LAU
الجامعة اللبنانية الأميركية
Lebanese American University

CSC 447: Parallel Programming for Multi-Core and Cluster Systems

Neural Networks

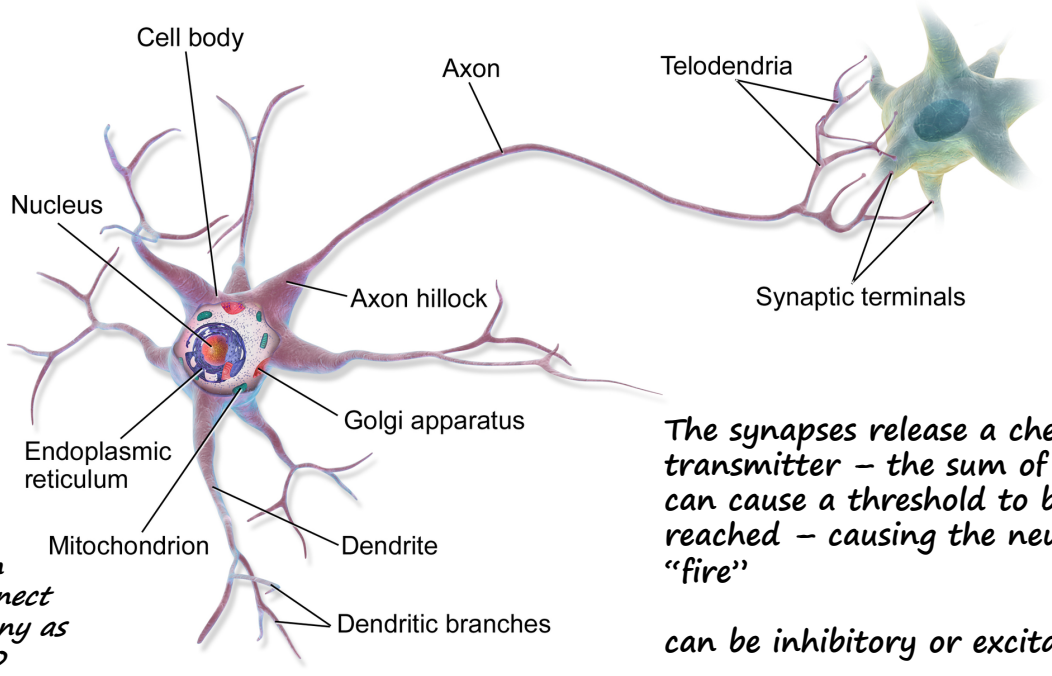
What are Neural Networks?

- Neural Networks (NNs) are networks of neurons, for example, as found in real (i.e. biological) brains.
- Artificial Neurons are crude approximations of the neurons found in brains. They may be physical devices, or purely mathematical constructs.
- Artificial Neural Networks (ANNs) are networks of Artificial Neurons, and hence constitute crude approximations to parts of real brains. They may be physical devices, or simulated on conventional computers.
- From a practical point of view, an ANN is just a parallel computational system consisting of many simple processing elements connected together in a specific way in order to perform a particular task.
- One should never lose sight of how crude the approximations are, and how over-simplified our ANNs are compared to real brains.

Neural Networks

- McCulloch & Pitts (1943) are generally recognised as the designers of the first neural network
  - Binary Neuron
  - Combining many simple units to get increased computational power
  - The idea of a threshold)
- Hebb (1949) developed the first learning rule (on the premise that if two neurons were active at the same time the strength between them should be increased)
- During the 50's and 60's many researchers worked on the perceptron amidst great excitement.
- 1969 saw the death of neural network research for about 15 years
  - The AI Winteer
  - See Minsky & Papert

