University of Guilan

Computational Intelligence

Subject4: ANNs in Practice

Instructor: Ali Tourani

A.Tourani1991@gmail.Com

Agenda

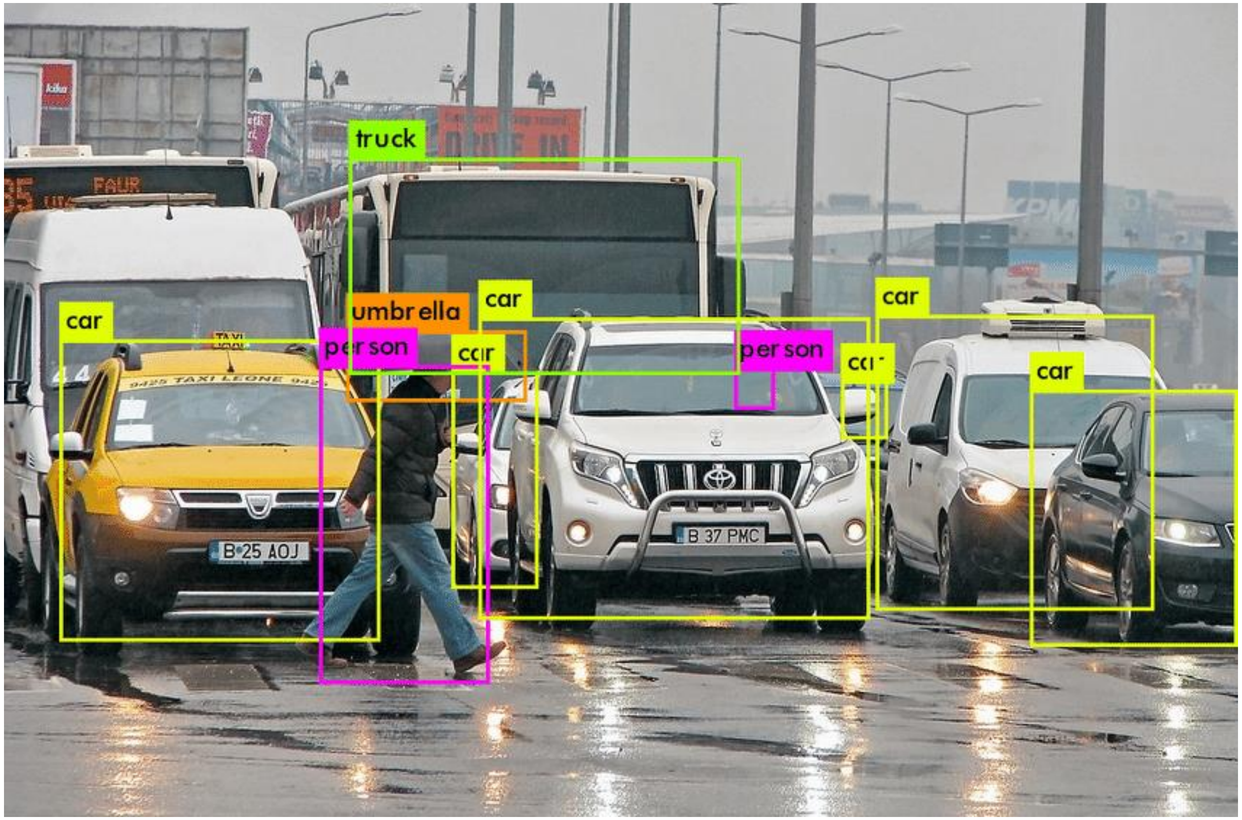
- How do ANNs learn?
- McCulloch-Pitts Neuron
- Hebb Neural Network
- Perceptron

How do ANNs learn?

- Choosing a set of weight is easy to solve linear problems
- What about non-linear issue?
- How can we adjust weights to recognize human face in an image?

