

31YB: Biologically Inspired Computing

Lecture 6:
(Mon, 13th Oct)

Delta Rule & Introduction to Multi-layer Perceptrons

Recap: Rosenblatt's Perceptron Learning Rule (PLR)

(for training the Perceptron/McCulloch Pitts Neuron Model)

1. Initialise weights at random
2. For each training pair/pattern (\mathbf{x}, y_{target}):
 Compute output y (of Perceptron/McCulloch-Pitts Neuron Model)
 Compute error, $d = (y_{target} - y)$
 Use the error to update weights as follows:
 $\Delta \mathbf{w} = \mathbf{w} - \mathbf{w}_{old} = \eta d \mathbf{x}$ or, $\mathbf{w} = \mathbf{w}_{old} + \eta d \mathbf{x}$
 where η is a constant learning rate (see later)

3. Repeat 2 until convergence [i.e. error d is zero]

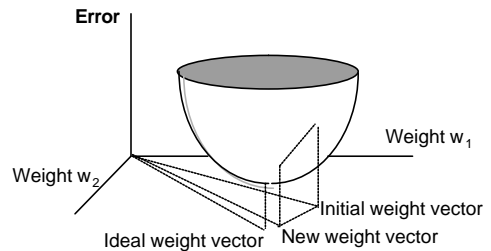
Example of PLR Update rule (assume $\eta = 0.5$) :

if $y = +1$ & $y_{target} = -1$ then $\mathbf{w} = \mathbf{w}_{old} - \mathbf{x}$
 if $y = -1$ & $y_{target} = +1$ then $\mathbf{w} = \mathbf{w}_{old} + \mathbf{x}$
 if $y = y_{target}$ then $\mathbf{w} = \mathbf{w}_{old}$

Recap: For interest (NOT Examinable): Proving the Delta Rule

- To prove the learning rule for the DR (which has same equation as PLR, but with important differences as mentioned earlier), we must show that it implements gradient-descent in a suitable measure of error, E .
- **Mathematical Exercise for you!**
- *Hint:* Assume there is no threshold activation function, and show that the weight change, $\Delta \mathbf{w}$ always moves in the opposite direction from the gradient, that is, it implements a gradient descent. (See hand-out + tutorial 1 question)

Graphically: Delta Rule performs Gradient Descent



Review:

what are the limitations of perceptrons (i.e. McCulloch Pitts/threshold Neurons/Units) ?

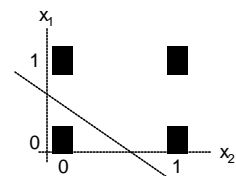
...

Perceptron Learning Theorem

Recap: A perceptron (threshold unit) can *learn* anything that it can *represent* (i.e. anything separable with a hyperplane)

OR function

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	1

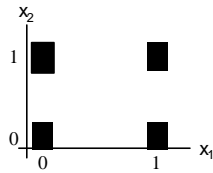


The Exclusive OR problem

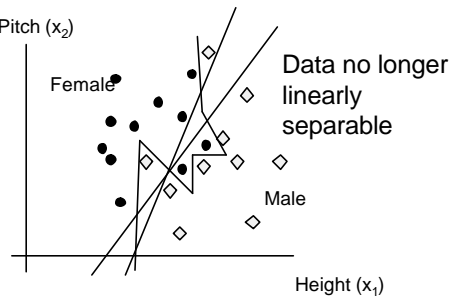
Recap: A Perceptron cannot represent Exclusive OR since it is not linearly separable.

XOR function

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0



Voice Pitch (x_2)

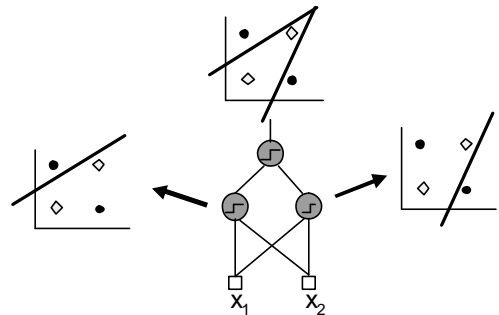


What is a good decision boundary ?

So: To sum up

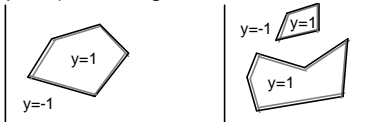
- PLR can learn associations, but:
 - can only learn linearly separable classifications
- DR can produce a mapping which minimise mean squared-error, but:
 - error is almost certain to remain non-zero because of the limited range of functions f possible
- Both DR & PLR limited by single-layer architecture they work in
- Any solutions?
 - multi-layered architecture, i.e. a Multi-layered Perceptron (MLP) can overcome these limitations. It can learn arbitrary mappings or classifications... (see next!)

Piecewise linear classification using an MLP with threshold (perceptron) units



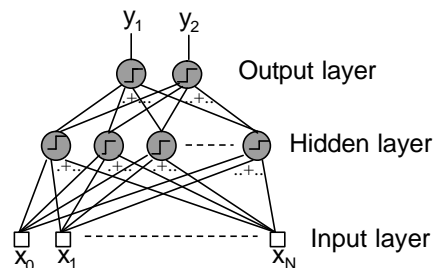
Arbitrarily complex decision boundaries

can be generated by MLPs with threshold (perceptron) units, e.g.

