

Artificial Neuron Model and General Network Structures

Andrzej Kordecki

Neural Networks (ML.ANK385 and ML.EM05): Lecture 02

Division of Theory of Machines and Robots
Institute of Aeronautics and Applied Mechanics
Faculty of Power and Aeronautical Engineering
Warsaw University of Technology

Table of Contents

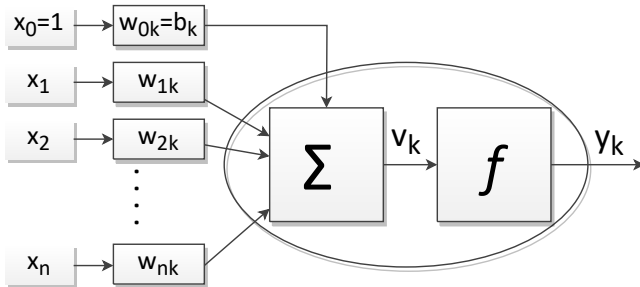
- 1 Artificial Neuron
 - Artificial Neuron
 - Activation Functions
 - Perceptron
 - Linear Classification
 - Exclusive OR

- 2 Neural Networks
 - Neural Networks
 - Network Architectures

Artificial Neuron

Artificial Neuron

A neuron is an information-processing unit that is fundamental to the operation of a neural network.



Input:

$$u_k = \sum_{i=1}^n w_{ik} x_i$$
$$v_k = \sum_{i=1}^n w_{ik} x_i + w_{0k}$$

Output:

$$y_k = f(v_k) = f(x_i, w_{ik})$$

Artificial Neuron

$$y = f(v) = f(x, w)$$

Terminology:

- x_i - input signals,
- w_{ik} - weights (parameters) used to scale the i input connected to neuron k ,
- $w_{0k} = b_k$ - bias or activation threshold for input value $x_0 = 1$,
- v_k - effective input of neuron k is the function of the inputs and weights (local field or activation potential),
- f - activation function of the neuron,
- y_k - output of neuron k .

Neuron is a scalar function with multiple arguments.

Artificial Neuron

Neuron properties:

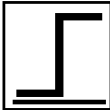
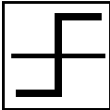
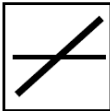


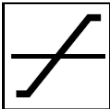

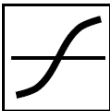
- Multiple inputs and One output,
- Numerical in/out signals,
- Weights are usually are a small number,
- Value range in most cases is limited: $\{0, 1\}$ or $\{-1, 1\}$,
- Weights don't have to be assigned to inputs, though they may and often are.

Activation Functions

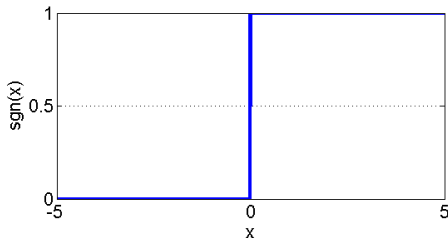
The activation function defines the output of neuron for given an input or set of inputs.

- The type of activation function depends on location and architecture neural network,
- Nonlinear activation functions allow networks to compute solution using only a smaller number of neurons,
- Given network layer (or neuron group) usually have single activation function,
- Often preliminary tests are necessary.

Activation Functions

Unit step		Symmetric unit step	
Linear		Linear positive	
Linear, saturated		Sym. linear, saturated	
Logistic sigmoid		Hyperbolic tangent	

Threshold function

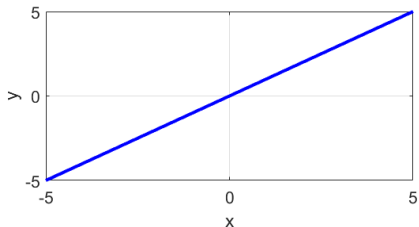


Output:

$$y(v) = \begin{cases} 1 & \text{if } v \geq 0 \\ 0 & \text{if } v < 0 \end{cases}$$

- binary classifier - yes or no,
- ambiguous in case of multiple neurons representing more classes,
- intermediate activation values rather than binary.
- McCulloch-Pitts (Threshold Logic Unit) use threshold function.

Linear function

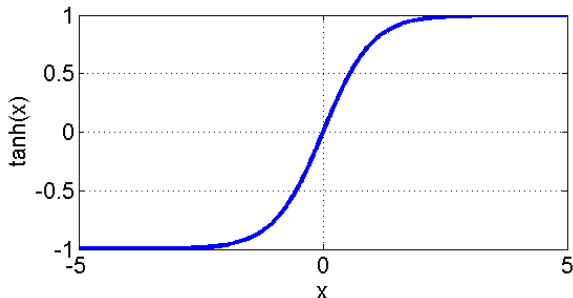


Output:

$$y(v) = av$$

- in case $a = 1$ is identity function,
- error in prediction is constant and not depending on the change x ,
- final activation function of last layer of linear neural network is just a linear function of the input of first layer.

