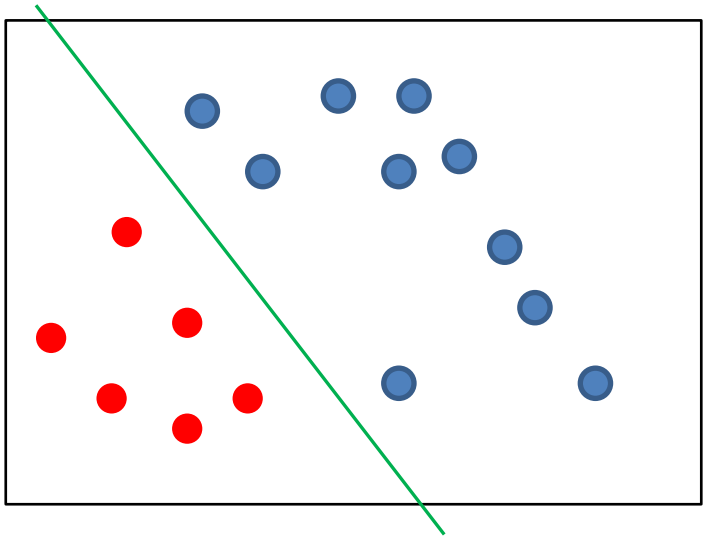
Perceptron

Presenter: Seok Ho-Sik

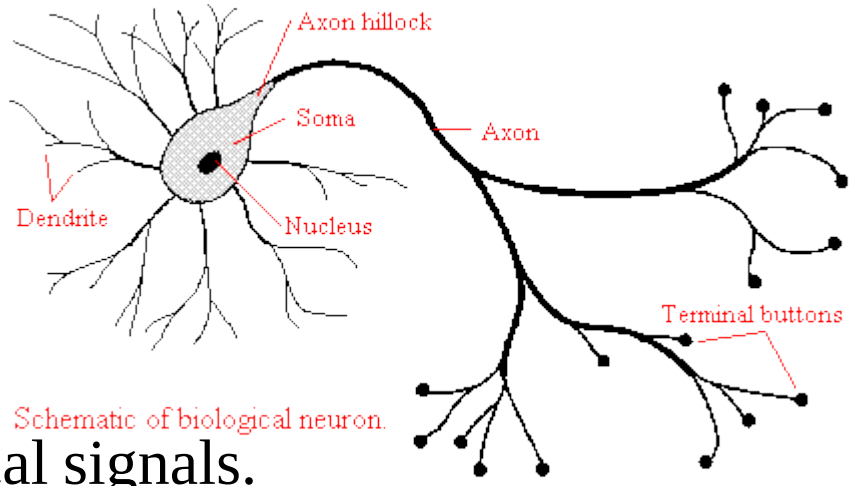
Reference: Machine Learning, Tom Mitchell, McGraw Hill

Hypothesis space and classification



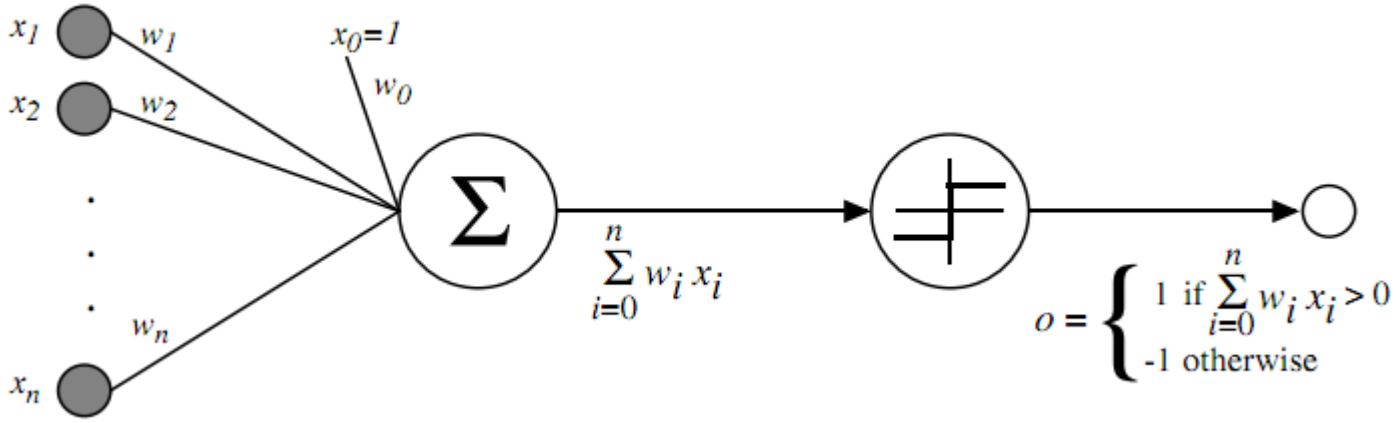
- Question: how to separate these two groups?