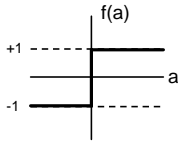


- But this *theoretical* result doesn't tell us *how* to set the weights (i.e. no learning algorithm)
- Rosenblatt (inventor of Perceptron/PLR) could only train a single (output) layer of Perceptrons

Note: The General Multi-layer Perceptron (MLP) architecture (with threshold units) would look like...



Recap: The threshold activation function, $f(a)$, used in Rosenblatt's Perceptron:



One possible approach to MLP Training:
Can we use a generalised form of the PLR/Delta rule to train the MLP?

- Recall the PLR/Delta rule: adjusts neuron's weights to reduce error at neuron's output:

$$\mathbf{w} = \mathbf{w}_{old} + \mathbf{hd}\mathbf{x}$$

where $\mathbf{d} = y_{target} - y$

◆ **Main Problem**- 'How to adjust the weights in an earlier layer so they reduce the error in the later layer?' i.e. how can we back-propagate the error d ?

- answer to this crucial question is of great historical significance – see next!

Historical Perspective:

Rise of ANN: Recall: Neural nets applied in the 1950's & 60's to weather prediction, ECG analysis and simple pattern recognition using Perceptrons (McCulloch-Pitts Neuron Models).

Fall of ANN: Minsky & Papert Report: Perceptrons (1969): Single layer nets theoretically incapable of solving many simple problems (e.g. XOR !). Extension to multi-layer nets sterile since no learning algorithm (cf. previous slide!).

- Most researchers left field, funding scarce
- Research almost halted for over 15 years (cf. Lighthill & Artificial Intelligence) ..until..invention/discovery of ...

Back-Propagation

Learning algorithm proposed by:
 Rumelhart, Hinton & Williams 1986
 Parker 1982
 Werbos 1974
 Brison *et al.* 1963!

- **Permits training of multi-layer networks (MLP)** – led to current rise of ANN (mid-80's to-date)
 – **Note: Hopfield Network developed in mid-80's also responsible for the resurgence – see later!**

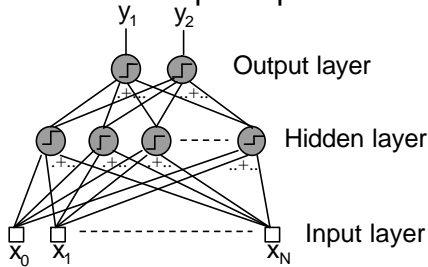
Today MLPs have been trained

- To convert text to speech (1987)
- To recognise handwritten text (1987)
- To classify cervical cells as normal/abnormal (1990)
- To detect faces in complex scenes (1995) etc.
- ◆ Most ANN applications use MLPs
- ◆ It works (see later! + see labs, assignment)

Summary of course so far

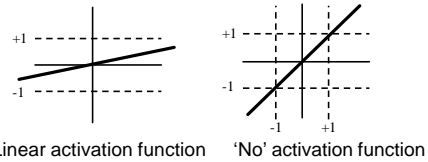
- Biologically Inspired Computing
- Learning & Generalization
- Perceptron Learning Rule (PLR)
- Delta Rule (DR)
- History
 - Rise of Neural Nets
 - Fall due to Minsky & Papert
 - Rise due to Multi-layer nets
- Next, how does the Back Propagation (BP) learning work for the MLP

Recap: Multi-layer Perceptron (MLP) with threshold perceptron units



Above MLP can form arbitrarily complex decision boundaries BUT **How to train it** ?

One Possible Solution to MLP Training: Use Linear activation functions (instead of threshold functions) for deriving generalized form of DR for training MLP?

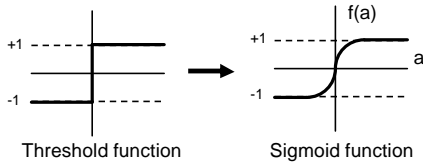


BUT: A neuron with a linear activation function performs a purely linear operation on its inputs. Multiple layers of such neurons are no more powerful than a single layer!

Another Possible Solution to MLP Training:

Another Simple idea! for deriving generalized form of DR for training MLP:

- Alter non-linear Perceptron (threshold) activation function so that we can relate the output to the weights in a more informative (non-linear) way (real values of outputs possible), and help derive Generalized DR



You'll hear claims for biological plausibility but this mathematical 'fix' is not based on biology!!

Choice of Sigmoid (S-shaped) functions -Properties

- Approximates a threshold function
- Differentiable (smooth everywhere)
 - hence Delta Rule (DR) applicable

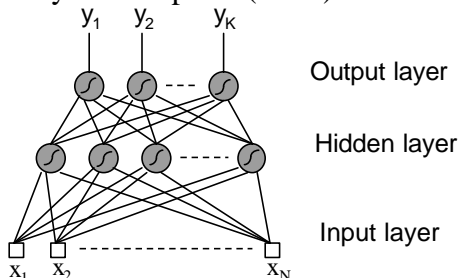
- Positive slope

Popular choice is:
$$y = f(a) = \frac{1}{1 + e^{-a}}$$

Simple derivative:

$$f'(a) = f(a)(1 - f(a))$$

Hence, resulting in the following general Multi-layer Perceptron (MLP) used today



Note: Now sigmoid activation functions on hidden & output Perceptron units (rather than threshold)

Lecture Summary

- MLPs: piecewise-linear
- 'Rise and fall' of ANN's
- Sigmoid activation functions for training MLP
- NOTE:** Tutorial 1 during next normal Lecture slot (Thursday, 16 October, 10:00, 2A87B) & Practicals start next week + Assignment handed out next week too!
- Next lecture (Monday, 20 October)
 - More on MLP's and back-propagation