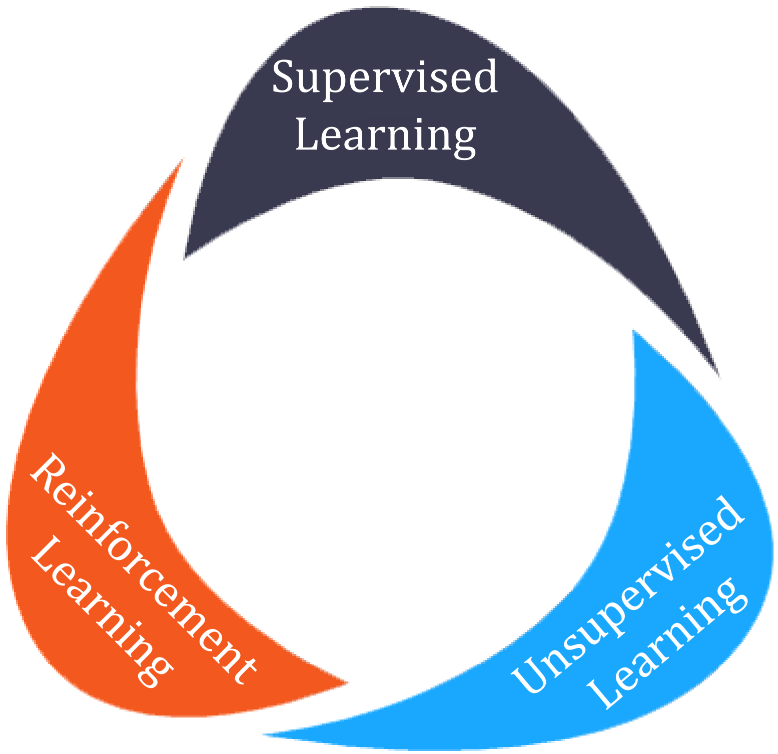
- ***Solution:*** providing feedback to the ANN on how to update weights to learn from its experience
  - Learning process: altering the network's weights using learning algorithms
  - Finding a set of weight matrices for mapping any input to a correct output

How do ANNs learn?



How do ANNs learn?

Learning Types

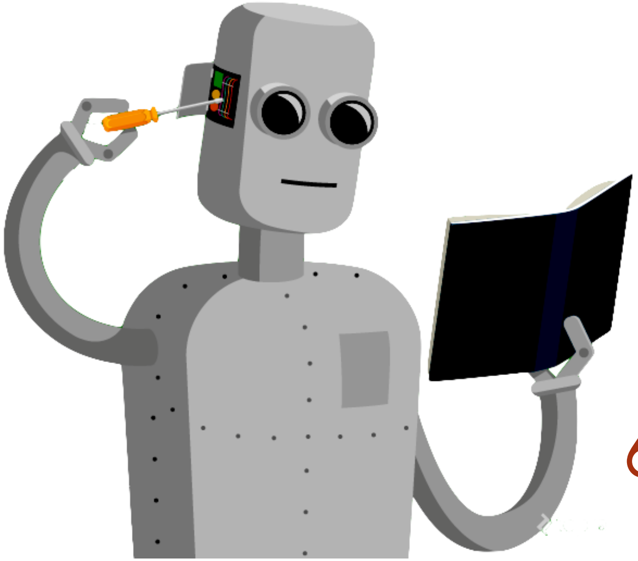


How do ANNs learn?

University of Guilan

Learning Types - Supervised Learning

- While training, the desired output is also provided with the input
- Possible to calculate an error based on its real and calculated outputs
  - Why? to make corrections to the network by updating its weights
  - Basically, feedforward networks
- There is a target class for any inputs
- Related to: Classification problems



How do ANNs learn?

