Artificial Neural Networks *week 4 & week 5*

Biological neurons have cell bodies, dendrites (input structures), and axons (output structures). Axons connect to dendrites via synapses. Electro-chemical signals travel from the dendrite, through the cell body, to the axon, then onto other neurons. Neurons only fire if their input signals exceed a threshold in a short period of time. Synapses vary in strength: strong connections allow large signals while weak connections only allow small signals.

Artificial neural networks emulate a biological neural system, consisting of nodes ('neurons') and weights ('neuronal connections'). Each node has weighted connections to several other nodes in adjacent layers and takes input from connected nodes, using the weighted inputs combined and a simple function to compute its output values. Knowledge is stored in the connections between neurons.

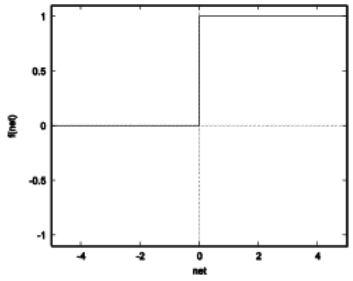
To train a neural network: introduce data; the network computes an output; the output is compared to the desired output; the network's weights are modified based on the difference between the two to reduce error. To use a neural network: introduce new data to the network; the network computes an output based on its training.

In a single layer network, an adder sums up all the inputs, modified by their respective weights (linear combination). An activation function controls the amplitude of the output of the neuron. An acceptable output range is usually between -1 or 0 and 1.

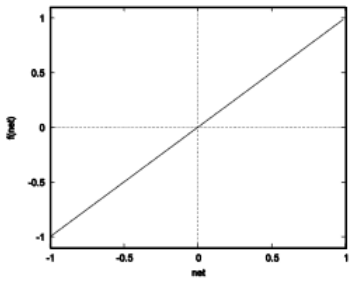
Mathematical model:
$I_1 \dots I_n$ = inputs,
$w_1 \dots w_n$ = weights,
net: summation,
f = activation function,
θ = bias/threshold,
O = output,

$$\bar{\mathbf{I}} = \begin{bmatrix} I_1 \\ I_2 \\ \vdots \\ I_n \end{bmatrix} \qquad \bar{\mathbf{w}} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix} \qquad O = f\left(\sum_{j=1}^{n} w_j I_j - \theta\right) = f\left(\bar{\mathbf{w}}^T \bar{\mathbf{I}} - \theta\right) = f(net)$$

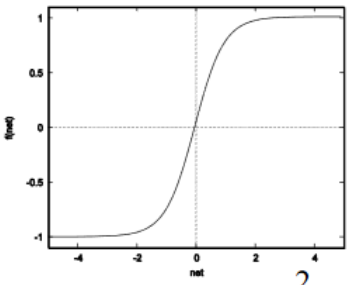
Activation functions:



Step: $f(net) = \begin{cases} 1, & net \geq 0 \\ 0, & net < 0 \end{cases}$



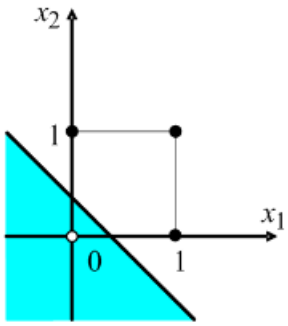
Linear: $f(net) = net$



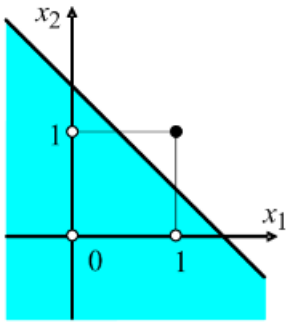
Sigmoid: $f(net) = \frac{2}{1+e^{-\lambda net}} - 1$

Bias can be implemented as an extra input. Connection strengths are modelled by a set of weights. The training process involves changing the weights and bias values.

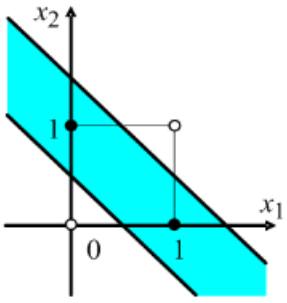
Single layer networks are simple and easy to implement and learn quickly: several examples are usually enough. However, they can only learn linearly separable functions. The graphs below show how three Boolean problems should be split, however a single layer network could only perform the left-most two as the third requires two lines.



$OR(x_1 \cup x_2)$



$AND(x_1 \cap x_2)$



$Exclusive\,OR$
$(x_1 \oplus x_2)$

Multi-layer neural networks are feedforward networks. They consist of an input layer of source neurons, at least one middle or hidden layers of computational neurons, and an output layer of computational neurons. Input signals are propagated in a forward direction on a layer-by-layer basis.

These networks can learn non-linear relationships, can solve 'difficult' problems (e.g. classification, stock prediction), and can usually run well with just one hidden layer. They can, however, be very slow as they have more complex learning methods and require lots of training data.

Artificial neural networks should be used when you are working with lots of data, non-linear and multidimensional input/output mapping, and lots of time. Learning rates should be very small, around 0.1. Artificial neural networks have many

uses, for example: pattern recognition, clustering, optimisation, control, and medical or business applications.

Their processing is massively parallel, meaning they are good for real-time applications. Artificial neural networks are self-trainable (they learn by themselves), only needing to increase or decrease connection strengths. They are excellent for generalisations and uncertain information, working well with noisy data. Once they have learned from data, they don't need to be reprogrammed.

However, artificial neural networks have high processing times if they are large, times which can rise quickly as the problem size grows. They don't explain their results, may not be guaranteed to converge to an optimal solution, and can possibly be over-trained.

Here's a link to week 5's step-by step run through on one page.