Neural Networks

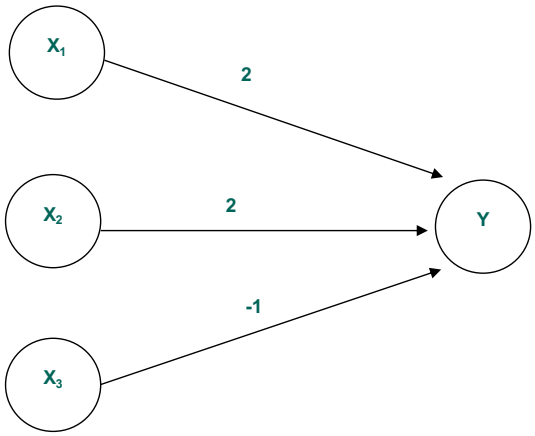


A neuron may connect to as many as 100,000 other neurons

The synapses release a chemical transmitter – the sum of which can cause a threshold to be reached – causing the neuron to "fire"

can be inhibitory or excitatory

The First Neural Neural Networks

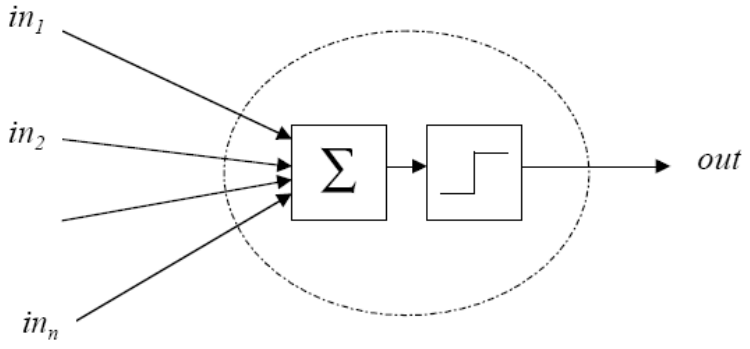


Neurons in a McCulloch-Pitts network are connected by directed, weighted paths

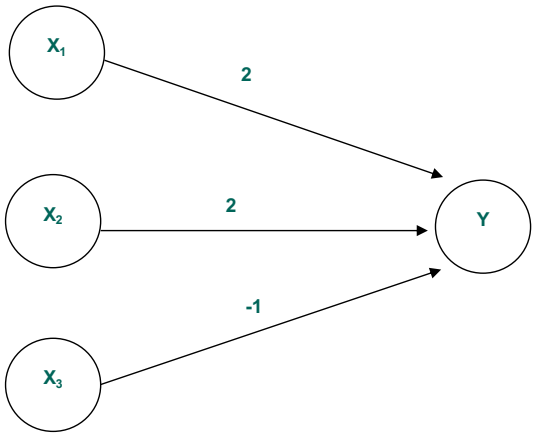
If the weight on a path is positive the path is excitatory, otherwise it is inhibitory

The McCulloch-Pitts Neuron

- This vastly simplified model of real neurons is also known as a Threshold Logic Unit :
  - A set of synapses (i.e. connections) brings in activations from other neurons.
  - A processing unit sums the inputs, and then applies a non-linear activation function (i.e. squashing/transfer/threshold function).
  - An output line transmits the result to other neurons.

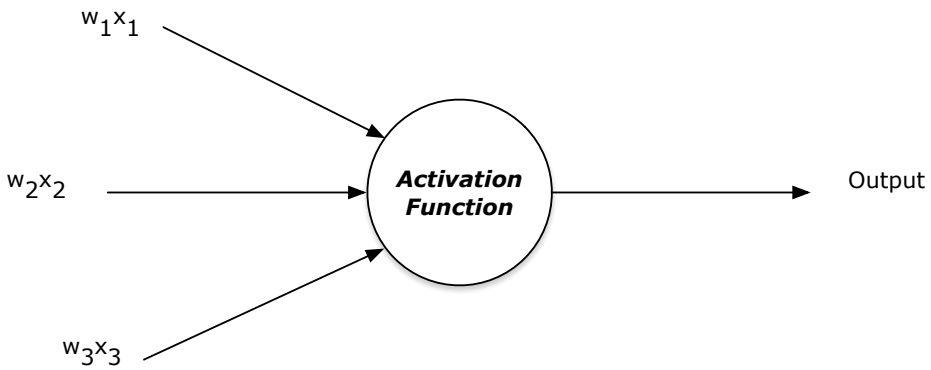


The First Neural Neural Networks



All excitatory connections into a particular neuron have the same weight, although different weighted connections can be input to different neurons