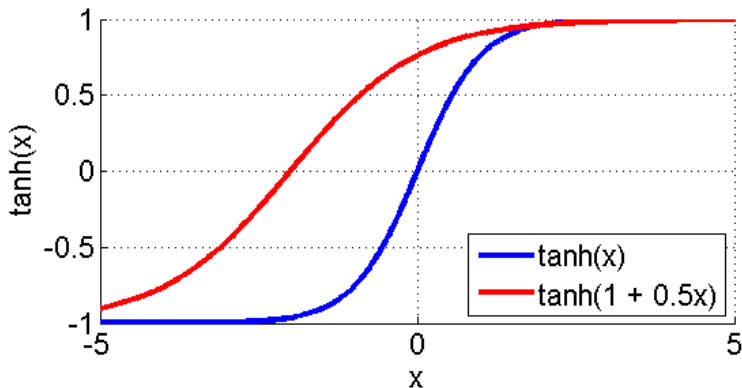
Hyperbolic Tangent function



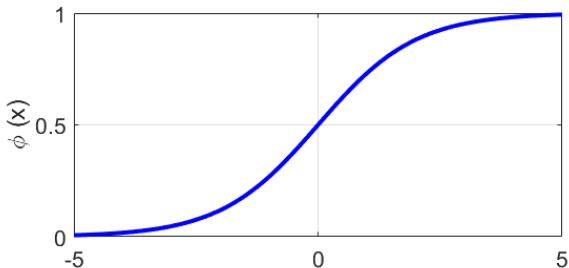
- Function:
$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$
- Output:
$$y(v) = \tanh(v)$$
- sigmoid function ("S" - shaped),
- continuous and differentiable,
- nonlinear and monotonic,
- bounded.

Hyperbolic Tangent function

Example: Weights impact on function shape for $w_0 = 1$ and $w_1 = 0.5$.



Logistic function



- Function:

$$\phi(x) = \frac{1}{1 + e^{-x}}$$

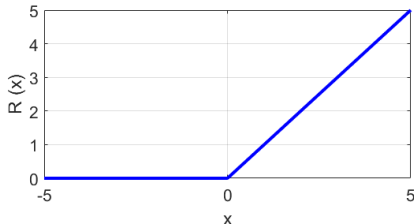
- Output:

$$y(v) = \phi(v)$$

- sigmoid function ("S" - shaped),
- the same properties as tanh function,
- different values range: logistic (0, 1) and tanh (-1, 1).

ReLu function

Rectified linear unit (ReLU)



Output:

$$y = \max(0, v)$$

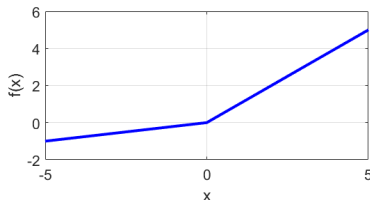
Simple function with many advantages:

- Sparse activation,
- Better gradient propagation,
- Efficient computation.

The most popular activation function for deep neural networks (hidden layers).

Leaky ReLU function

Dying ReLU problem: ReLU neurons can sometimes be pushed into states in which they become inactive for essentially all inputs. It may be mitigated by using leaky ReLUs instead, which assign a small positive slope.

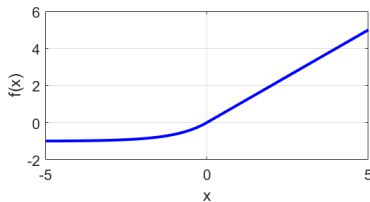


Output:

$$y(v) = \begin{cases} v & \text{if } v \geq 0 \\ 0.01v & \text{if } v < 0 \end{cases}$$

ELU function

The Exponential Linear Unit (ELU) have negative values which allows them to push mean unit activations closer to zero.

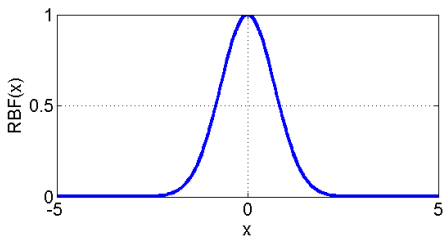


Output:

$$y(v) = \begin{cases} v & \text{if } v \geq 0 \\ \alpha(\exp v - 1) & \text{if } v < 0 \end{cases}$$

If the input value x is less than 0, we get a value slightly below zero (reduce vanishing gradient problem).