Learning Types - Supervised Learning

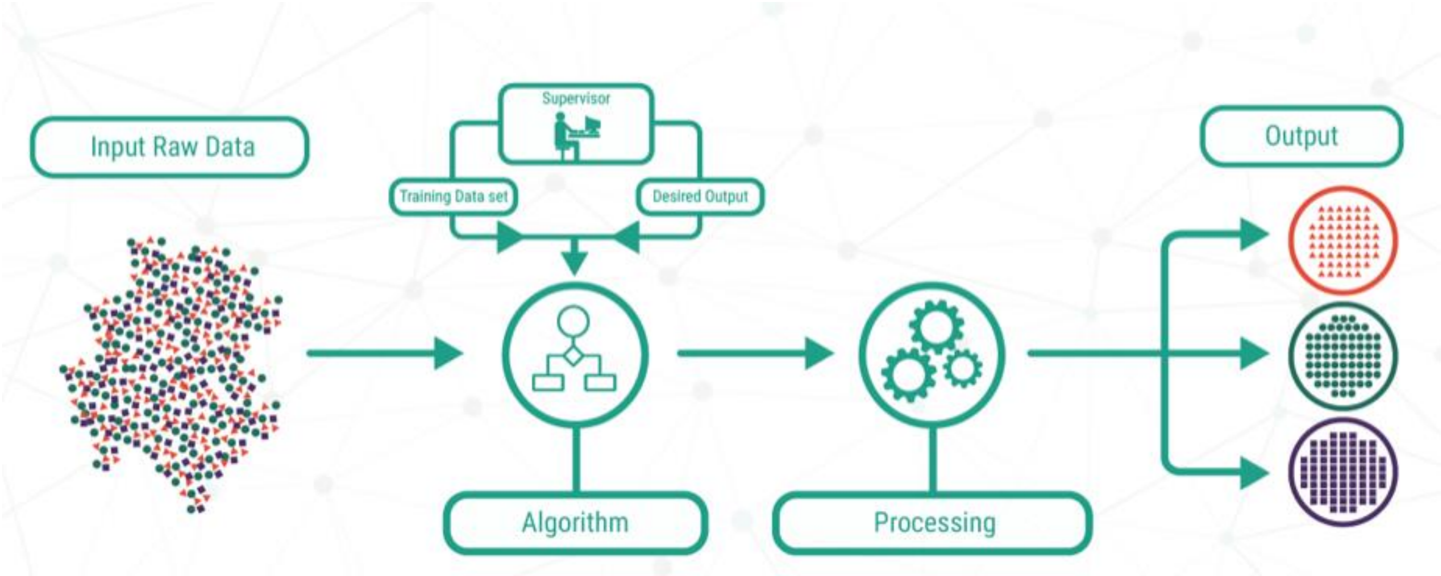
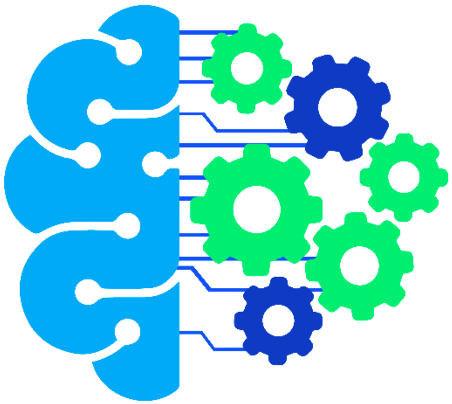


Image from https://www.technative.io/

How do ANNs learn?

Learning Types - Unsupervised Learning

- The ANN should find a pattern within the provided inputs
  - No external aid!
  - The patterns are recognized based on similarities
  - For instance, predicting a user's preferences based on the preferences of other similar users
- Related to: Clustering (data mining)



How do ANNs learn?

