Artificial Intelligence

CSE 440

Chapter 18

Fall 2017

Mirza Mohammad Lutfe Elahi
Department of Electrical and Computer Engineering
North South University

Biological inspiration

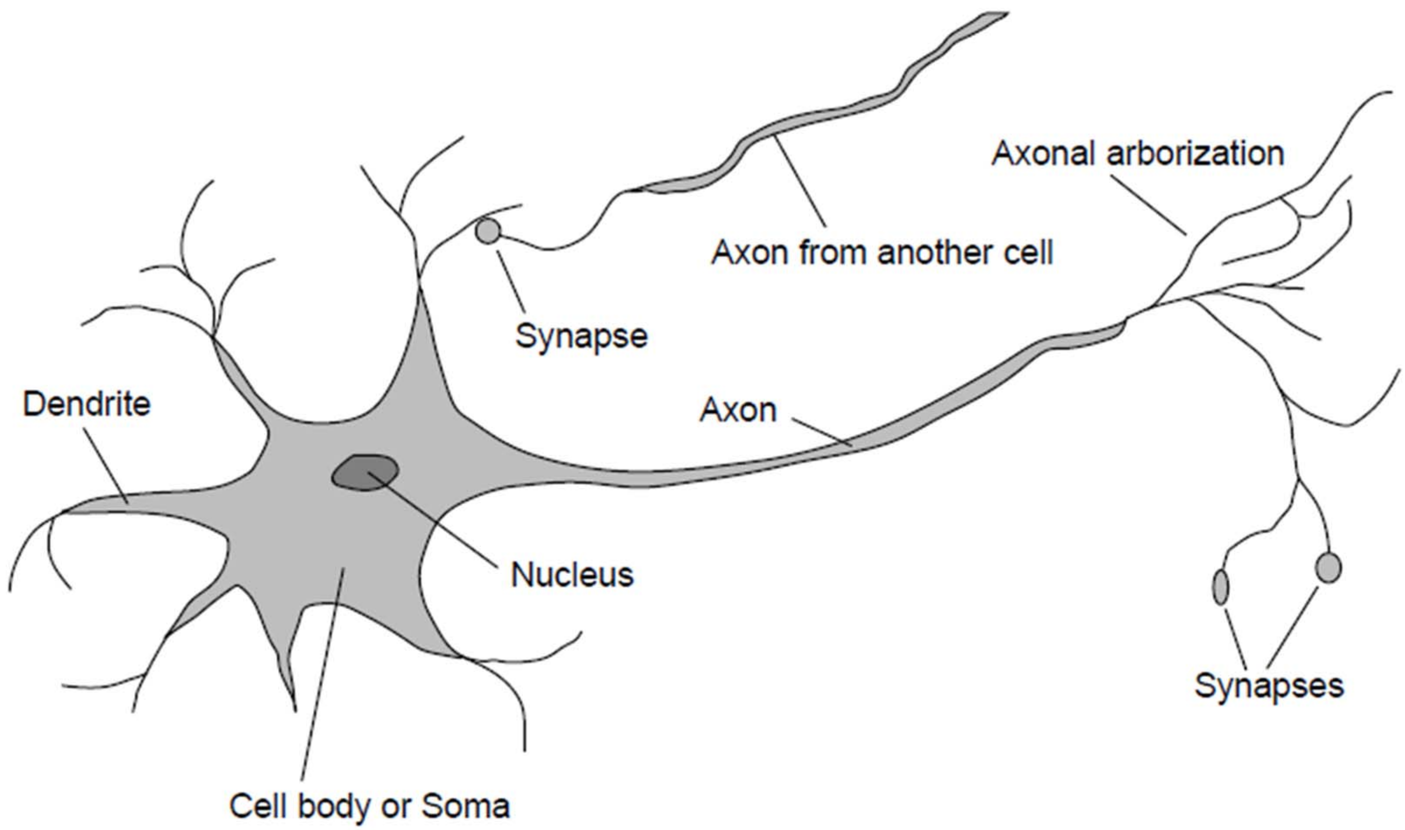
- Animals are able to react adaptively to changes in their external and internal environment, and they use their nervous system to perform these behaviours.

- An appropriate model/simulation of the nervous system should be able to produce similar responses and behaviours in artificial systems.

