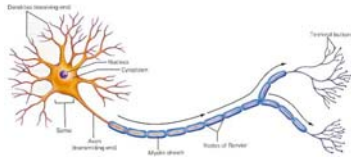


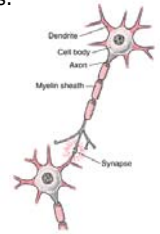
## The Connectionist Approach

- Most rationalist approaches to AI involve modeling cognitive processes in human behaviour
- The connectionist approach takes this one step further, creating models that are based on the activity of **neurons** in the brain



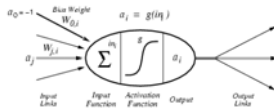
## Biological Neural Cells

- Each neuron is composed of three parts:
  - **dendrites**: receive electrical signals from other neurons
  - **axon**: transmits electrical pulse to other neurons
  - **cell body**: decides what kind of electrical pulse to transmit, given input signals
- The body's nervous system and the brain's higher functions are composed of networks of these neurons
  - **biological neural network**
- Gave rise to computational models of these networks
  - **artificial neural networks**



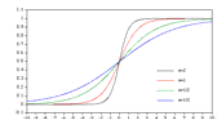
## Artificial Neural Networks

- Artificial neural networks (or simply neural networks) are made up of a series of nodes that represent the neurons of the brain
  - Each node (or unit) is connected to other nodes through directed links
  - Each node is defined by its **input function**, which sums the activations energies from other nodes, and an **activation function** that produces an output value for this node, based on the result of the input function.



## Neural Units

- **Input function**
  - the input function is a sum of the weighted inputs coming into the neural unit
  - the weights on each link are adjusted to achieve the learning mechanism for the neural network
  - the input function also has a **bias weight**  $W_{0j}$ , whose value is the threshold for the activation function
- **Activation function :  $g(\sum W_{ij}x_j)$** 
  - a function which gives a value near 1 when the "right" inputs are given, and a value near 0 otherwise
  - must be non-linear
    - e.g. threshold function, sign function, sigmoid function



$$\text{sig}(x) = 1 / (1 + e^{-ax})$$

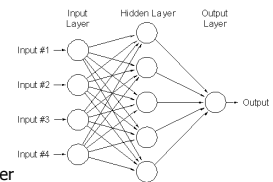
$$\partial \text{sig} / \partial x = \text{sig}(x) * (1 - \text{sig}(x))$$

## Neural Network Structure

- Two major kinds of neural networks:
  - **Feed-Forward (acyclic) networks**: node structures where the directed links between them do not form any loops
  - **Recurrent (cyclic) networks**: some nodes link back to previous nodes in the structure
    - important for maintaining state information from one reading of the neural network to the other
- Every neural network has three layers:
  - **input layer**: the nodes that record the input values for the training case, one node per input value (no processing)
  - **output layer**: the nodes whose activation functions produce the output values for the neural network
  - **hidden layer(s)**: the nodes between the input and output layers

## Hidden Layer "Rules"

- Hidden layers of neural networks store internal "knowledge", used in processing the output
- **Rules of Thumb**:
  - only one layer is really necessary
  - no universal rule for number of nodes in the hidden layer, except that more nodes implies more detail in output distribution → potential for overfitting, in sparse data cases
  - some problems require no hidden layer (i.e. linear separation problems)
    - use single-layer feed-forward neural network → **perceptron**



## Perceptrons

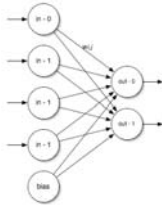
- Perceptrons can differentiate different sections in the input data, as long as they are linearly separable
  - similar to support vector machines

- How does learning occur within the network?

- want to minimize the squared error of the neural network

$$E = \frac{1}{2} \text{Err}^2 \equiv \frac{1}{2} (y - h_w(x))^2$$

- $y$  is the true output that should be expected, given the input  $x$ , weights  $w$  and actual output  $h_w(x)$



## Perceptron Learning

- In order to minimize this error, find the difference between the expected and actual output, and use that to adjust the weights that affect that output
- Adjustment will be a gradient descent method, meaning that the rate of change (derivative of  $E$ ) will be used to determine the amount of change in  $W$ .
- Factors in **Back-propagation algorithm** on weight  $W_j$ :
  - rate of change of  $E = \partial E / \partial W_j = \text{Err} \cdot \partial \text{Err} / \partial W_j = -\text{Err} \cdot g'(\text{in}) \cdot x_j$
  - current value of weight  $W_j$
  - learning rate**  $\alpha \rightarrow$  affects learning steps in algorithm
- Note:** sigmoid activation function makes the rate of change calculation easy, since derivative is well-formed

## Multilayer Learning

- Weight adjustment in perceptron is:

$$W_j \leftarrow W_j + \alpha \cdot \text{Err} \cdot g'(\text{in}) \cdot x_j$$

- In multilayer networks, this calculation only allows us to adjust the weights that feed into the output layer of the network. How can the weights for the other layers be adjusted?
  - Answer:** by propagating the error at the outputs back into each node in the hidden layer, adjusting for the intermediate weight values (see handout)

## Qualities of Neural Networks

- Advantages:
  - Biological basis
    - learning models human learning techniques; more sound than other contrived techniques
  - Auto-organization
    - internal representation of knowledge is not outlined or affected by human operators
  - Fault tolerance
    - partial destruction of network structure is compensated for by parallel operation of remaining network
  - Flexibility
    - learned model can work on unseen data
  - Speed
    - once model is trained, responses are obtained in real-time
- Disadvantages
  - Arcane quality
    - can't inspect internal workings to determine what it learned, since hidden state values don't mean anything at a higher level