Artificial neural networks, simple supervised learning

AIMS

- Understand the aim of using neural networks for pattern classification.
- Describe how a set of examples of stimuli and correct responses can be used to train an artificial neural network to respond correctly via changes in synaptic weights governed by the firing rates of the pre- and post-synaptic neurons and the correct post-synaptic firing rate.
- Describe how this type of learning rule is used to perform pattern recognition in a perceptron.
- Discuss the limitation of the perceptron.

READING

Books
1,2,5.

recap: Learning in neural networks

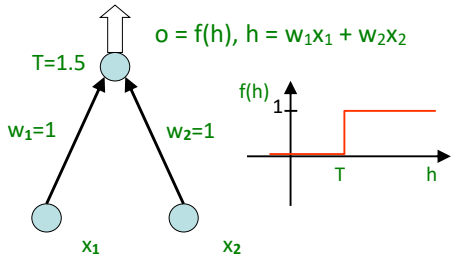
The problem: find connection weights such that the network does something useful.

Solution:

Experience-dependent learning rules to modify connection weights, i.e. learn from examples.

1. 'Unsupervised' (no 'teacher' or feedback about right and wrong outputs)
2. 'Supervised':
   A. Evolution/genetic algorithms
   B. Occasional reward or punishment ('reinforcement learning')
   C. Fully-supervised: each example includes correct output.

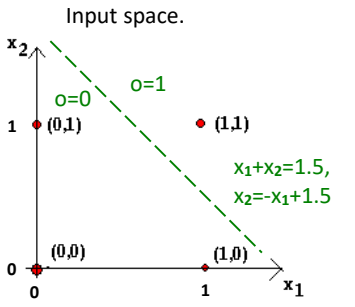
Pattern classification: linear separation of categories



Consider the network with weights w_1 and $w_2 = 1$, threshold T=1.5 and a threshold logic transfer function.

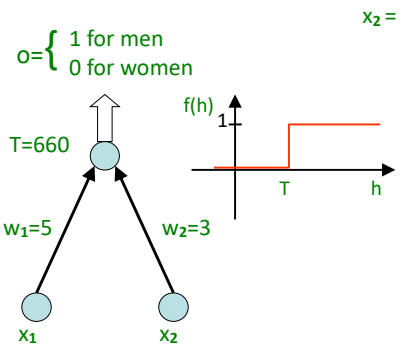
x_1	x_2	h	output
0	0	0	0
0	1	1	0
1	0	1	0
1	1	2	1

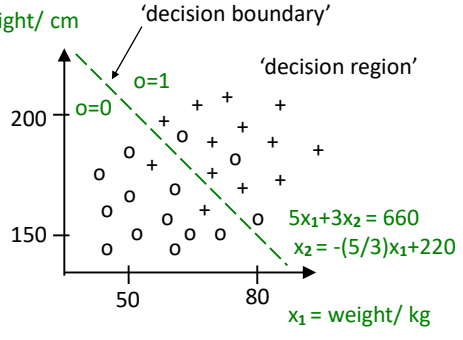
It performs the logical 'AND' function



The line separating o=1 from o=0 is where the net input equals the threshold:
$w_1 x_1 + w_2 x_2 = T$, ie line:
$x_2 = -(w_1/w_2)x_1 + T/w_2$ (in y = mx+c format).
Changing the weights changes the line. So learning to classify two groups of input patterns means finding weights so that the line separates the groups.

Pattern classification, example

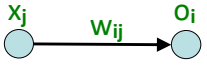




'Learning' or 'training' on example patterns is used to define the weights and thus the 'decision region'. Performance depends on how well it 'generalises' i.e. how well it classifies new data.
These networks perform 'linear discrimination': the decision boundary is always linear.

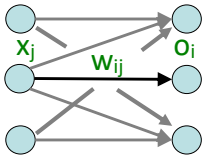
Reminder: the 'Hebb rule

$w_{ij} \rightarrow w_{ij} + \varepsilon\, x_j\, o_i$



Δw_{ij}	x_i = 0	x_i = 1
x_j = 0	-	-
x_j = 1	-	+

Supervised learning: the 'delta rule'



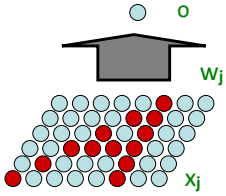
- Given 'training set' of inputs $\underline{\mathbf{x}}^n$ $(x_1^n, x_2^n ..)$ with target output valuess $\underline{\mathbf{t}}^n$ $(t_1^n, t_2^n ..)$ where $n=1,2,..$
- Present input pattern n and find the corresponding output values $\underline{\mathbf{o}}^n$ $(o_1^n, o_2^n ..)$
- Change connection weights according to:

  $w_{ij} \rightarrow w_{ij} + \varepsilon\, x_j^n (t_i^n - o_i^n)$
- Present the next input pattern: $n \rightarrow n+1$

The term $(t_i^n - o_i^n)$ is also known as 'delta' δ_i^n: the 'error' made by output i for input pattern n.
The delta rule is like the Hebb rule, but with post-synaptic term δ to change weights so as to reduce 'error'.

