# An Introduction to Neural Networks

Kristian Guillaumier 2006, 07

# Main References

- Wikipedia Article.
- NeuralNetworkSolutions.com
  - http://www.neuralnetworksolutions.com/resources.php

# Introduction

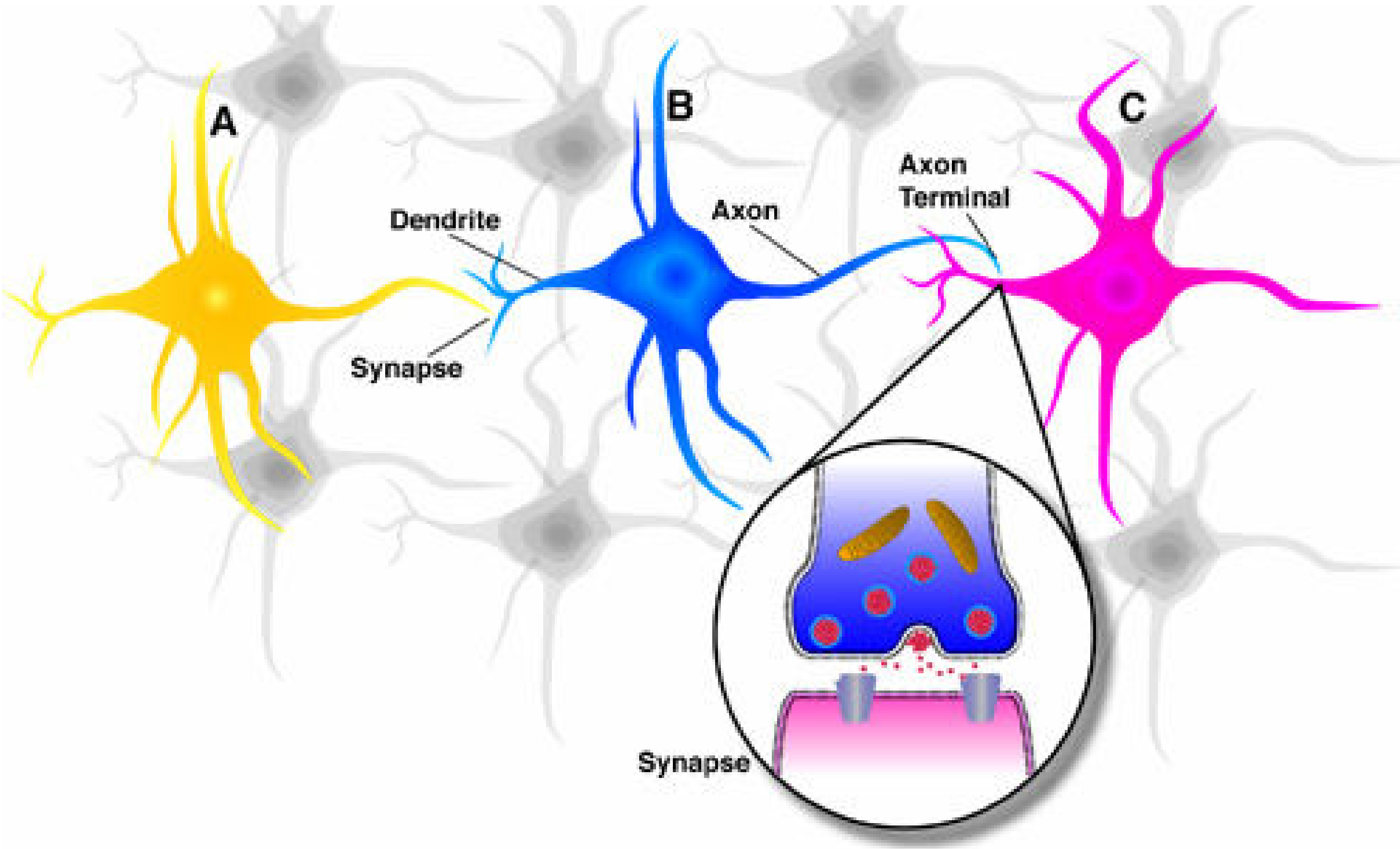
- Modeled after the structures in the brain.
- Highly parallel – unlike traditional sequential computing models.
- Interconnected processing elements called nodes or neurons (brain components).
- Applicable to:
  - Pattern recognition (e.g. OCR).
  - Trend prediction.
  - Etc…

# Motivation

- Rigid instruction processing is useful for cased where a problem can be described by a sequence of well defined steps: an algorithm.
- Other problems are very hard to describe using such a model because these steps might not be properly defined – the algorithm must be known.
- Neural networks can generalise and can be used in cases where no clear algorithm exists or data is noisy.