Learning Types - Unsupervised Learning

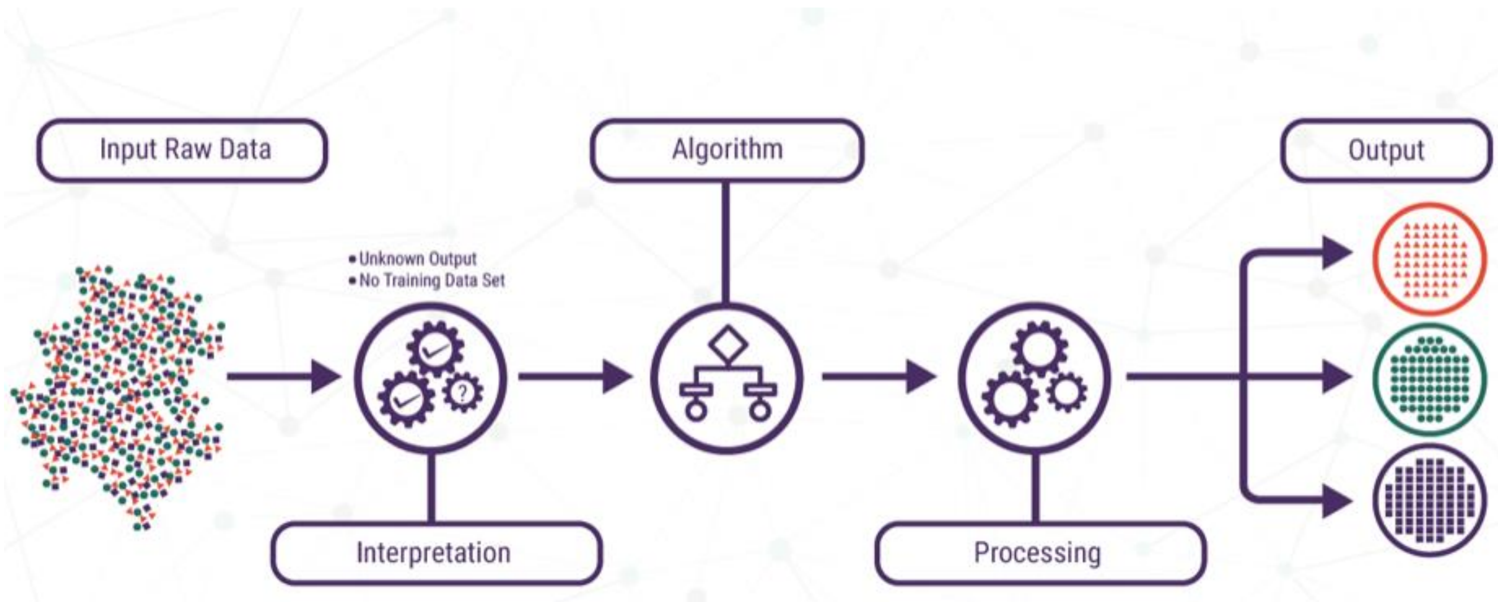


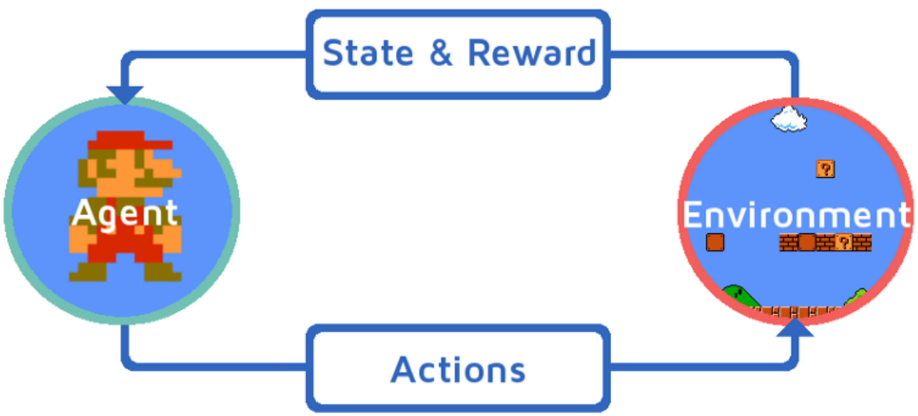
Image from https://www.technative.io/

How do ANNs learn?

University of Guilan

Learning Types - Reinforcement Learning

- Similar to Supervised Learning, a feedback is provided
- But we just inform the ANN about how good it was predicted
- Goal: maximizing the reward through trial-and-error