Perceptron



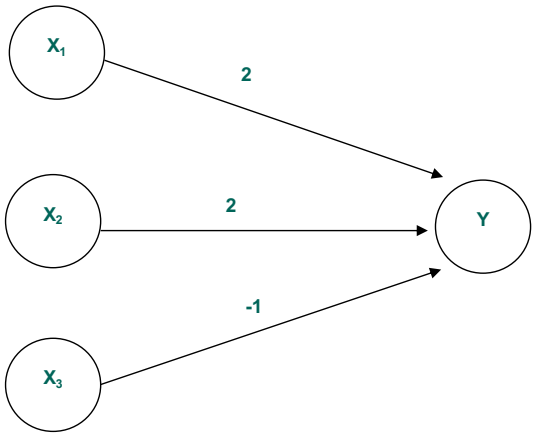
We can connect any number of McCulloch-Pitts neurons together in any way we like.

An arrangement of one input layer of McCulloch-Pitts neurons feeding forward to one output layer of McCulloch-Pitts neurons is known as a Perceptron.

Activation Functions are applied to the inputs at each neuron

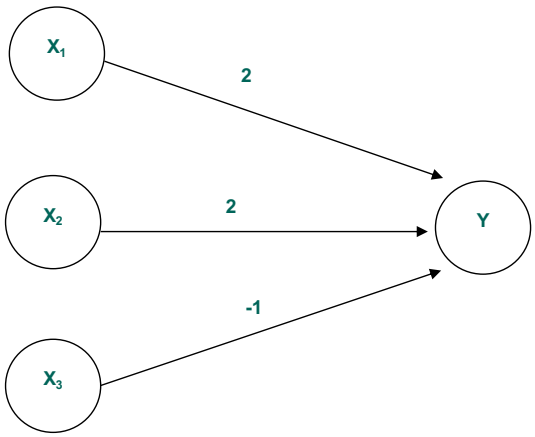
The activation of a neuron is binary. That is, the neuron either fires (activation of one) or does not fire (activation of zero).

The First Neural Neural Networks



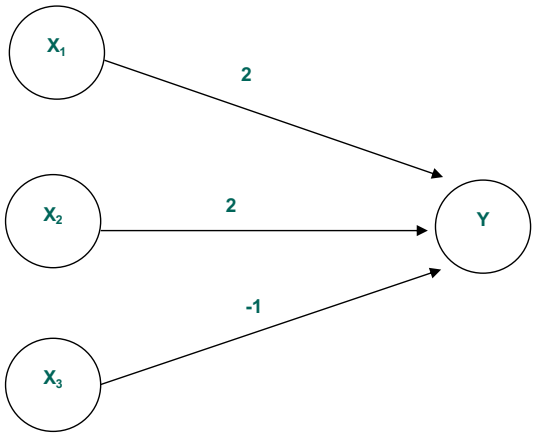
Each neuron has a fixed threshold. If the net input into the neuron is greater than or equal to the threshold, the neuron fires

The First Neural Neural Networks



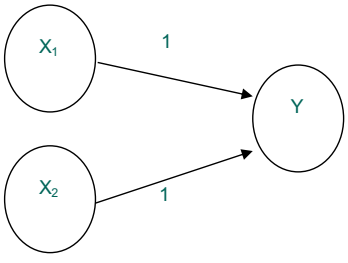
The threshold is set such that any non-zero inhibitory input will prevent the neuron from firing

The First Neural Neural Networks



It takes one time step for a signal to pass over one connection.