Neural networks



- Neuron
  - A cell in the brain whose principal function is the collection, processing, and dissemination of electrical signals.
  - The brain's information-processing capacity is thought to emerge from networks of such neurons.
- Artificial Neural Networks
  - Connectionism, parallel distributed processing etc.
  - Some of the earliest AI work aimed to create artificial neural networks.

A simple mathematical model of a neuron



- The unit's output activation is

$$o(x_1, \ldots, x_n) = \begin{cases} 1 & \text{if } w_0 + w_1 x_1 + \cdots + w_n x_n > 0 \\ -1 & \text{otherwise.} \end{cases}$$

$$o(\vec{x}) = \begin{cases} 1 & \text{if } \vec{w} \cdot \vec{x} > 0 \\ -1 & \text{otherwise.} \end{cases}$$

w_j is the weight on the link from unit j to an output unit.

Units in neural networks

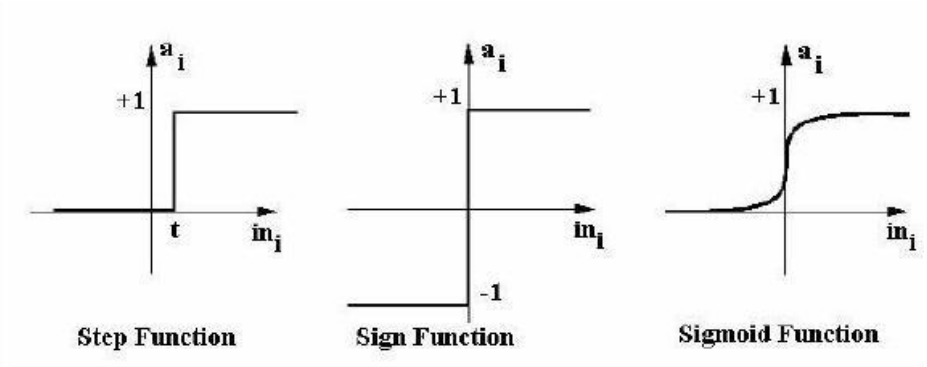
- Neural networks are composed of nodes and directed links.
- Link
  - A link from input unit *j* to output unit *i* serves to propagate the activation x_j from unit *j* to unit *i*.
  - Each link also has a numeric weight $w_{j,i}$ which determines the strength.
- Weighted sum
  - Each unit *i* computes a weighted sum of its inputs:

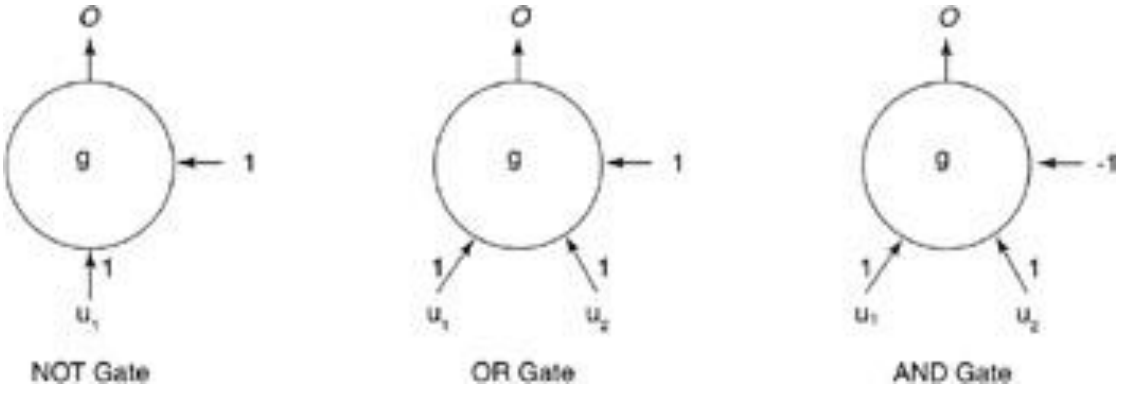
$$in_i = \sum_{j=0}^{n} w_{j,i} x_j$$

Activation function

- Activation function is used to derive the output.

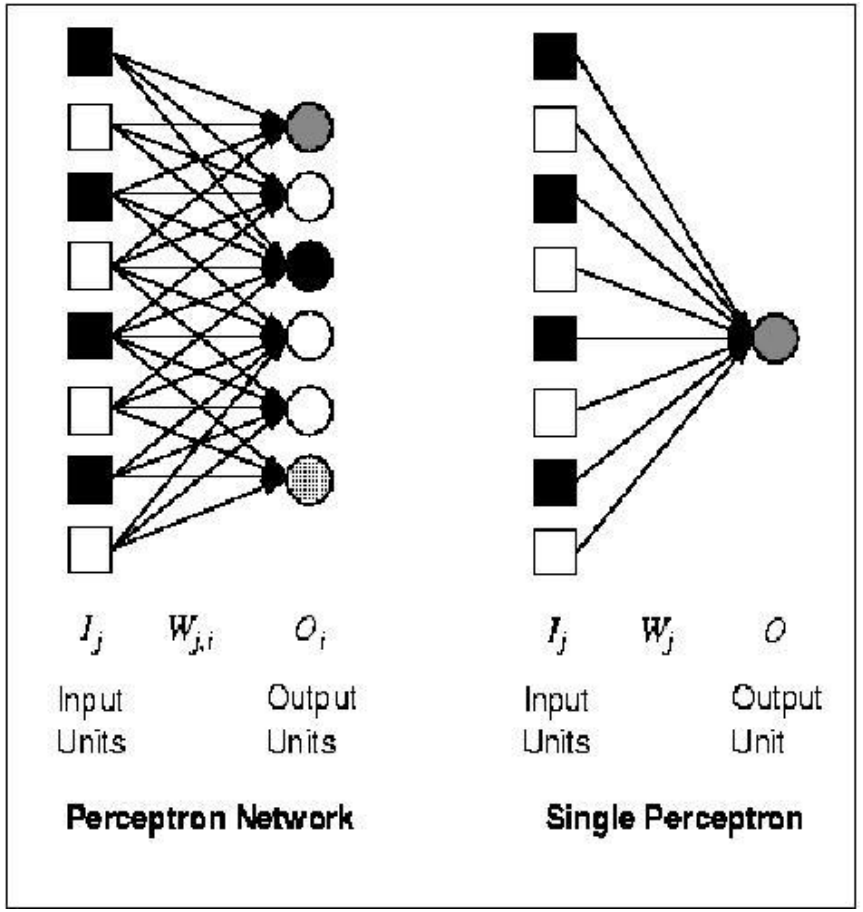
$$o_i = g(in_i) = g(\sum_{j=0}^{n} w_{j,i} x_j)$$





Perceptrons (1/2)

- Single layer feed-forward neural networks
  - A feed-forward neural network: it has no internal state other than the weights themselves.
  - Feed-forward networks are usually arranged in layers, such that each unit receives input only from units in the immediately preceding layer.