Radial Basis Function RBF



- Function:

$$rbf(x) = \sum_{i=1}^N a_i \exp(-b_i \|x - c_i\|^2)$$

$w \in \{a, b, c\}$

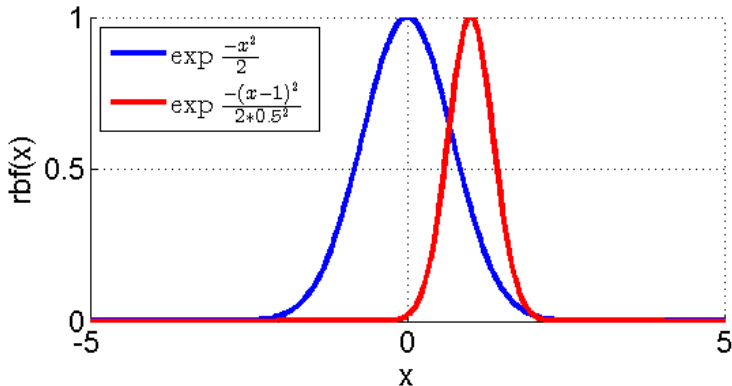
- Output:

$$y = \exp\left(-\frac{1}{2w_{n+1}} \sum_{i=1}^N (x_i - w_i)^2\right)$$

- continuous,
- differentiable,
- non-monotonic,
- bounded,
- local influence of nonlinearity

Radial Basis Function RBF

Example: Weights impact on Gaussian RBF plot for $n = 1$, $w_0 = 1$ and $w_1 = 0.5$.

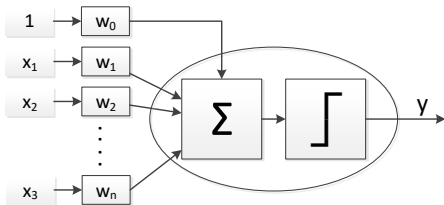


Which one do we use?

- Derivative of activation function define direction and how much to change (update) weights in process of neural network learning,
- Know the problem you are trying to solve, i.e. sigmoid works well for a simple classifier network or ReLu function in Deep Learning,
- Some network types may only work with specific activation functions,
- Customization and testing.

Perceptron

This Perceptron neuron is vastly simplified model of real neurons:



Input:

$$v = w_0 + \sum_{i=1}^n w_i x_i$$

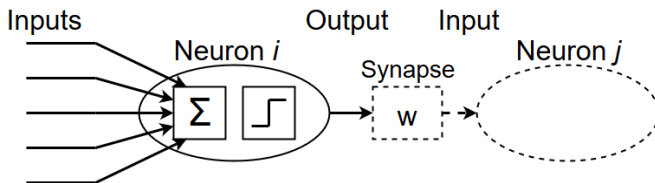
Output:

$$y = f(v)$$

- Binary Signals - input/output,
- Threshold unit step function with threshold value w_0 ,
- An output line transmits the result to other neurons,

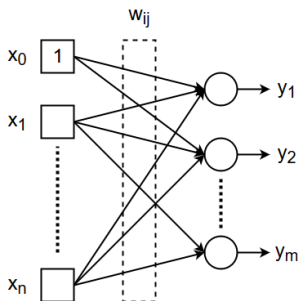
Perceptron

We can connect any number of Perceptron neurons together in any way we like.



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

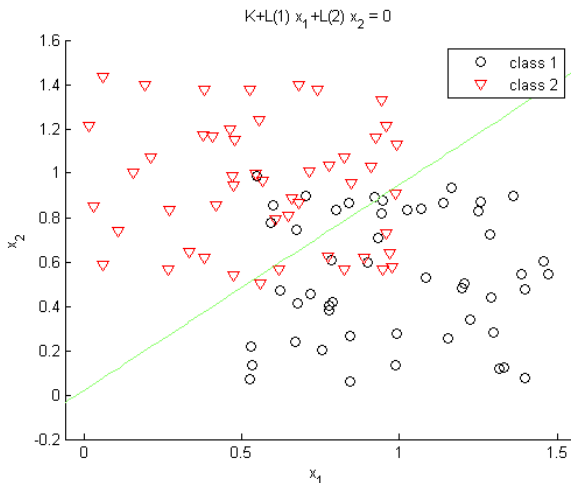
Perceptron



- We can use Perceptron to implement the basic logic gates, i.e. NOT, AND, and OR,
- It is then a well known result from logic that we can construct any logical function from these three operations.
- But, we generally avoid decomposition of complex problems into simple logic gates.
- Networks usually have a much more complex architecture.