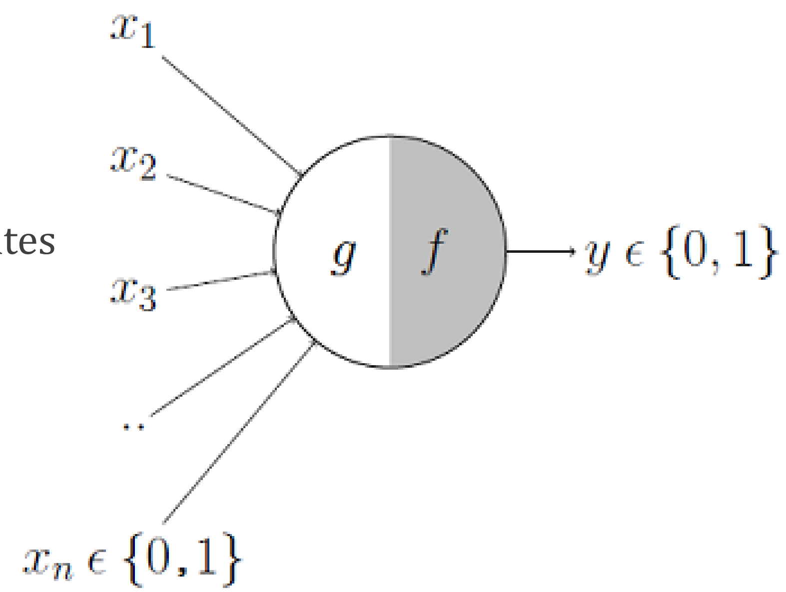


How do ANNs learn?

Learning Types - Reinforcement Learning

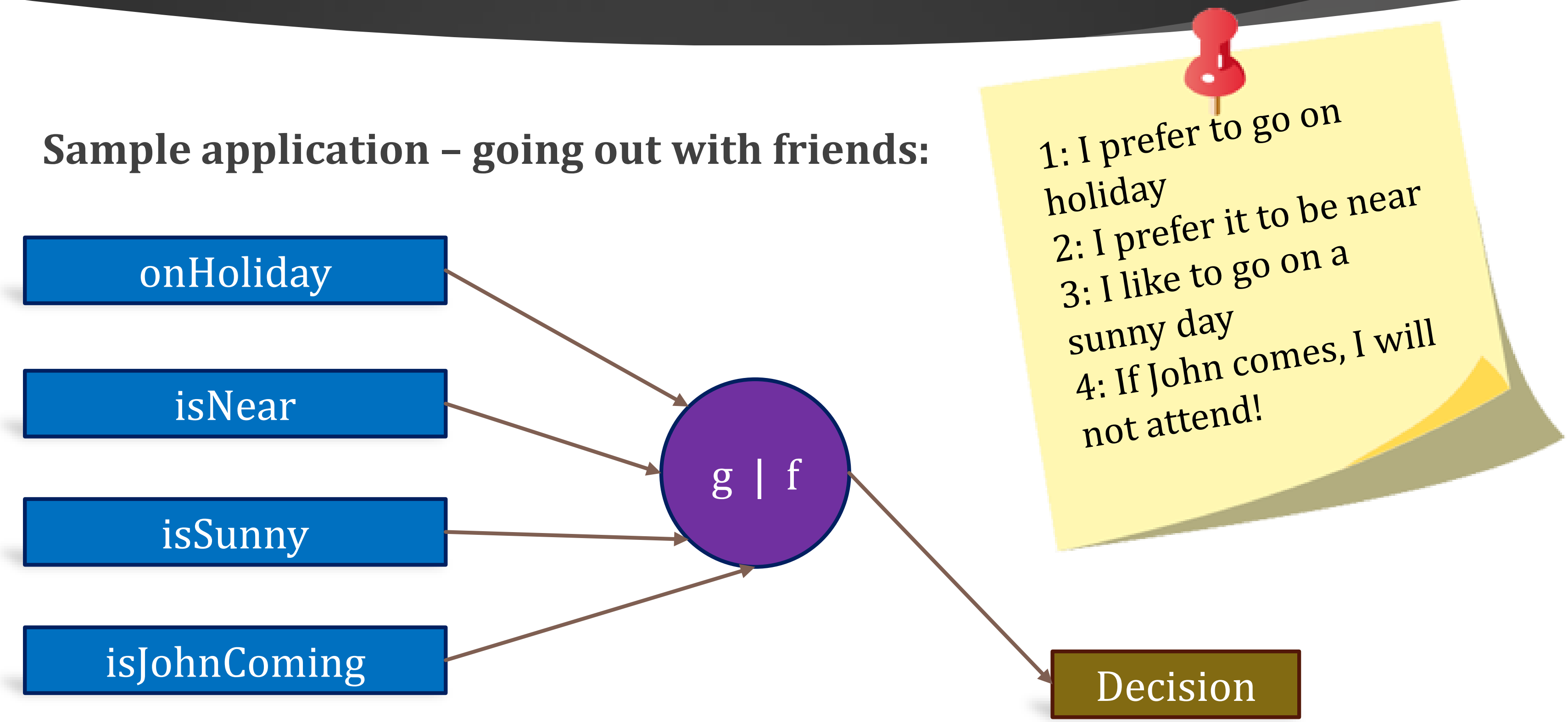
McCulloch-Pitts Neuron

- Introduced in 1943 by McCulloch (neuroscientist) and Pitts (logician)
- Properties:
  - The inputs and outputs are binary
  - The AF is binary
  - The first part of the network ***(g)*** aggregates the input data
  - The second part ***(f)*** makes a decision



