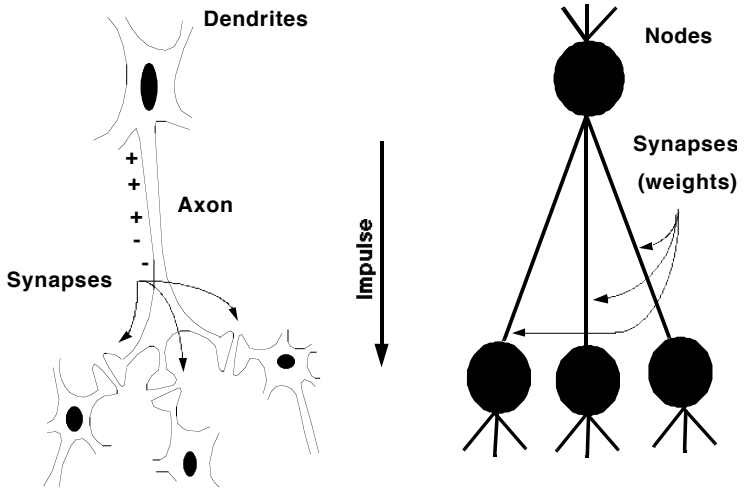
PERCEPTRONS

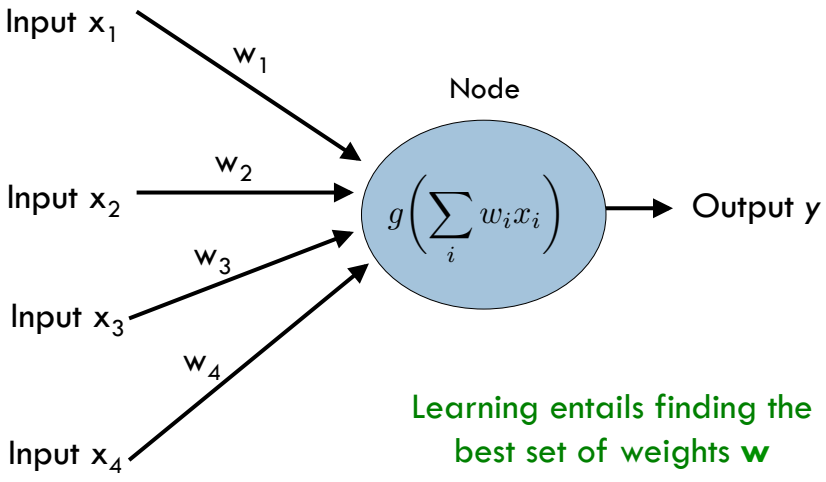
Today

- Reading
  - AIMA 18.6-18.8
  - Note: 18.6 covers regression but also sets up the mathematical background/notation for neural networks
- Goals
  - Perceptron (networks)
  - Perceptron training rule
  - (Feed-forward neural networks)