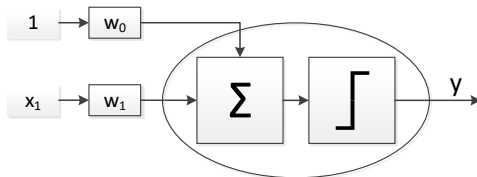
Linear Classification

We can use perceptron as linear classifier.



Classification

Classification: neuron with threshold function, one input signal and one output signal

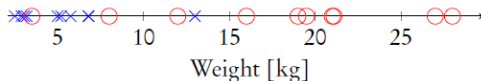


- Input:

$$v = w_0 + w_1 x_1$$

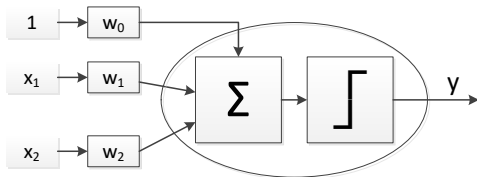
- Classification:

$$y = \begin{cases} 1 & \text{if } x_1 \geq -w_0/w_1 \\ 0 & \text{if } x_1 < -w_0/w_1 \end{cases}$$



Linear Separation

Classification: Perceptron neuron with **two** input signal and output signal



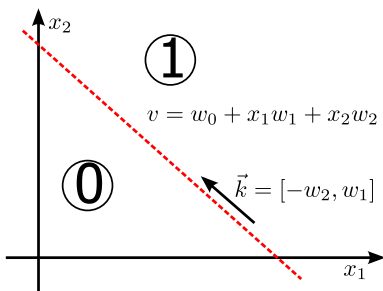
- Classification:
 - Neural potential:
$$v = w_0 + w_1x_1 + w_2x_2$$
- $$y = \begin{cases} 1 & \text{if } x_2 \geq -(w_0 + w_1x_1)/w_2 \\ 0 & \text{if } x_2 < -(w_0 + w_1x_1)/w_2 \end{cases}$$

Linear Separation

$$y = \begin{cases} 1 & \text{if } x_2 \leq \frac{w_1}{w_2}x_1 + \frac{w_0}{w_2} \\ 0 & \text{if } x_2 > \frac{w_1}{w_2}x_1 + \frac{w_0}{w_2} \end{cases}$$

Parallel vector (inclination):

$$k = \begin{bmatrix} -w_2 \\ w_1 \end{bmatrix} \quad k = \begin{bmatrix} w_2 \\ -w_1 \end{bmatrix}$$



Linear Separation

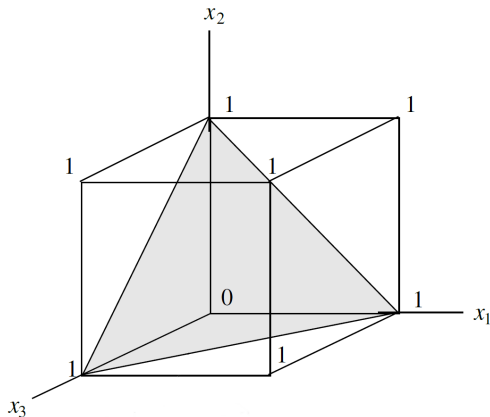
- If we have two inputs, then the weights define a decision boundary that is a straight line in the two dimensional input space of possible input values.
- If we have n inputs, the weights define a decision boundary that is an $n-1$ dimensional hyperplane:

$$w_0 + w_1x_1 + w_2x_2 + \dots + w_nx_n = 0$$

- This hyperplane is clearly still linear (i.e. straight/flat) and can still only divide the space into two regions.

Linear Separation

The hyperplane for 3 inputs



Example

Our task is to determine the weights and thresholds for neurons implementation of logic operations.

Inspection

NOT

x_1	y
0	1
1	0

OR

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	1

AND

x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1

Example

Example of AND logic: we have two weights w_1 and w_2 and the threshold w_0 .

$$y = \begin{cases} 1 & \text{if } w_2x_2 + w_1x_1 + w_0 \geq 0 \\ 0 & \text{if } w_2x_2 + w_1x_1 + w_0 < 0 \end{cases}$$

So the training data lead to:

x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1

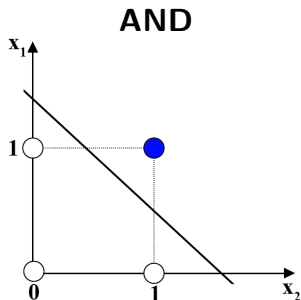
$$\begin{aligned} w_1 0 + w_2 0 + w_0 &< 0 \\ w_1 0 + w_2 1 + w_0 &< 0 \\ w_1 1 + w_2 0 + w_0 &< 0 \\ w_1 1 + w_2 1 + w_0 &\geq 0 \end{aligned}$$

$$\begin{aligned} w_0 &< 0 \\ w_1 &< -w_0 \\ w_2 &< -w_0 \\ w_1 + w_2 &\geq -w_0 \end{aligned}$$

There is an infinite number of solutions.