McCulloch-Pitts Neuron

Sample application – going out with friends:



McCulloch-Pitts Neuron

Sample application – going out with friends:

- Two types of inputs:
  - **Inhibitory**: inputs with maximum effects on the decision making
    - Note: irrespective of others
    - Sample: if x4 (isJohnComing) is 1, my output will always be 0
  - **Excitatory**: will make the neuron fire when combined to others

- Thresholding Parameter (θ):
    - Sample: I attend the journey when the sum turns out to be more than 2 or more
    - How to calculate?!

McCulloch-Pitts Neuron

1: I prefer to go on holiday
2: I prefer it to be near
3: I like to go on a sunny day
4: If John comes, I will not attend!

Sample application – going out with friends:

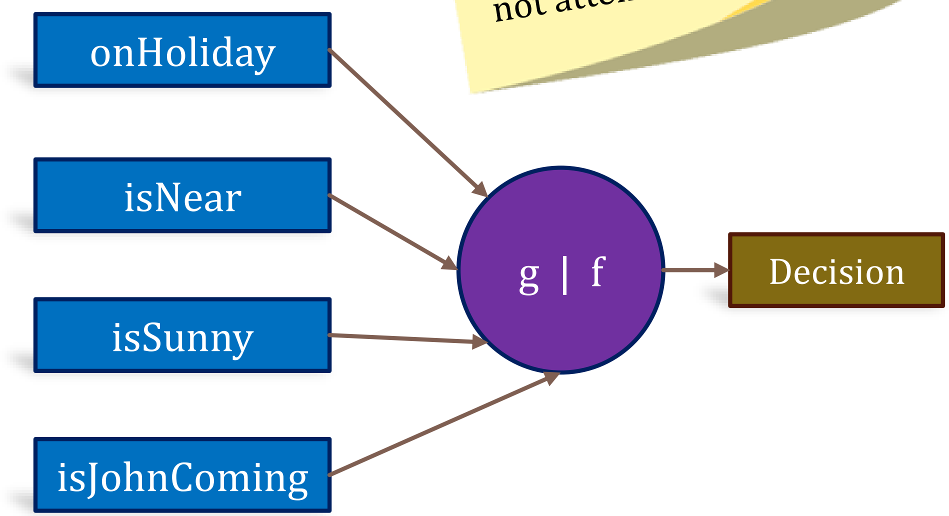
- Inhibitory nodes: *isJohnComing*
- Threshold: 3 (at least two conditions)

- So, if John comes ➔ 0 (not attend)
- If it is sunny, and holiday and

John doesn’t come ➔ 1 (attend)

- If John doesn’t come and it is sunny

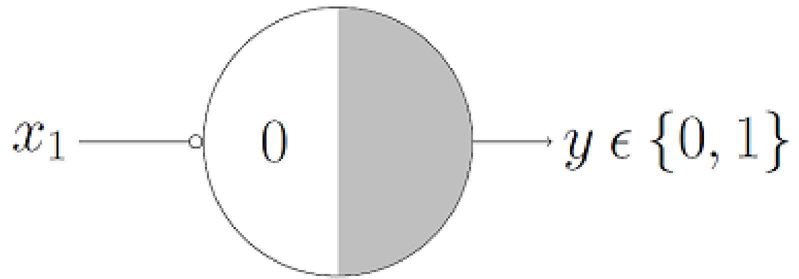
➔ 0 (not attend, lower than threshold)