- The nervous system is build by relatively simple units, the neurons, so copying their behaviour and functionality should be the solution.

Biological inspiration

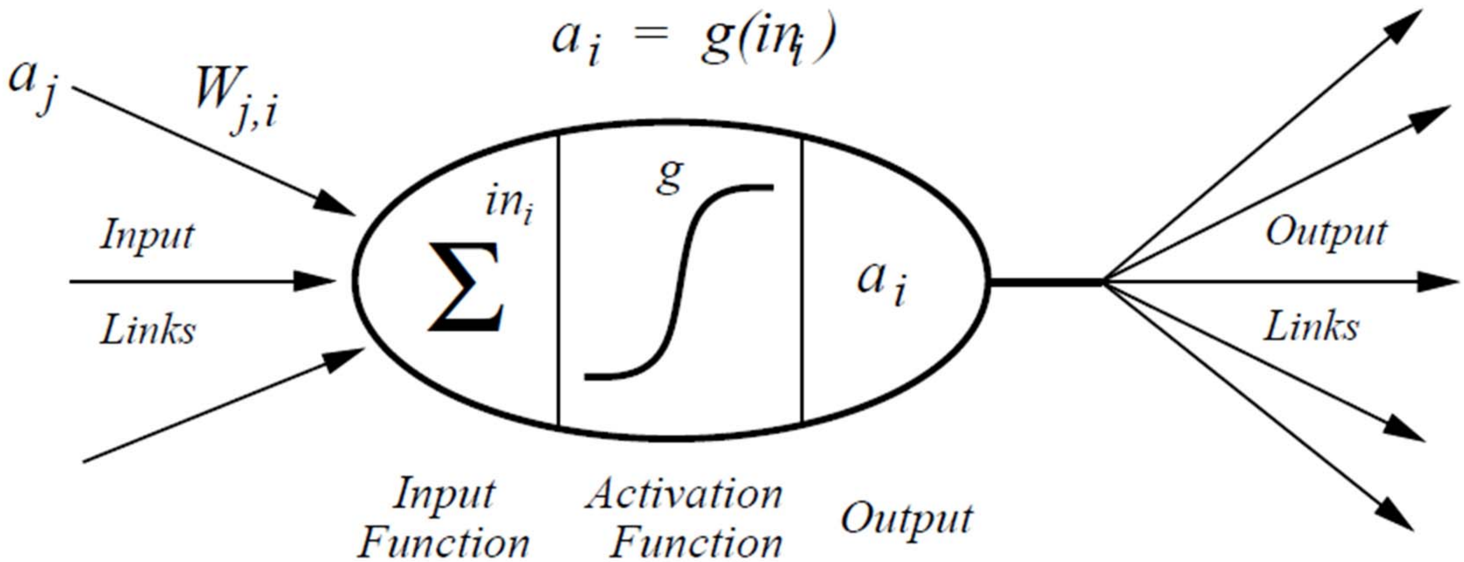
- 10^{11} neurons of > 20 types, 1014 synapses, 1ms–10ms cycle time
- Signals are noisy “spike trains” of electrical potential



Neural Networks

- A neural network consists of a set of nodes (neurons/units) connected by links
  - Each link has a numeric weight
- Each unit has:
  - A set of input links from other units
  - A set of output links to other units
  - A current activation level
  - An activation function to compute the activation level in the next time step

Basic definitions



- A gross oversimplification of real neurons, but its purpose is to develop understanding of what networks of simple units can do

Basic definitions

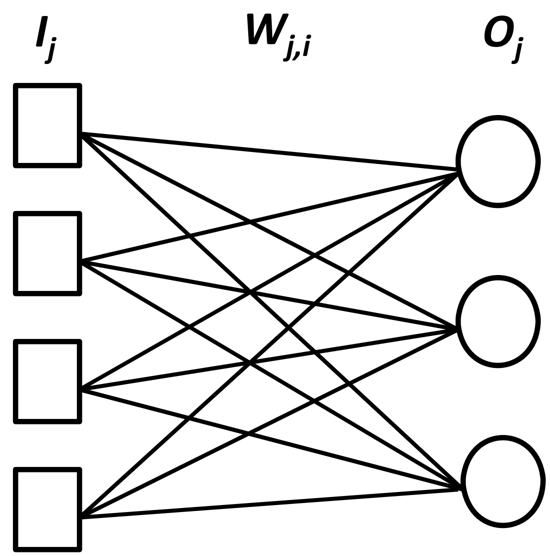
- The total weighted input is the sum of the input activations times their respective weights:

$$in_i = \sum_j W_{j,i}\, a_j$$

In each step we compute:

$$a_i \leftarrow g(in_i) = g(\sum_j W_{j,i}\, a_j)$$

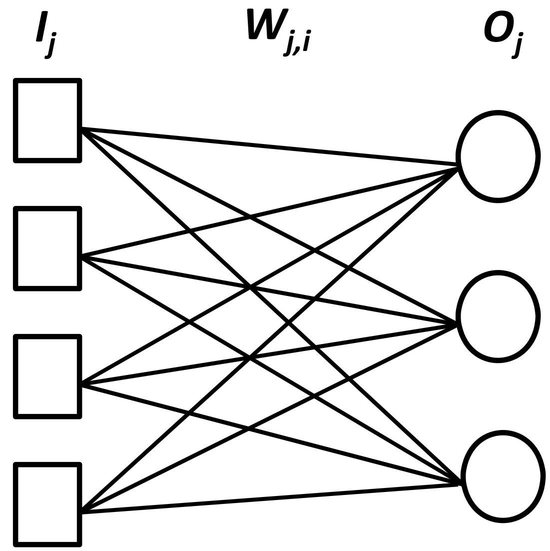
Learning in single layer network



- If the output for an output unit is O, and the correct output should be T, then the error is given by:

$$Err = T - O$$

Learning in single layer network



- The weight adjustment rule is:

$$W_j \leftarrow W_j + \alpha \text{ X } I_j \text{ X Err}$$

where α is a constant called the learning rate

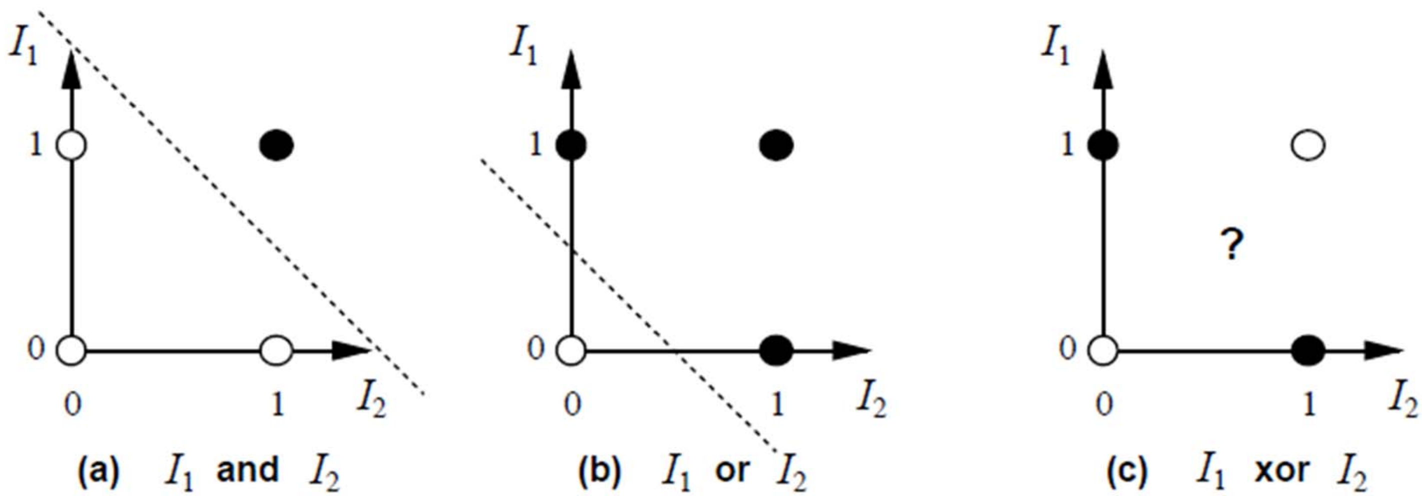
Learning Network Structures

- Different functions are implemented by different network topologies and unit **weights**.
- The lure of NNs is that a network need not be explicitly programmed to compute a certain function *f*.
- Given enough nodes and links, a NN can learn the function by itself.
- It does so by looking at a training set of input/output pairs for f and modifying its topology and weights so that its own input/output behavior agrees with the training pairs.
- In other words, NNs too learn by induction.

