Competitive learning

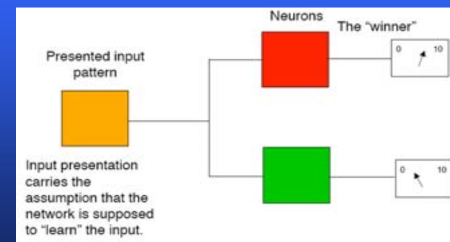
Topic 8

Note: lecture notes by Bob Keller (Harvey Mudd College, CA) are used

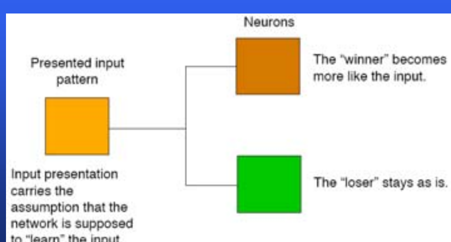
Main idea: combine unsupervised and supervised learning

- **Supervised learning**: training using desired response for given stimuli (“rote” learning)
- **Unsupervised learning**: classification by “clustering” of stimuli, without specified response
- **Hybrid**: e.g. unsupervised to form cluster, supervised to learn desired response to class

Two – way competition

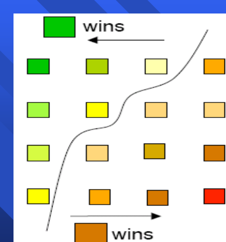


Two – way competition



Why not make the winner *exactly* like the input?

- There may be many more distinct input patterns than neurons.
- By “averaging” its behavior, a neuron can put a large number of distinct, but similar inputs into the same category.



An application example

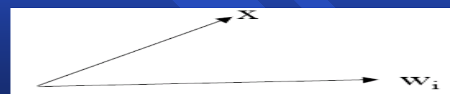
- Display an image file with “millions of colors” on a graphic display with, say, 256 colors.
- Each color in the image has to be mapped
- Map each image color into the closest one of the 256.
- The actual choice of the 256 might not be fixed; it is likely a limitation of some hardware table (of RGB values) rather than a limitation of the screen itself.
- In this case, a competitive network can **learn** a reasonable set of colors to use for a given image.

Measures of similarity or closeness (opposite: distance)

- Suppose x is an input vector and wi the weight vector of the ith neuron.
- One measure of distance is the **Euclidean distance**: $|| x - w_i || = sqrt(sumi((x_j - w_{ij})^2))$
  $= sqrt((x_j - w_{ij})*(x_j - w_{ij})^T)$ (vector inner product)
- Another measure of distance, used when the values are integer, is the "**Manhattan**" or "**city-block**" distance:
  $|| x - w_i || = sum_i(|x_j - w_{ij}|)$
- Another measure of distance, used when the values are **2-valued**, is the **"Hamming distance":** $sum_i(|x_j == w_{ij}|)$
  0 when the values are equal, 1 otherwise

A measure of similarity is given by the inner product

- The inner product x wi is **larger** when x is "closer to" wi.
- Usually it is best if x and wi are **normalized** before using this measure: || x || = || wi || = 1
- The normalized inner product is the *cosine*
- of the angle between x and wi as *vectors*



Example of different metrics

- Suppose x = [1 1 -1 1], w = [1 -1 -1 -1]
- Euclidean distance = sqrt($0^2 + 2^2 + 0^2 + 2^2$) = 2.83...
- Manhattan distance = 0 + 2 + 0 + 2 = 4
- Hamming distance = 0 + 1 + 0 + 1 = 2

smaller is closer

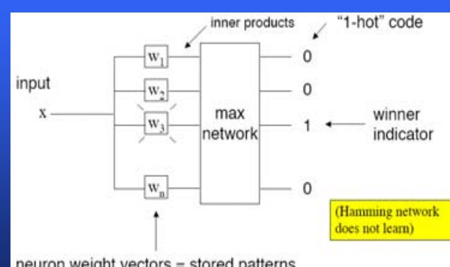
- inner product = [1 1 -1 1] $[1\ -1\ -1\ -1]^T$ = 0

larger is closer

Determining the winner

- The winner is the neuron with weight either:
- the smallest distance to the input, or
- the largest inner product with the input.
- Again, if inner products are used, it is best to normalize the weight and input first, or use only normalized values.

Example: Hamming network



Max sub-network

- a recurrent neural net that cycles values through neurons, eliminating one loser each cycle until only the winner is left.
- Each neuron has as inputs the outputs of all neurons including itself.
- Self-weights are 1;
  Weights from other neurons are -ε, where ε is any quantity < 1/(# of neurons).

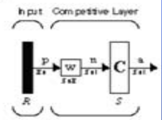
Max network

- Activation functions are “poslin”:
  poslin(x) = x if x > 0, 0 otherwise
- The network is operated synchronously.
- The initial outputs are forced to those of the input values.
- On each cycle, each neuron computes poslin(weighted inputs).
- For the ith neuron yi := poslin(yi -εΣ yj)

= (1+ε)yi -εΣ yj
- These weights are designed so that:
- all but one output is non-zero after n cycles (assuming inputs were originally distinct)
- all outputs persist at the same value after n cycles

Matlab compet function (non-learning)

COMPET(N) takes one input argument,
N - SxQ matrix of net input (column) vectors.
and returns output vectors **with 1 where each net input vector has its maximum value, and 0 elsewhere.**



a = compet(n)
= compet(Wp)

compet([-3; -1; 5; 2; -9]) ==> (3,1) 1

column vector — column separators — sparse matrix notation: row 3, col 1 = 1

Using Competition in Conjunction with Learning

- Input presented
- Winner selected
- The winner learns
- Others “close to” winner may learn as well.

Instar rule

Instar Rule (Stephen Grossberg)

$$ {}_i\mathbf{w}(q) = {}_i\mathbf{w}(q-1) + \alpha a_i(q)(\mathbf{p}(q) - {}_i\mathbf{w}(q-1)) $$

pattern - weight

1 for i = winner
0 otherwise

learning rate

Only winner learns

Kohonen rule

input

$$ {}_{i^*}\mathbf{w}(q) = {}_{i^*}\mathbf{w}(q-1) + \alpha(\mathbf{p}(q) - {}_{i^*}\mathbf{w}(q-1)) $$

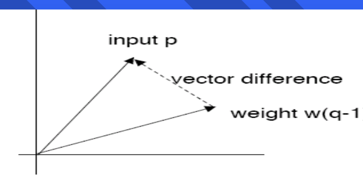
index of winners

$$ {}_{i^*}\mathbf{w}(q) = (1-\alpha)\,{}_{i^*}\mathbf{w}(q-1) + \alpha\mathbf{p}(q) $$

$$ {}_i\mathbf{w}(q) = {}_i\mathbf{w}(q-1) \qquad i \quad i^* $$

In the general Kohonen rule, there can be multiple “winners”.

Graphical representation



$$ {}_{i^*}\mathbf{w}(q) = {}_{i^*}\mathbf{w}(q-1) + \alpha(\mathbf{p}(q) - {}_{i^*}\mathbf{w}(q-1)) $$

$$ {}_{i^*}\mathbf{w}(q) = (1-\alpha)\,{}_{i^*}\mathbf{w}(q-1) + \alpha\mathbf{p}(q) $$