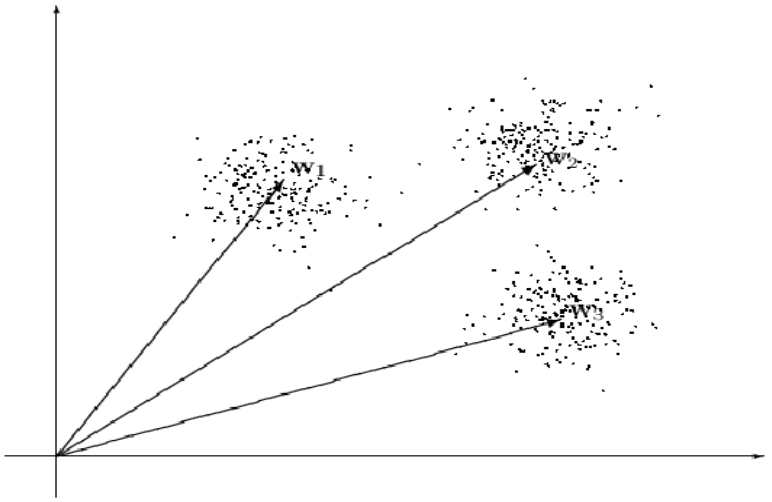
$$\underline{w}(t+1) = \underline{w}(t) + \begin{cases} h_w(\underline{x} - \underline{w}(t)) \\ h_L(\underline{x} - \underline{w}(t)) \end{cases}$$

where $h_w(t) >> h_L(t)$

has the effect of slowly moving losing units towards denser regions pattern space.

Many other ways as we will discuss later on

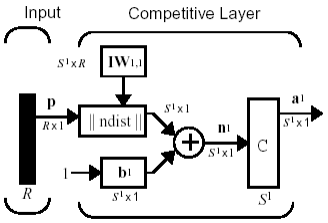
A 2-D Pattern with three clusters of data



MATLAB Toolbox

- **net = newc(PR,S,KLR,CLR)**
  - **Description of function**
    - **Competitive layers are used to solve classification problems**
  - **NET = NEWC(PR,S,KLR,CLR) takes these inputs,**
    - **PR - Rx2 matrix of min and max values for R input elements.**
    - **S - Number of neurons.**
    - **KLR - Kohonen learning rate, default = 0.01.**
    - **CLR - Conscience learning rate, default = 0.001.**
    - **Returns a new competitive layer.**

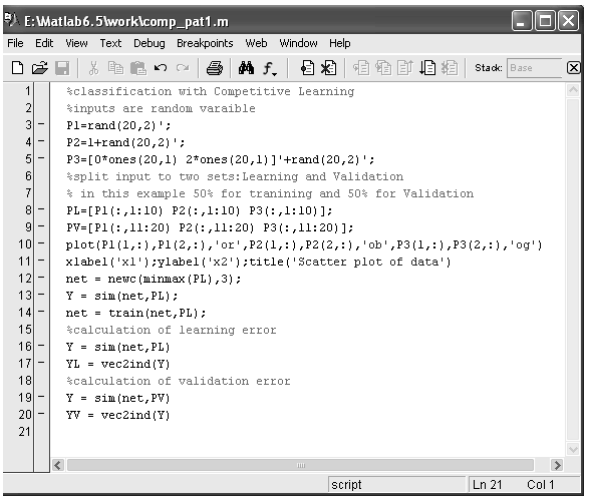
- **The architecture for a competitive network is shown below**



Lecture 9-19

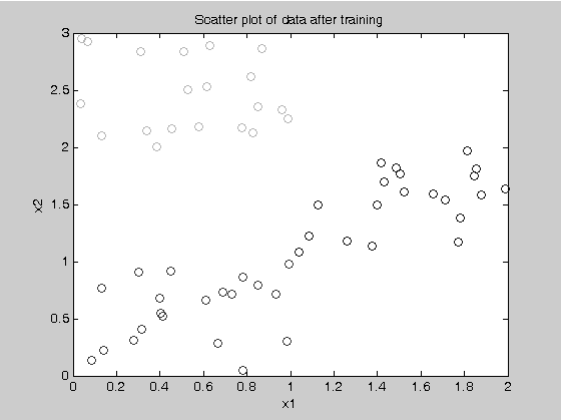
Example of classification with Competitive Network

- **A 2-D pattern with 3 classes**
- **See the M_file**

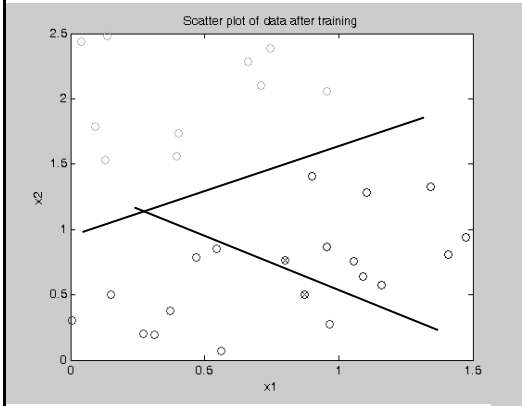


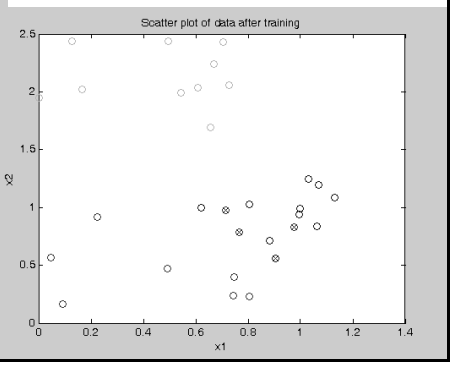
Lecture 9-20

■ **Three classes without overlapping**

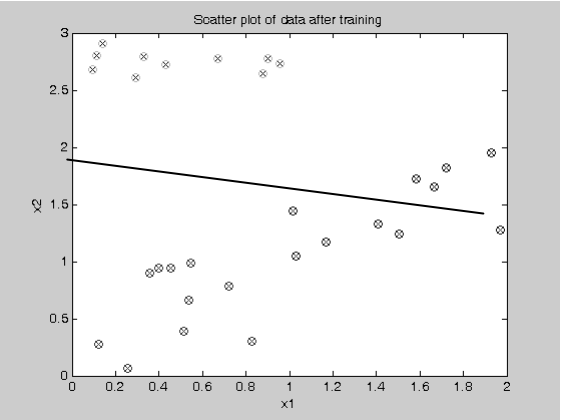


■ **Three classes with overlapping**



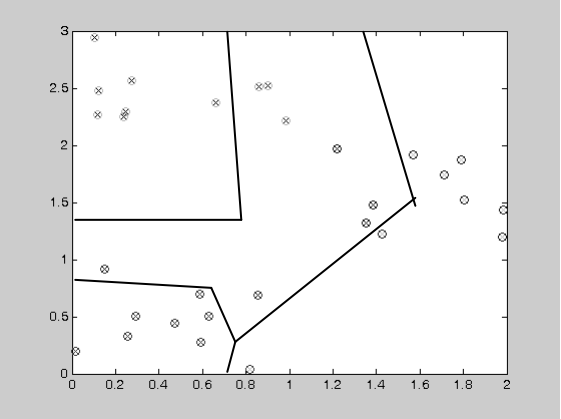


■ **Three classes are classified in two classes**



Lecture 9-23

■ **Three classes are classified as four classes**



Lecture 9-24