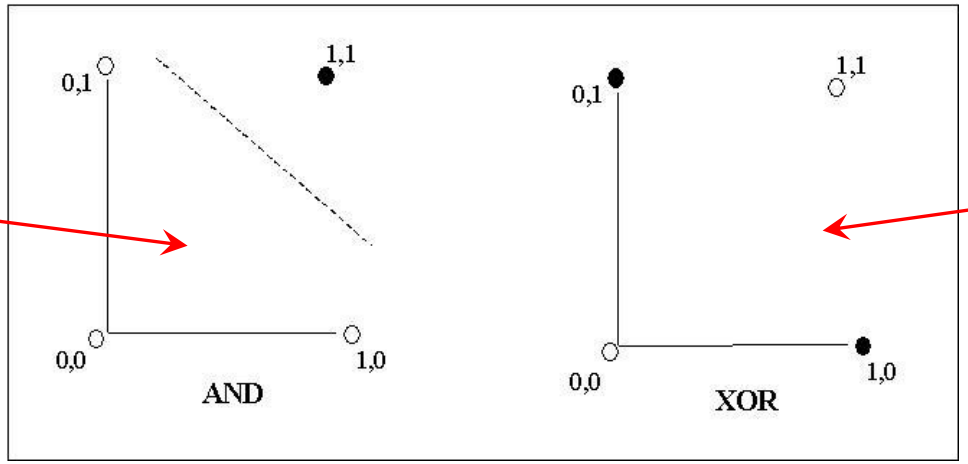
Perceptrons (2/2)

- Hypothesis space of a perceptron (representation power of a perceptron)
- The case where the threshold perceptron return 1

$$\sum_{j=0}^{n} w_j x_j > 0 \quad \text{or} \quad \vec{w} \bullet \vec{x} > 0$$

- $W \bullet X = 0$ defines a hyperplane in the input space.
  - A perceptron returns 1 if and only if the input is on one side of that hyperplane → ***linear separator***

Linearly separable function



Not Linearly separable function

Learning algorithm (1/4)

- Basic idea
  - To adjust the weights of the network to minimize some measure of the error on the training set.

$$w_i \leftarrow w_i + \Delta w_i \qquad \Delta w_i = \eta(t - o)a_i$$

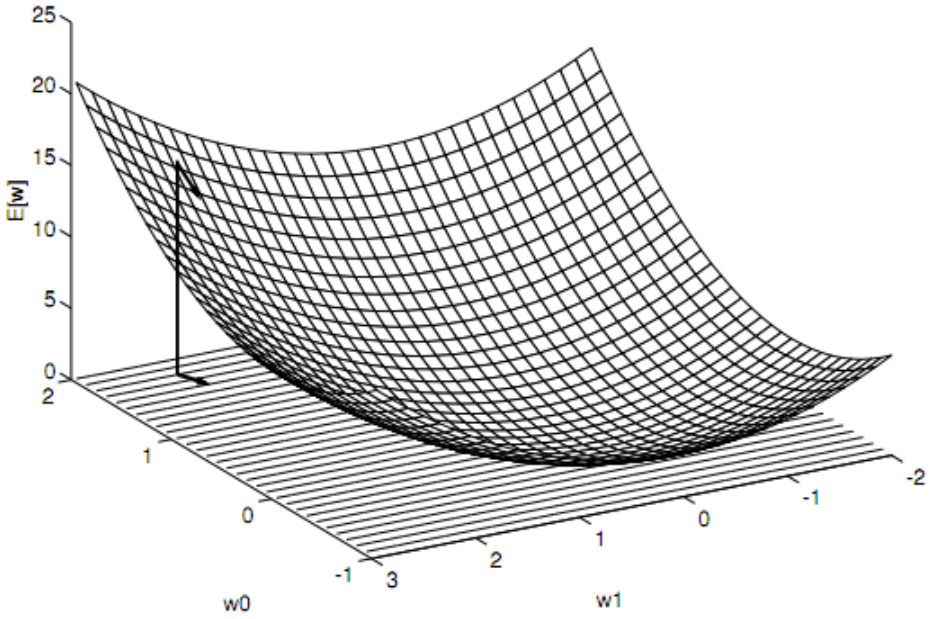
Target value

  - t: target value
  - o: output of the network
  - η: learning rate
- Classical measure of error
  - Sum of squared error