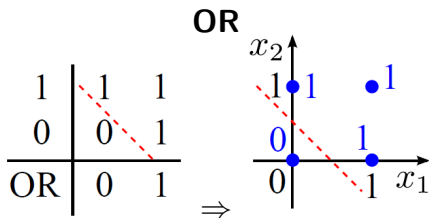
Decision Boundaries

How can we find weights without having to compute and solve large numbers of inequalities? Answer: Decision boundaries.



- Draw a line that satisfies the logic conditions ($w_1x_1 + w_2x_2 + w_0 > 0$),
- Assume weights: $w_1 = 1$ and $w_2 = 1$ ($x_1 = -x_2 - w_0$),
- $-w_0$ is the point of intersection of the line with the axis x_1 ,
- find weight value threshold:
 $w_0 = (-2, -1)$ ($1 = 0 - w_0$)
- Finally, the equation is
 $y = (x_1 + x_2 - 1.5)$,

Decision Boundaries

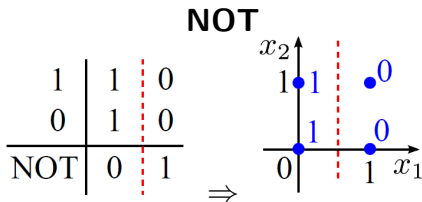


$$w_1 = 1$$

$$w_2 = 1$$

$$w_0 \in (-1, 0)$$

$$y = \text{sgn}(x_1 + x_2 - 0.5)$$



$$w_1 = -1$$

$$w_2 = 0$$

$$w_0 \in (0, 1)$$


$$y = \text{sgn}(-x_1 + 0.5)$$

$$y = \text{sgn}(w_2 x_2 + w_1 x_1 + w_0)$$

Exclusive OR

Is it possible to calculate XOR with use of one boundary

1	1	0
0	0	1
XOR	0	1



Exclusive OR

XOR logic:

$$y = \text{sgn}(w_2x_2 + w_1x_1 + w_0)$$

So the training data lead to:

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

$$w_1 0 + w_2 0 + w_0 < 0$$

$$w_1 0 + w_2 1 + w_0 \geq 0$$

$$w_1 1 + w_2 0 + w_0 \geq 0$$

$$w_1 1 + w_2 1 + w_0 < 0$$

$$w_0 < 0$$

$$w_1 \leq w_0$$

$$w_2 \leq w_0$$

$$w_1 + w_2 > w_0$$

- Inequalities are incompatible with each other - no solution.
- We need two straight lines to separate the different outputs/decisions.
- We need more complex network.

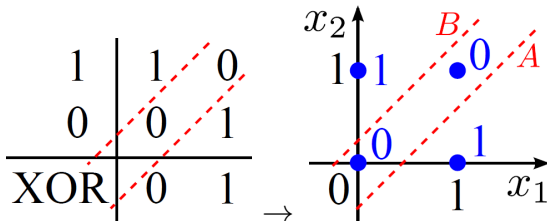
Exclusive OR

1	1	0
0	0	1
XOR	0	1

Questions:

- How to design perceptron that can implement XOR?
- How many layers and neurons should consist of such a network?

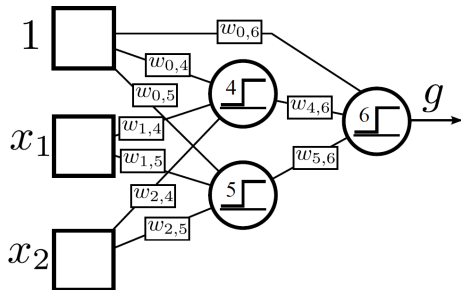
Exclusive OR



Architecture:

- Two inputs (plus threshold) and single output.
- Two neurons in hidden layer.
- And one neuron in output layer.