# Biological Ideas

(image http://chaos.mind.ilstu.edu/curriculum2/neuro/neuron_1.html)

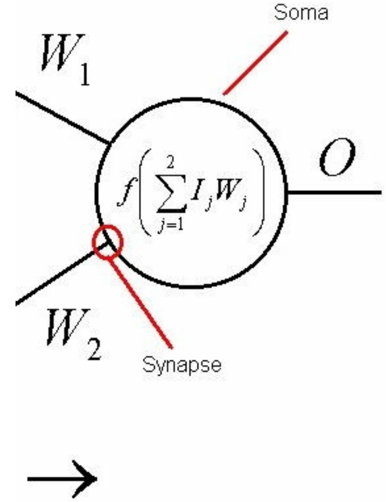


# Biological Ideas

- Neurons receive inputs from neighbours.
- Signal transfer via an electrochemical synapse (cell-to-cell junction).
- Signal is carried along the dendrite into the neuron.
- If some threshold (electrical gate, condition) is met, an output is produced and transmitted via the axon (propagate the signal).

# The Threshold Logic Unit

- Proposed by McCulloch and Pitts.
- A number of input signals are transmitted to the artificial neuron.
- Each neural input has a weight.
- The effect of these signals is based on their input values and their weights.
- Consider a input neuron.
  - Inputs = (1,1,0).
  - Weights = (1,-0.5,1).
  - Result = (1*1 + 1*-0.5 + 0*1).
  - Result = 0.5.
- If result exceeds a threshold, output 1 else output 0.
  - Sigmoid/Gate/Hyperbolic Tangent…

