

## Supervised learning: backpropagation

"Learning is a process by which the free parameters of a neural network are adapted through a continuing process of stimulation by the environment in which the network is embedded. The type of learning is determined by the manner in which the parameter changes take place."

Haykin, 1994, "Neural Networks, A comprehensive foundation".p 50

## Revision: unsupervised learning Hebbian Learning

When two cells fire at the same time the strength of the connection between them should be increased

The weights are modified according to the activation of the input and output units

$$\Delta w_{ij} = \epsilon a_i a_j$$

$\Delta w_{ij}$  delta  $w_{ij}$ , is the weight from unit  $i$  to unit  $j$

$\epsilon$  Epsilon is the learning rate  
 $a_i$  is the activation of unit  $i$

Benefits:

- One shot learning
- Simple
- Biologically plausible
- additive

Drawbacks

- Some patterns cannot be learned
- Interference between similar patterns

## Delta Learning

Supervised learning always has a teacher

The weights are modified relative to the difference between the target and actual outputs

$$\Delta w_{ij} = \epsilon a_i e_j$$

where

$$e_j = t_j - a_j$$

$t_j$  is the teaching input, or target for unit  $j$

Benefits:

- Guaranteed to learn for all problems that can be solved without hidden units
- Can learn non-orthogonal patterns
- Accuracy improves with training

Drawbacks

- Needs repeated passes through the training set
- Some patterns still can't be learned

## The perceptron convergence procedure

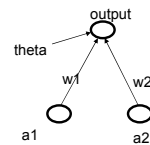
PCP is the delta rule with the learning rate, epsilon=1, and threshold activation fns

Apply the PCP to solve the AND problem for  $w_1=-0.5$  and  $w_2=0.5$  and  $\theta=1.5$

Pattern	Input1	Input2	Target
	A1	A2	T
P1	1	1	1
P2	1	0	0
P3	0	1	0
P4	0	0	0

$$netin = a_1 w_1 + a_2 w_2 - \theta$$

$$output = 1 \text{ if } (net > 0), \text{ else } 0$$



See excel table

## Multi-layer networks

- Some problems cannot be solved with a two layer net (one input layer, one output layer) as demonstrated by Minsky & Papert in 1969.
- E.g., XOR
- A hidden layer is required. The hidden layer forms a recoding of the input patterns
- With the right connections and sufficient hidden units, a multi-layer net can perform any mapping from input to output.
- How does one train a network with hidden units?
- This problem is the credit assignment problem – which hidden unit weights contribute to the output error?

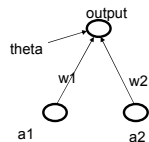
## XOR

the PCP will never solve the XOR problem

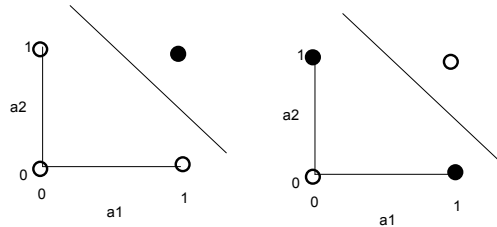
Pattern	Input1	Input2	Target
	A1	A2	T
P1	1	1	0
P2	1	0	1
P3	0	1	1
P4	0	0	0

$$netin = a_1w_1 + a_2w_2 - \theta$$

$$output = 1 \text{ if } (net > 0), \text{ else } 0$$



## AND and XOR



## Backpropagation

General procedure:

1. Assign random values to the weights
2. Present a pattern and target and calculate the error at the output layer
3. Estimate the error at the previous layer
4. Change all the weights a little bit to minimize the errors

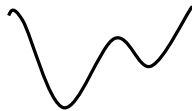
$$\Delta w_{ij} = -\epsilon \left[ \frac{\partial E}{\partial w_{ij}} \right]$$

$$= \epsilon d_i a_j$$

$$d_j = \delta \text{ (if output unit)}$$

$$\text{or } \sum d_k w_{jk} \text{ (if not)}$$

$$\delta_k = t_k - o_k$$



## The backpropagation algorithm

for a feedforward network

(also called a multi-layer perceptron)

- Forward pass: for each pattern
  - propagate activation from the input to the output
- Backward pass:
  - Calculate the error at the output
  - Propagate the error backwards through the network to estimate the contribution to the error from each unit
  - Change each weight by a small amount so as to reduce the total error

## The forward pass

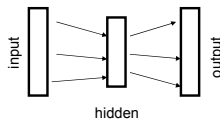
- Input to hidden
  - For each hidden unit,  $h_i$

$$net_{hi} = \sum_j a_j w_{ij} + \theta_i$$

$$h_i = f(net_{hi})$$

$$= 1 / [1 + e^{-net_{hi}}]$$

- Hidden to output
  - As above.



## The backward pass

- Error for each unit

$$error = (t_k - o_k)^2$$

- Pattern sum squared error (PSS)

$$PSS = \sum_k (t_k - o_k)^2$$

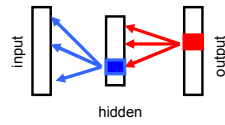
- Calculate the derivative of the error wrt each H->O weight

$$\Delta w_{ij} = \partial TSS / \partial w_{ij}$$

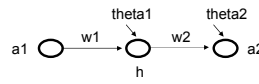
- Propagate that error backwards

$$w_{ij}^{new} = w_{ij}^{old} + \epsilon \Delta w_{ij}$$

## The backward pass



## Example 1 1 network

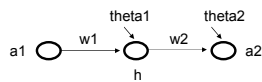


Input	Target
0	0
1	1

For this example, set  $\theta_1 = \theta_2 = 0$

1. Write the activation eqns for the hidden unit
2. Write the activation eqns for the output units
3. Write the equation for the TSS

## Example 1 1 network



Input	Target
0	0
1	1

For this example, set  $\theta_1 = \theta_2 = 0$

1. Write the activation eqns for the hidden unit

$$net_h = a_1 w_1; h = 1 / (1 + e^{-a_1 w_1})$$

2. Write the activation eqns for the output units

$$net_2 = h w_2; a_2 = 1 / (1 + e^{-h w_2})$$

3. Write the equation for the TSS

$$TSS = \sum_{p=0,1} (t - a_2)^2$$

$$= (0 - a_{2(p=0)})^2 + (1 - a_{2(p=1)})^2$$