Output of the network

$$E = \frac{1}{2} Err^2 = \frac{1}{2}(t_d - o_d)^2$$

Learning algorithm (2/4)
- Gradient descent rule



- Gradient
  - To calculate the direction of steepest descent along the error space.

$$\nabla E[\vec{w}] \equiv \left[\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \cdots \frac{\partial E}{\partial w_n}\right]$$

$$\vec{w} \leftarrow \vec{w} + \Delta\vec{w}$$

$$\Delta\vec{w} = -\eta \nabla E(\vec{w})$$

$$w_i \leftarrow w_i + \Delta w_i$$

$$\Delta w_i = -\eta \frac{\partial E}{\partial w_i}$$

  - Negative sign is present in order to order move the weight vector in the direction that *decrease E*.

Learning algorithm (3/4)
- Gradient descent rule

$$
\begin{aligned}
\frac{\partial E}{\partial w_i} &= \frac{\partial}{\partial w_i}\frac{1}{2}\sum_d (t_d - o_d)^2 \\
&= \frac{1}{2}\sum_d \frac{\partial}{\partial w_i}(t_d - o_d)^2 \\
&= \frac{1}{2}\sum_d 2(t_d - o_d)\frac{\partial}{\partial w_i}(t_d - o_d) \\
&= \sum_d (t_d - o_d)\frac{\partial}{\partial w_i}(t_d - \vec{w}\cdot\vec{x_d}) \\
\frac{\partial E}{\partial w_i} &= \sum_d (t_d - o_d)(-x_{i,d})
\end{aligned}
$$

$$\Delta w_i = \eta \sum_{d \in D} (t_d - o_d) x_{id}$$

D: training data set

Learning algorithm (4/4) - Gradient descent rule

GRADIENT-DESCENT($training\_examples, \eta$)

Each training example is a pair of the form $\langle \vec{x}, t \rangle$*, where* $\vec{x}$ *is the vector of input values, and* t *is the target output value.* η *is the learning rate (e.g., .05).*

- Initialize each w_i to some small random value
- Until the termination condition is met, Do
  - Initialize each Δw_i to zero.
  - For each $\langle \vec{x}, t \rangle$ in $training\_examples$, Do
    - Input the instance $\vec{x}$ to the unit and compute the output o
    - For each linear unit weight w_i, Do

$$\Delta w_i \leftarrow \Delta w_i + \eta(t - o)x_i$$

  - For each linear unit weight w_i, Do

$$w_i \leftarrow w_i + \Delta w_i$$

Exercise

http://lcn.epfl.ch/tutorial/english/perceptron/html/index.html

- Observe the behavior of the Perceptron in the following cases
  - Case 1: y = [0 0 0 0]
  - Case 2: y = [1 0 0 0]
  - Case 3: y = [0 1 1 0]
  - Case 4: y = [0 0 0 1]
  - Case 5: y = [1 0 0 1]
  - Case 6: the effect of learning rate (change the learning rate at your will and observe the result)

Homework

- Due: Sep. 22nd
- Hardcopy submission to TA
- Problem1
  - Can the Perceptron solve the following classification task? **Answer** the question and **prove your answer** (hint: consider contents in slide 8):

$$Patterns\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 2 & 1 & 3 \\ 4 & 2 & 6 \end{bmatrix} \quad Outputs = [0 \quad 1 \quad 1]$$

- Problem 2
  - Explain the effect of learning rate using the following applet.
  - http://freeisms.com/PerceptronAppletItself.html
  - Experiment with learning rates of ***0.06, 0.30, 0.6, 1.0*** and discuss the effect of learning rate using the "error view."
  - 1 page