Computing with NNs

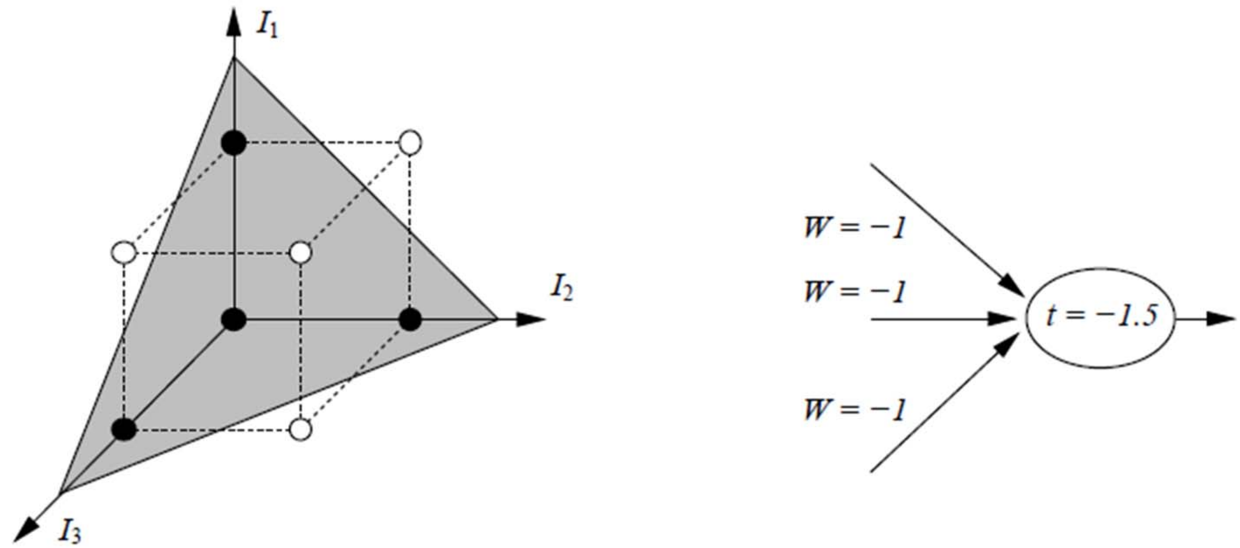
- The structure of a NN is given by its nodes and links.
- The type of function a network can represent depends on the network structure.
- Fixing the network structure in advance can make the task of learning a certain function impossible.
- On the other hand, using a large network is also potentially problematic.
- If a network has too many parameters (ie, weights), it will simply learn the examples by memorizing them in its weights (overfitting).

Linearly Separable Functions on a 2-dimensional Space



- A black dot corresponds to an output value of 1. An empty dot corresponds to an output value of 0.

Linearly Separable Functions on a 3-dimensional Space



(a) Separating plane

(b) Weights and threshold

- The minority function: Return 1 if the input vector contains less ones than zeros. Return 0 otherwise.

Learning Steps

Learn by adjusting weights to reduce error on training set

The squared error for an example with input $\mathbf{x}$ and true output y is

$$E = \frac{1}{2}Err^2 \equiv \frac{1}{2}(y - h_{\mathbf{W}}(\mathbf{x}))^2 \ ,$$

Perform optimization search by gradient descent:

$$\begin{aligned}\frac{\partial E}{\partial W_j} &= Err \times \frac{\partial Err}{\partial W_j} = Err \times \frac{\partial}{\partial W_j}\left(y - g(\Sigma_{j=0}^{n} W_j x_j)\right) \\ &= -Err \times g'(in) \times x_j\end{aligned}$$

Simple weight update rule:

$$W_j \leftarrow W_j + \alpha \times Err \times g'(in) \times x_j$$

E.g., +ve error $\Rightarrow$ increase network output
$\Rightarrow$ increase weights on +ve inputs, decrease on -ve inputs