Motivation: Our Nervous System

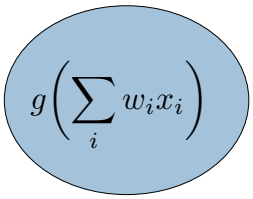


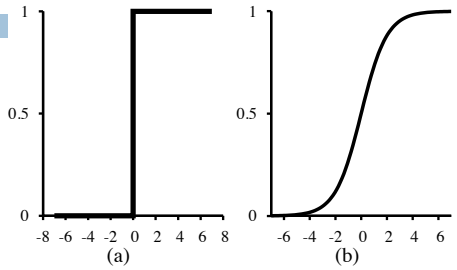
A single perceptron



Learning entails finding the best set of weights **w**

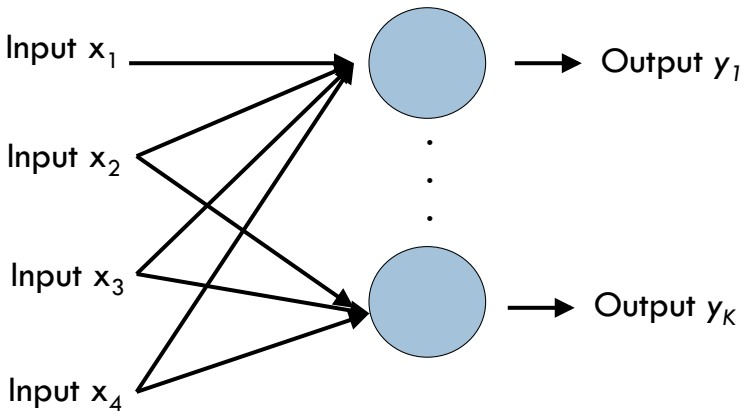
Activation function





- The activation function determines if the "electrical signal" entering the neuron is sufficient to cause it to fire
  - Threshold function – range is {0,1}
  - Sigmoid function – range [0,1]
  - Hyperbolic tangent function – range [-1,1]

A perceptron network

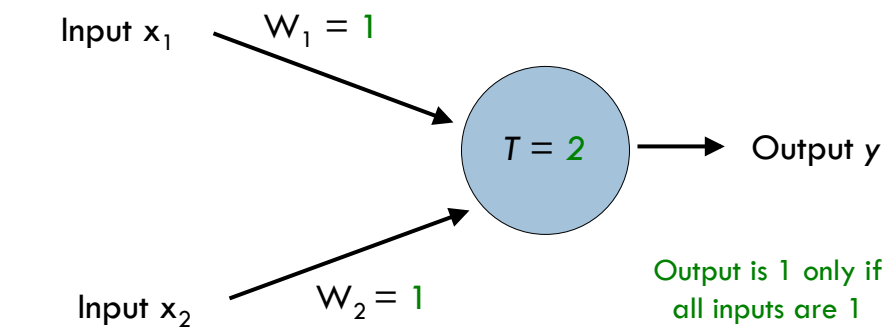


Reduces to K independent perceptrons

Example: logical operators

- **AND**: If all inputs are 1, return 1. Otherwise return 0
- **OR**: If at least one input is 1, return 1. Otherwise return 0
- **NOT**: Return the opposite of the input
- **XOR**: If exactly one input is 1, then return 1. Otherwise return 0

AND



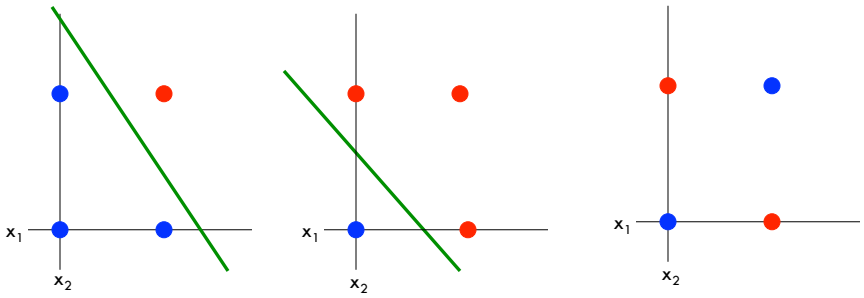
Inputs are 0 or 1

Perceptrons: Linearly separable functions

x_1	x_2	x_1 and x_2
0	0	0
0	1	0
1	0	0
1	1	1

x_1	x_2	x_1 or x_2
0	0	0
0	1	1
1	0	1
1	1	1

x_1	x_2	x_1 xor x_2
0	0	0
0	1	1
1	0	1
1	1	0



Perceptron Training rule

- Need an algorithm for finding a set of weights w such that
  - The predicted output of the neural network matches the true output for all examples in the training set
  - Predicts a reasonable output for inputs not in the training set

Perceptron Training Rule

1. Begin with randomly initialized weights
2. Apply the perceptron to each training example (each pass through examples is called an epoch)
3. If it misclassifies an example **modify the weights**
4. Continue until the perceptron classifies all training examples correctly

(Derive gradient-descent update rule)