The perceptron (Rosenblatt, 1962)

output =1 if pattern detected (threshold logic function)



'Artificial retina'

Present input patterns $\underline{\mathbf{x}}^n$ (some are pattern to be detected with target $t^n=1$, others are 'foils' to be ignored with target $t^n=0$).

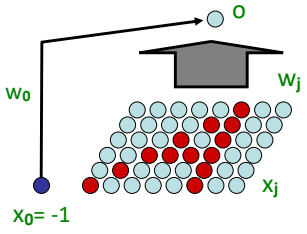
For each pattern apply the 'delta rule':
$w_j \rightarrow w_j + \varepsilon\, x_j^n (t^n - o^n)$

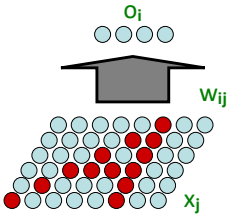
After many presentations of the whole training set (in random order) it can find the best linear discriminator of targets from foils.

Notice that w_j changes only if $t^n \neq o^n$ and $x_j^n \neq 0$.
If $t^n > o^n$ then w_j increases; if $t^n < o^n$ then w_j decreases,
i.e. the delta rule changes weights so as to reduce the error

Perceptrons, thresholds and multiple outputs.

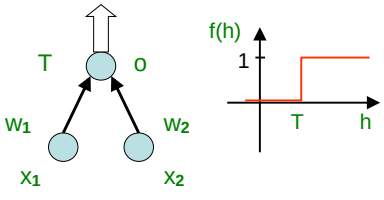
Q: The delta rule only learns weights, how do we find the value for the threshold T?
A: Just use T=0 and add another input x_0=-1, then the weight from it w_0 can serve the same purpose: the condition for output to be active: $w_1x_1+w_2x_2+..>T$
is the same as $w_0x_0+w_1x_1+w_2x_2+..>0$ if x_0=-1 and w_0=T
- Then the delta rule can find connection weight w_0 (which is the same as finding T).





Perceptrons can have many output units (forming a single-layer neural network) – each output is trained using the delta rule independently of the others.

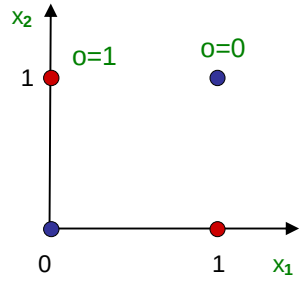
Non-linearly separable problems



$w_i \rightarrow w_i + \varepsilon\, x_i^n (t^n - o^n)$

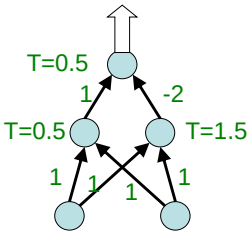
Perceptrons perform linear discrimination and can't solve 'non-linearly separable' problems (Minsky & Papert, 1969)

e.g. XOR 'exclusive OR'

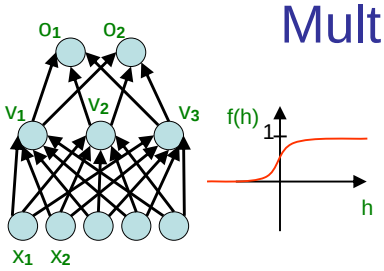


o	x_1 0	+1
0	**0**	**1**
x_2 +1	**1**	**0**

'Hidden' processing units can give more power, e.g:



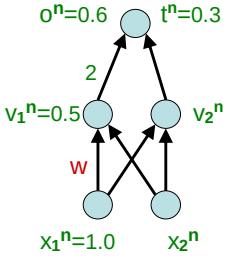
Multi-layer perceptrons



Delta rule:

$w_{ij} \rightarrow w_{ij} + \varepsilon\, x_j^n (t_i^n - o_i^n)$

Example:



If w decreases the error for input pattern n reduces

Q: How train connection weights if the 'internal representation' is not known (i.e. t_i^n not known for 'hidden layer', x_j^n not known for output?

A: if the neurons use a smooth (continuous) transfer function, changing a connection weight from any (active) neuron anywhere in the network will change the output firing rate slightly. If the output o_i^n is now closer to its target value t_i^n then the change was good..

But note: The effect of changing a connection weight depends on the values of the errors $\delta_i^n = (t_i^n - o_i^n)$, connections w_{ij} and activations v_i further up in the network..