31YB: Lecture 3

Mon, 29th Sep 2003

Lecture Overview

- Decision Surfaces
- Examples: Simple Logical Predicates

Recap: Elements of model neuron III: Axon

- In this model, it simply transmits the neurons output to other, or even the same
    - neurons
    - May be system output
    - might, in other models, implement a delay
- The model's output function corresponds to the axon hillock of the real neuron
- Many different **output functions** have been used e.g. threshold, sigmoidal etc.
- For example, the model neuron may be a *threshold unit*
    - (as in the original McCulloch-Pits model),
- using a threshold θ (which could equal e.g. 0.5)

  $Y_j = 1$ if $A_j \geq \theta_j$ or,

  $Y_j = 0$ if $A_j < \theta_j$

  – This is a non-linear operation
- Another simple possibility is a linear unit

  $Y_j = k * A_j$

  - which gives a purely linear unit

Recap: Other Output functions

- Other than the threshold function, one can use e.g.
- Logistic (or squashing) function in a model neuron

$Y_j = 1/(1+exp(-k * A_j + B_j))$

- or

$Y_j = tanh(k * A_j + B_j)$

- Note that *k* determines the slope, and B_j the output of the neuron for 0 input (which is termed the bias, usually a constant - *see later*).
- **More generally**, $Y_j = f(A_j) = f(\Sigma_{i=1}^{n} w_{ji} I_i)$
- Normally, a *monotonically increasing* function *f* is used. (Why? - *see later*)

Recap: Note!

- Real neurons are a great deal more complex
- Yet for most of the work we will consider, very simple neural models will suffice.
- NN systems gain their power by using a large number of very simple processing elements in concert.
- There is also a great deal of interconnection
  - and this is like the brain: there are miles of axon "wire" in every cubic cm of brain !

What can a threshold unit do?

- *Simple threshold units* have output, Y, equal to 1 if

$$\Sigma_{i=1}^{N} w_i I_i > \theta$$

And output 0 otherwise

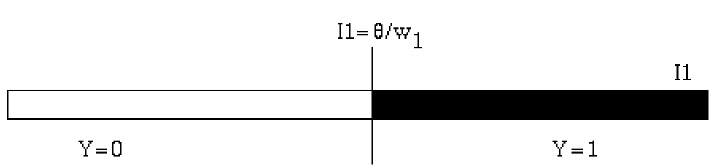
- N+1 parameters, (N weights, θ threshold parameter)
- known as a *decision unit*

Decision Surfaces

- The decision surface is the surface at which the output of the unit is precisely equal to the threshold i.e. $\Sigma w_i I_i = \theta$
  - On one side of this surface, the output of decision unit, Y, will be 0, and on the other side it will be 1.

In 1-dimension: the surface (just a point) is:

$$I_1 = \theta / w_1$$



Decision Surfaces in 2-D

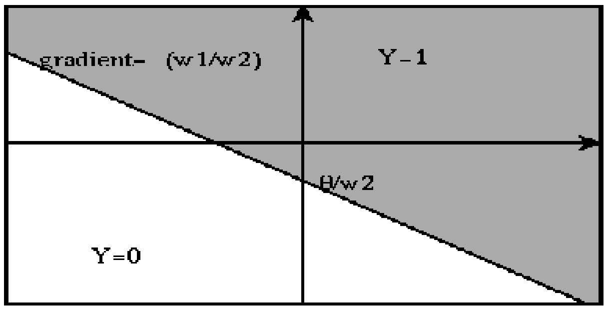
In 2 dimensions, the surface is

$$w_1 I_1 + w_2 I_2 = \theta$$

which we can write

$$I_2 = -\frac{w_1}{w_2} I_1 + \frac{\theta}{w_2}$$

which is the equation of a line of gradient -(w1/w2) with intercept θ/w2



General decision surfaces

- In general, the decision surface is a *hyperplane* of dimension one less than the dimension of the input space.

Input Space	*Hyperplane*
line	point
surface	line
3-d Volume	surface

- the decision surface cuts the input space into two halves
- If the threshold is 0, then the decision surface passes through the origin
- instead of a threshold, we can use a *bias unit*, which uses a bias 'weight' w_b, giving (for N=2)

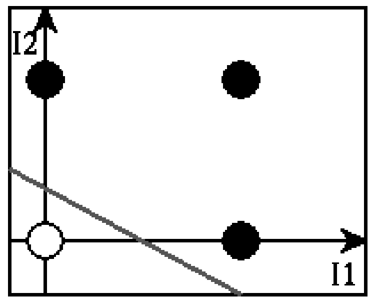
$$w_1 I1 + w_2 I2 + w_b.1 = 0$$

- Precisely equivalent to a threshold of $\theta = -w_b$, which is multiplied by an additional 'bias' input which is always 1
  - but makes *all* the parameters into weights – *more later!*

Decision surfaces for simple logical predicates: OR

- Consider a 2-input neuron system, restricting I1 & I2 to be 0 or 1.
- identify 0 with FALSE and 1 with TRUE, we can consider the logical predicates such as AND and OR

I1	I2	I1 OR I2
0	0	0
0	1	1
1	0	1
1	1	1

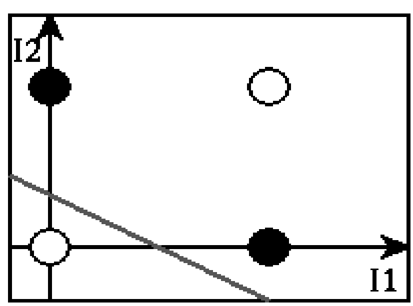


- OR and AND (try yourself) are easy to implement: we can see that the gradient of the decision surface should be negative, and the intercept between 0 & 1 for implementing OR logic (& greater than 1 for AND logic)

Simple logical predicates: EOR

- Can we implement Exclusive-Or this way?

I1	I2	I1 EOR I2
0	0	0
0	1	1
1	0	1
1	1	0



- EOR poses impossible problems:
- no single straight-line decision surface can separate the filled circles from the open circles (problem of **non-linear separability** – *see later*)
- EOR cannot be implemented in this way, so what can we do? – use a **network** of neurons! - *see later*

Next Lecture: **Learning & Generalization**