Exclusive OR



4 – separation along A

1	1	0
0	0	1
XOR	0	1

5 – separation along B

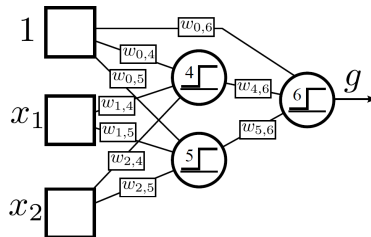
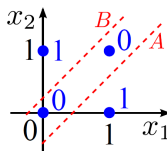
1	1	0
0	0	1
XOR	0	1

6 – OR or AND

1	1	0
0	0	1
XOR	0	1

Exclusive OR

1	1	0
0	0	1
XOR	0	1

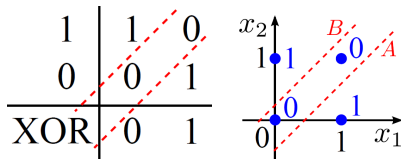


Equations of 3 neurons:

$$\begin{cases} y_4 = f(v_4) \\ y_5 = f(v_5) \\ g = y_6 = f(v_6) \end{cases}$$

$$\begin{cases} v_4 = w_{0,4} + w_{1,4}x_1 + w_{2,4}x_2 \\ v_5 = w_{0,5} + w_{1,5}x_1 + w_{2,5}x_2 \\ v_6 = w_{0,6} + w_{4,6}y_4 + w_{5,6}y_5 \end{cases}$$

Exclusive OR



$$\begin{cases} y_4 = f(w_{0,4} + w_{1,4}x_1 + w_{2,4}x_2) \\ y_5 = f(w_{0,5} + w_{1,5}x_1 + w_{2,5}x_2) \\ y_6 = f(w_{0,6} + w_{4,6}y_4 + w_{5,6}y_5) \end{cases}$$

Line B (value below the line):

$$w_{0,5} = -0.5, w_{1,5} = -1, w_{2,5} = 1$$

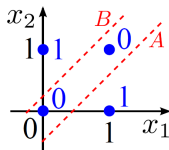
$$y_4 = f(-x_1 + x_2 - 0.5)$$

Line A (value above the line - change for the same class):

$$w_{0,4} = -0.5, w_{1,4} = 1, w_{2,4} = -1$$

$$y_4 = f(x_1 - x_2 - 0.5)$$

Exclusive OR



Line B with all steps:

$$w_1x_1 + w_2x_2 + w_0 \geq 0$$

$$w_2x_2 = -w_1x_1 - w_0$$

$$\text{assume : } w_2 = 1, w_1 = -1$$

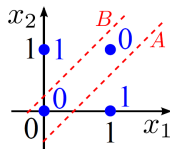
$$x_2 = x_1 - w_0$$

$$\text{check : } \quad x_2 = 0, x_1 = 0 \quad x_2 = 1, x_1 = 0$$

$$w_0 \in (-1, 0) \Rightarrow \quad w_0 = -0.5$$

$$-x_1 + x_2 - 0.5 \geq 0$$

Exclusive OR



Line A with all steps, we want to be in same class as line B (\leq):

$$w_1x_1 + w_2x_2 + w_0 \leq 0$$

$$w_2x_2 = -w_1x_1 - w_0$$

$$\text{assume : } w_2 = 1, w_1 = -1$$

$$x_2 = x_1 - w_0$$

$$\text{check : } \quad x_2 = 0, x_1 = 0 \quad x_2 = 0, x_1 = 1$$

$$w_0 \in (0, 1) \Rightarrow \quad w_0 = 0.5$$

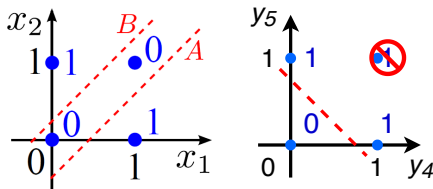
$$x_1 - x_2 - 0.5 \geq 0$$

Exclusive OR

$$\begin{cases} y_4 = f(w_{0,4} + w_{1,4}x_1 + w_{2,4}x_2) \\ y_5 = f(w_{0,5} + w_{1,5}x_1 + w_{2,5}x_2) \\ y_6 = f(w_{0,6} + w_{4,6}y_4 + w_{5,6}y_5) \end{cases}$$

Input		Layer 1				Output
		Potential		Output		
x_1	x_2	v_4	v_5	y_4	y_5	y_6
0	0	-0,5	-0,5	0	0	0
0	1	-1,5	0,5	0	1	1
1	0	0,5	1,5	1	0	1
1	1	-0,5	-0,5	0	0	0

Exclusive OR



$$\begin{cases} y_4 = f(w_{0,4} + w_{1,4}x_1 + w_{2,4}x_2) \\ y_5 = f(w_{0,5} + w_{1,5}x_1 + w_{2,5}x_2) \\ y_6 = f(w_{0,6} + w_{4,6}y_4 + w_{5,6}y_5) \end{cases}$$