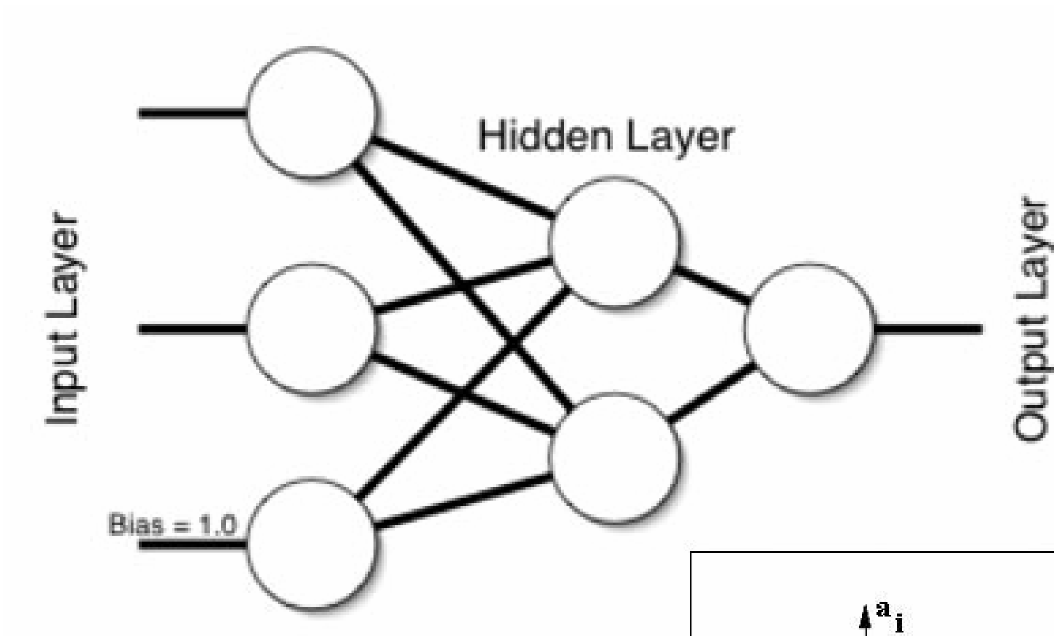


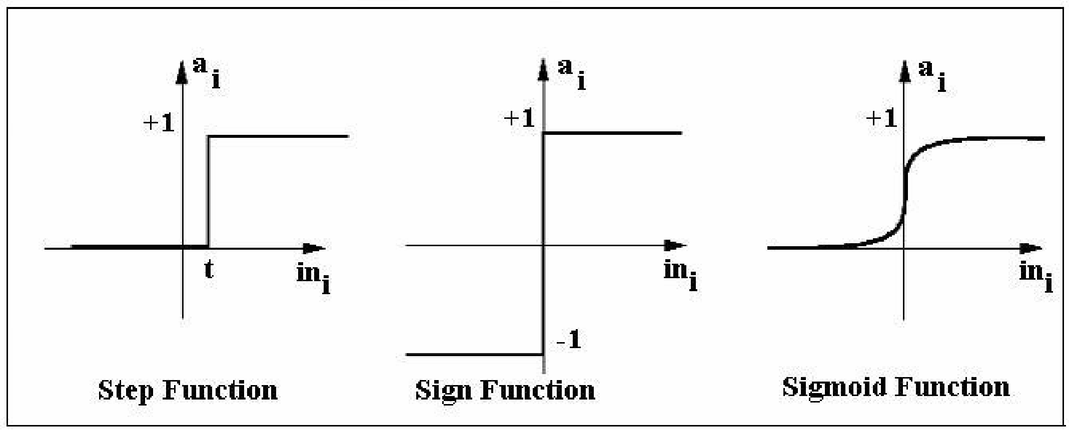

$$a = w_1 x_1 + w_2 x_2 + \ldots + w_n x_n$$

$$a = \sum_{i=1}^{n} w_i x_i$$

# General Structure, Sigmoid Function

(http://www.oreillynet.com/pub/a/mac/2005/10/21/artificial-intelligence.html?page=2)
(http://newsroom.spie.org/Images/Imported/18_256_0_2006-01-25/18_fig3.jpg)



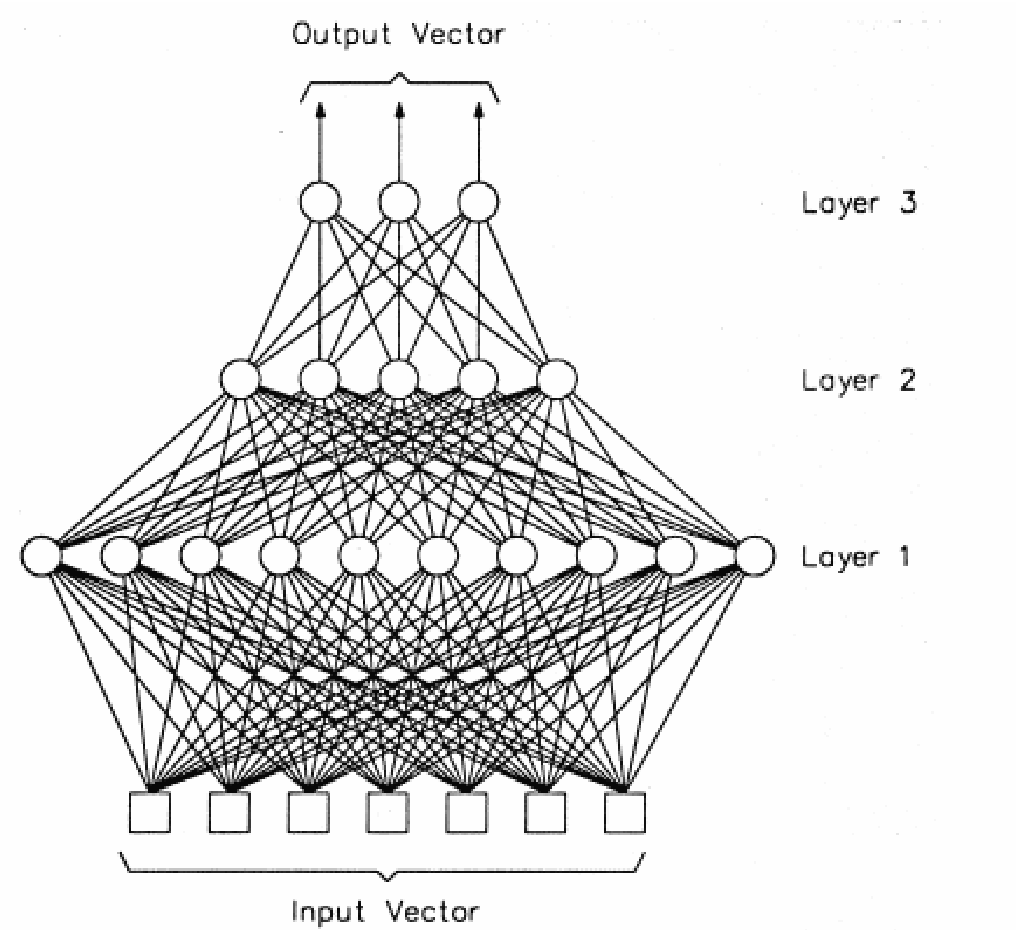



# Connectivity

- Early attempts were random connections – they failed.
- Most applications of neural nets use 3 layers:
  - An input layer (from sensors).
  - A hidden layer (there can be many of these).
  - An output layer.
- Inputs and outputs are usually labeled as ‘vectors’.
- “Feed Forward”-ness (no loops).