The First Neural Neural Networks

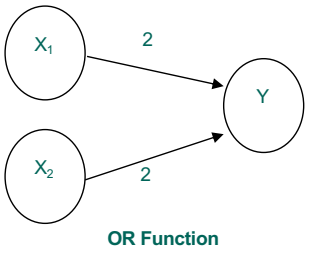


AND Function

AND		
X1	X2	Y
1	1	1
1	0	0
0	1	0
0	0	0

Threshold(*Y*) = 2

The First Neural Neural Networks

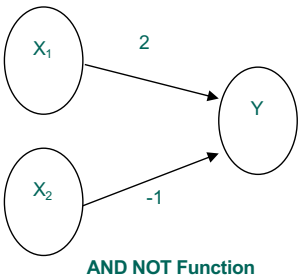


OR Function

OR		
X1	X2	Y
1	1	1
1	0	1
0	1	1
0	0	0

Threshold(*Y*) = 2

The First Neural Neural Networks

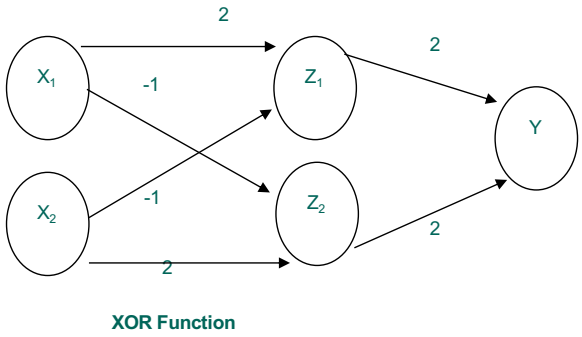


AND NOT Function

AND NOT		
X1	X2	Y
1	1	0
1	0	1
0	1	0
0	0	0

Threshold(*Y*) = 2

The First Neural Neural Networks

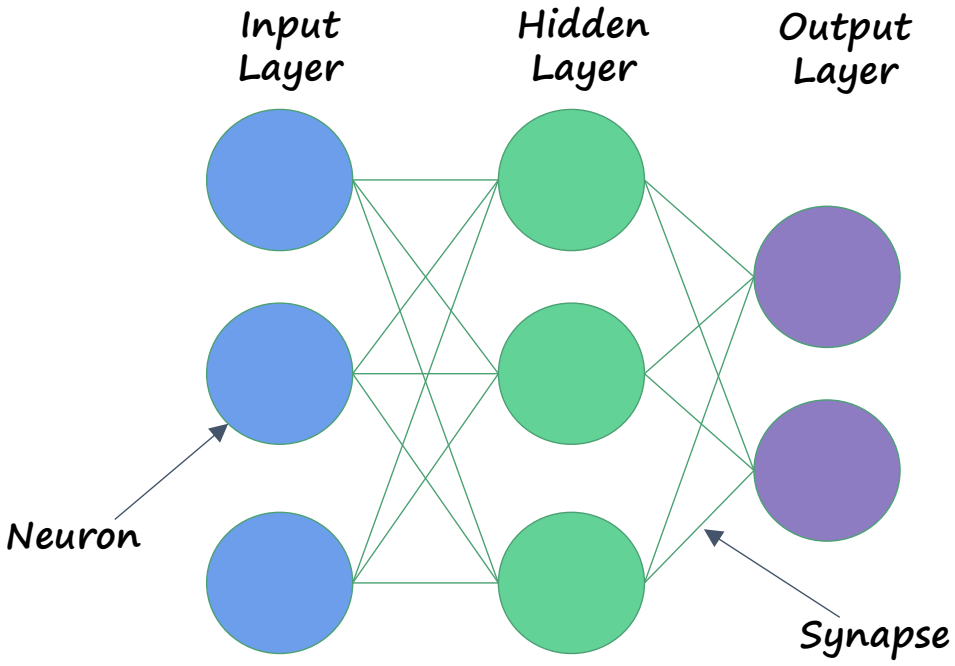


XOR Function

XOR		
X1	X2	Y
1	1	0
1	0	1
0	1	1
0	0	0

X_1 XOR X_2 = (X_1 AND NOT X_2) OR (X_2 AND NOT X_1)

Deep Neural Network Architecture



Deep Neural Network Architecture

```
import numpy as np # import Numpy library to generate

weights = np.around(np.random.uniform(size=6), decimals=2) # initialize the weights
biases  = np.around(np.random.uniform(size=3), decimals=2) # initialize the biases

x_1 = 0.5 # input 1
x_2 = 0.85 # input 2

z_11 = x_1 * weights[0] + x_2 * weights[1] + biases[0]
z_12 = x_1 * weights[2] + x_2 * weights[3] + biases[1]

a_11 = 1.0 / (1.0 + np.exp(-z_11))
a_12 = 1.0 /(1.0 + np.exp(-z_12))

z_2 = weights[5] * a_12 + weights[4] * a_11 + biases[2]

a_2 = 1.0 /(1+np.exp(-z_2))

print('The output of the network for x1 = 0.5 and x2 = 0.85 is {}'.format(np.around(a_2, decimals=4)))
```