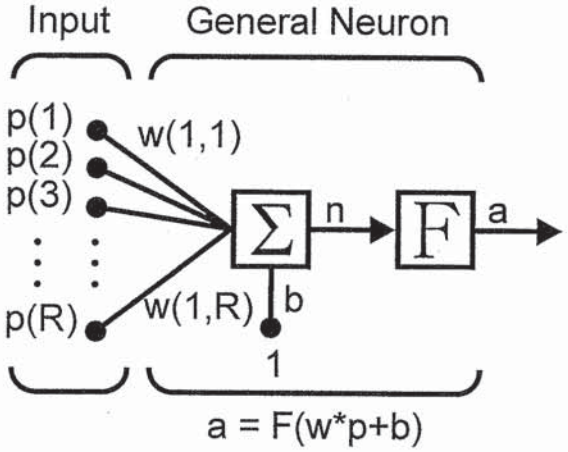
Backpropagation (反向傳播算法)

Introduction

- Backpropagation was created by generalizing the Widrow-Hoff learning rule to multiple-layer networks and nonlinear differentiable transfer functions.
- Networks with biases, a sigmoid layer, and a linear output layer are capable of approximating any function with a finite number of discontinuities.
- Reference: Rumelhart DE, Hinton GE, Williams RJ, "Learning internal representation by error propagation", Rumelhart D and McClelland J, editors. *Parallel Data Processing*, Vol.1, Chapter 8, the M.I.T. Press, Cambridge, MA 1986 pp. 318-362.
- Feed-forward networks with up to three layers can be created and simulated with **initff** and **simuff**. Such networks can then be trained with the backpropagation training function **trainbp**, or with a faster version of backpropagation, **trainbpx**. Another function, **trainlm**, is even faster but requires large amount of memory.

Neuron Model

An elementary backpropagation neuron with **R** inputs is shown below.



- Neurons may use any ***differentiable*** transfer function **F** to generate their output.
    - Backpropagation networks often use the log-sigmoid transfer function **logsig**. The function generates outputs between 0 and 1 as the neuron's net input goes from negative to positive infinity.

$$\text{logsig}(n) = \frac{1}{1+e^{-n}}$$

    - Alternatively, backpropagation networks may use the tan-sigmoid transfer function **tansig**. The function generates outputs between –1 and 1.

$$\text{tansig}(n) = \frac{2}{1+e^{-2n}} - 1$$

    - Occasionally, the linear transfer function **purelin** is used. The outputs can take any value.

  The above transfer functions are differentiable and they are also *monotonic increasing* functions. That is, the output of each function increases as its input increases.

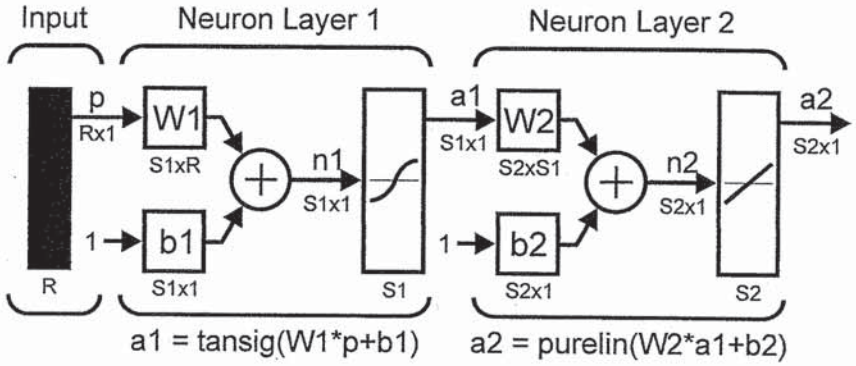
- In backpropagation it is important to be able to calculate the derivatives of any transfer function used. Each of the transfer functions above, **tansig**, **logsig**, and **purelin**, have a corresponding derivative function: **deltatan**, **deltalog** and

deltalin. To get the name of a transfer function's associated delta function, call the transfer function with the string **'delta'**.

```
tansig('delta')
ans = deltatan
```

Network Architecture (網絡架構)

Backpropagation networks often have *one or more* hidden layers of sigmoid neurons followed by an output layer of linear neurons.



- Multiple layers of neurons with nonlinear transfer functions allow the network to learn nonlinear and linear relationships between input and output vectors. The network shown above can be used as a general function approximator. It can approximate any function with a finite number of discontinuities, arbitrarily well, given sufficient neurons in the hidden layer.
- The output layer determines the range of output values. Linear output layer lets the network produce values outside the range –1 to +1. Sigmoid output layer (such as logsig) constrain the outputs (such as between 0 and 1).
- The function **simuff** simulates a feed-forward network. It takes the network input **p**, and the **weights (權重) W**, **biases (偏移) b**, and **transfer function (轉換函數)** for up to three layers, and returns each layer's output **a**.

  [a1, a2] = simuff(p, W1, b1, 'tansig', W2, b2, 'purelin');
- Below, **simuff** is called to calculate the outputs of one- and three-layer network.

  a = simuff(p, W, b, 'tansig')

  [a1, a2, a3] = simuff(p, W1, b1, 'tnasig', W2, b2, 'logsig', W3, b3, 'purelin')
- If **simuff** is called with a single output argument, it will always return the output of the last layer.

  a2 = simuff(p, W1, b1, 'tansig', W2, b2, 'purelin')