Line C:

$$w_{4,6} = 1, w_{5,6} = 1, w_{0,6} = -0.5$$

$$g = g(y_4 + y_5 - 0.5)$$

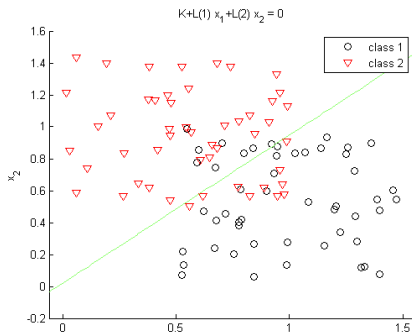
Exclusive OR

$$\begin{cases} y_4 = f(w_{0,4} + w_{1,4}x_1 + w_{2,4}x_2) \\ y_5 = f(w_{0,5} + w_{1,5}x_1 + w_{2,5}x_2) \\ y_6 = f(w_{0,6} + w_{4,6}y_4 + w_{5,6}y_5) \end{cases}$$

Inputs		Layer 1				Output layer	
		Potentials		Outputs		Potential	Output
x_1	x_2	v_4	v_5	y_4	y_5	v_6	y_6
0	0	-0,5	-0,5	0	0	-0,5	0
0	1	-1,5	0,5	0	1	0,5	1
1	0	0,5	1,5	1	0	0,5	1
1	1	-0,5	-0,5	0	0	-0,5	0

Decision Boundaries

- Linear models with fixed basis functions have limited applicability
- Generally, we will work with input data that are not binary, and expect our neural networks to find more complex boundaries.



Neural Networks

Neural Networks

Neural Networks

A neural network is a directed graph consisting of neurons with interconnecting synaptic and activation links and is characterized by four properties:

- Each neuron is represented by a set of linear synaptic links, an externally applied bias, and a possibly nonlinear activation link. The bias is represented by a synaptic link connected to an input fixed at 1.

Simon Haykin, Neural Networks and Learning Machines, Third Edition, Pearson Education, 2009

Neural Networks

- The synaptic links of a neuron weight their respective input signals.
- The weighted sum of the input signals defines the induced local field of the neuron in question
- The activation function squashes the induced local field of the neuron to produce an output.

Simon Haykin, Neural Networks and Learning Machines, Third Edition, Pearson Education, 2009

Network Architectures

Neural Networks are a group of at least two connected neurons.

The manner in which the neurons of a neural network are structured is intimately linked with the learning algorithm used to train the network. Two main categories of neural networks:

- Feedforward (FFNN),
- Feedback (FBNN): recurrent and recursive neural networks.

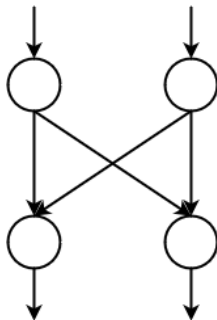
In FBNN modeling, it is very important to understand FFNN.

Feedforward Neural Networks

Characteristics of FFNN:

- Information flows in one direction,
- There is no feedback loops,
- Typical topology is multi-layered network,
- Network can be sparse with not all connections being present,
- FFNN are static,
- Function of network is a composition of many different functions, e.g.

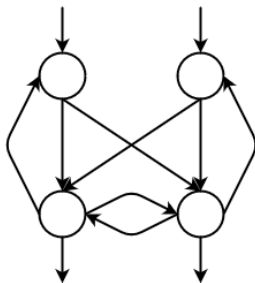
$$f(x) = f^{(3)}(f^{(2)}(f^{(1)}(x)))$$



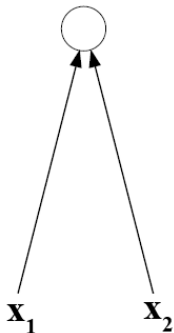
Feedback Neural Networks

Characteristics of FBNN:

- Information flows in both directions by introducing loops in the network,
- **NN must have feedback loops,**
- Typical topology is complex multi-layered network.
- FFBN are dynamic,
- Computation is not uniquely defined by the interconnection pattern and the temporal dimension must be considered, e.g.
$$f(x) = f(x_t, f(x_{t-1}, f(x_{t-2})))$$



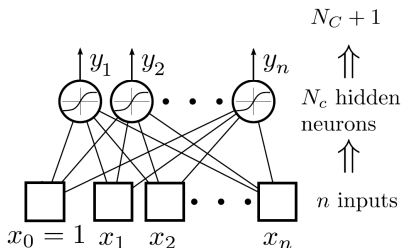
Network Architectures



In general, we may identify three common FFNN architectures:

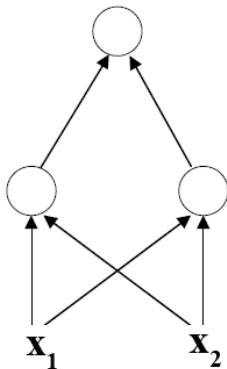
- **Single-Layer Feed-forward** - One input layer and one output layer of artificial neurons without feed-back connections.
- Multi-Layer Feed-forward.