# Connectivity

**(http://www.vias.org/tmdatanaleng/cc_ann_bp_function.html)**



# Supervised Learning

- ANNs can be used for both supervised and unsupervised learning:
- Supervised:
  - Machine learning technique based on training sets.
  - +ve and –ve training data.
  - Data may be noisy.
  - Data may classification or produce a value.
  - The algorithm must learn to generalise from the data to handle unforseen examples – depends on an inductive bias (more on this later).

# Learning an AND Gate

- Consider a single TLU with 2 inputs, 1 output.
- We want to learn the AND gate.
- Set weights at random.
- Input is a binary input vector (length = 2).
- Output is a bit.
- To learn we must:
  - Be able to measure error.
  - Adjust weights to minimise the error.
- Error is calculated by measuring the difference between the actual output produced and the target output desired.
- The weights are modified depending on (proportional to) the gravity of the error.
- Compensation is +v or –ve depending on whether the actual is less than or greater then the output.

# Learning an AND Gate

- Simple method:

$$\Delta w_i = \alpha(t - y)_i x_i$$

w = weight, α = constant, x = signal,

t = target output, y = actual output

- *Ex. X1= 1, X2=1, Y=0, T=1:*

$$\Delta w_1 = \alpha(1-0)_1 \cdot 1 = +\alpha$$

$$\Delta w_2 = \alpha(1-0)_2 \cdot 1 = +\alpha$$

- *Ex. X1=0, X2=1, Y=1, T=0:*

$$\Delta w_1 = \alpha(0-1)_1 \cdot 0 = 0$$

$$\Delta w_2 = \alpha(0-1)_2 \cdot 1 = -\alpha$$

# The XOR Function

- Minsky and Papert demonstrated that a number of functions cannot be computed by a single or double (input/output) layer ANN.
- An example is the XOR function.
- Consider what the inequalities required to compute the XOR function would have to be (they contradict themselves).
- Compare this with the inequalities of AND.

$$0w_1 + 0w_2 < \theta \rightarrow 0 < \theta$$

$$0w_1 + 1w_2 > \theta \rightarrow w_2 > \theta$$

$$1w_1 + 0w_2 > \theta \rightarrow w_1 > \theta$$

$$1w_1 + 1w_2 < \theta \rightarrow w_1 + w_2 < \theta$$

# Inductive Bias

- Read Wikipedia article for a good informal explanation (http://en.wikipedia.org/wiki/Inductive_bias).
- Consider examples from a language:

  ab

  aabb

  aaabbb

  aaaabbbb
- In general, is the language $a^n b^n$ where $n > 1$?
- What assumptions have to made?
- These assumptions are the inductive bias of the learner.
- Occam’s Razor.

# The Hidden Layer

- Hidden layers make the ANN more flexible.
- Research shows that the model we considered so far do not work properly with hidden layers.
- Refinements on the ANN to use hidden layers include:
  - The delta rule.
  - The back propagation algorithm.

# Applications of ANNs

- Stock prediction.
- Currencies.
- Optical character recognition.
- Protein sequence recognition.
- Weather forecasting.
- For more details see http://www.neuralnetworksolutions.com/nn/applications1.php.

# Final Thoughts

**(from http://www.tfhrc.gov/safety/98133/ch02/body_ch02_05.html)**

- Advantages:
  - No need to assume distribution (e.g. random distributions) as us usually required in statistical analysis.
  - Applicable to multi-variable non-linear problems.
  - Learning is mostly automated.
  - Exact understanding of an algorithm is not required.
- Disadvantages:
  - To minimise the possibility of over-training a lot of computational effort is required.
  - BLACK BOX.
  - Large training sets required.

# OCR: Example

**(image http://www.nd.com/welcome/whatisnn.htm)**

- See http://www.codeproject.com/csharp/Neural_Network_OCR.asp?df=100&forumid=206868&exp=0&select=1906515 for an example in C#.