Single-Layer FFNN



- an input layer of source nodes that projects directly onto an output layer of neurons,
- single-layer network referring to the output layer,
- we do not count the input layer of source nodes because no computation is performed there.

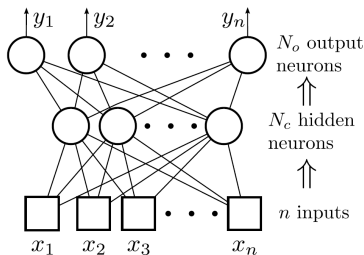
Network Architectures



In general, we may identify three common FFNN architectures:

- Single-Layer Feed-forward.
- **Multi-Layer Feed-forward** - One input layer, one output layer, and one or more hidden layers of artificial neurons without feed-back connections. The hidden layers sit in between the input and output layers, and are thus hidden from the outside world.

Multilayer FFNN

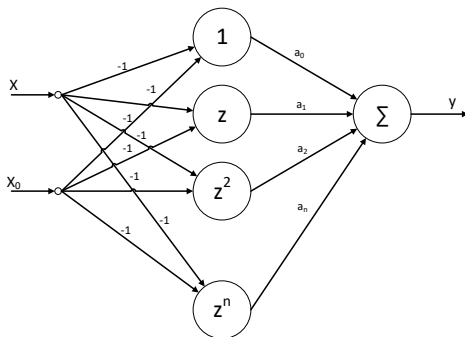


- The number of neurons depends on the task.
- The number of in/out easy to determine.
- The number of hidden layers depends on task.
- input to output data processing.
- Inputs are sometimes called layer.

Multilayer FFNN

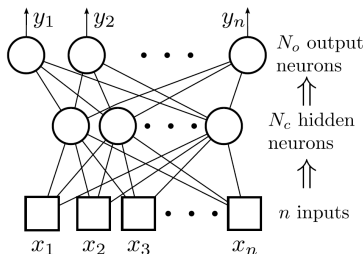
Example: Taylor network as Single Output and Single Input Layer NN

$$y = a_0 + a_1(x - x_0) + a_2(x - x_0)^2 + \dots + a_n(x - x_0)^n$$



Multi-layer Perceptron

Multi-layer perceptron (MLP)

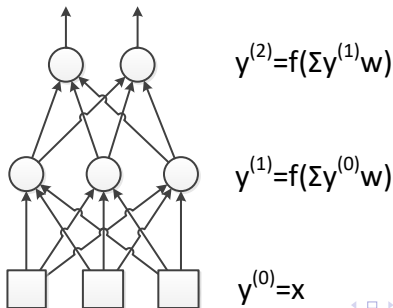


- Multiple fully connected layers,
- Non-linear activation functions,
- Given layer neurons have the same function,
- Typical topology.

Number of layers

Number of layers of neural network:

- Conventionally, the input layer is layer 0, and when we talk of an N layer network we mean there are N layers of weights and N non-input layers of processing units.
- Thus a two layer Perceptrons ($N = 2$):



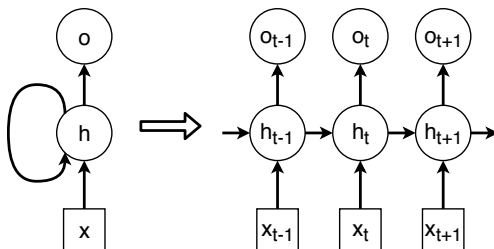
Recurrent neural network

A Recurrent Neural Network (RNN) is a class of artificial neural network that has memory or feedback loops that allow it to better recognize patterns in data.

- recurrent networks have connections feeding the hidden layers of the neural network back into themselves - these are called recurrent connections (hidden state),
- recurrent networks use their understanding of past events to process the input vector,
- network particularly useful when a sequence of data is being processed, but it can also be used on non-sequential data.

Recurrent neural network

Recurrent networks can be visualized as multiple copies of a neural network, with the output of one serving as an input to the next network.



Questions