McCulloch-Pitts Neuron

Boolean Functions using M-P Neuron

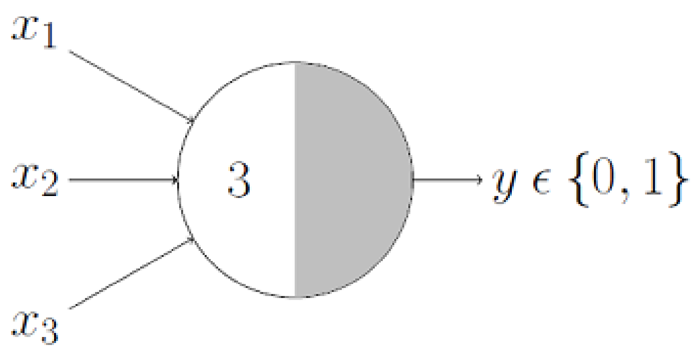
- NOT function
  - Fires with no conditions (takes the input as an inhibitory input)
  - Here, if $g(x) \geq 0$



McCulloch-Pitts Neuron

Boolean Functions using M-P Neuron

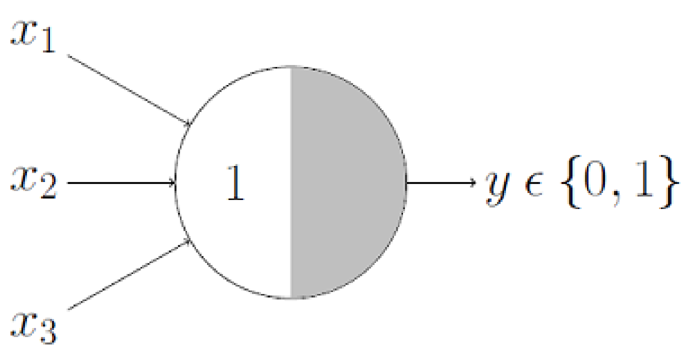
- AND function
  - Fires when ALL inputs are ON (or 1)
  - Here, if $g(x) \geq 3$



McCulloch-Pitts Neuron

Boolean Functions using M-P Neuron

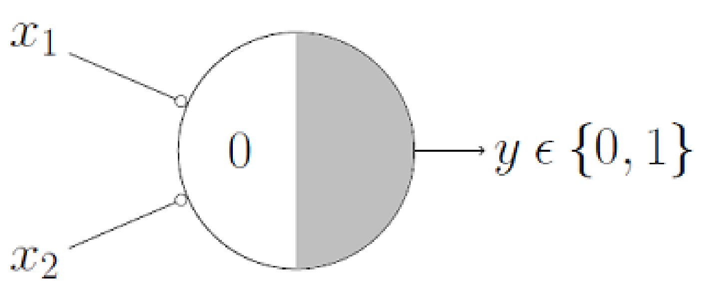
- OR function
  - Fires when AT LEAST ONE OF the inputs are ON (or 1)
  - Here, if $g(x) \geq 1$



McCulloch-Pitts Neuron

Boolean Functions using M-P Neuron

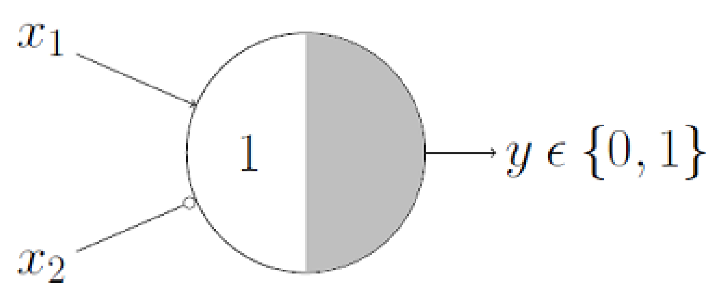
- NOR function
  - Fires when ALL of the inputs are ZERO
  - Here, if $g(x) \geq 0$



McCulloch-Pitts Neuron

Boolean Functions using M-P Neuron

- A more complex function
  - Here, x2 is inhibitory
    - So if x2=1 ➔ y=0
  - It triggers when:
    - x1 is 1 and x2 is 0



$$x_1 \ AND \ !x_2{}^*$$

McCulloch-Pitts Neuron

University of Guilan

Limitations of M-P Neuron

- Inability to have non-Boolean inputs
- Sometimes, we do not know about the threshold
- There is no priorities in choosing nodes (no weights)
- Inability to solve non-linear problems (e.g. XOR)

McCulloch-Pitts Neuron

University of Guilan

Extension – Weighted M-P Neuron

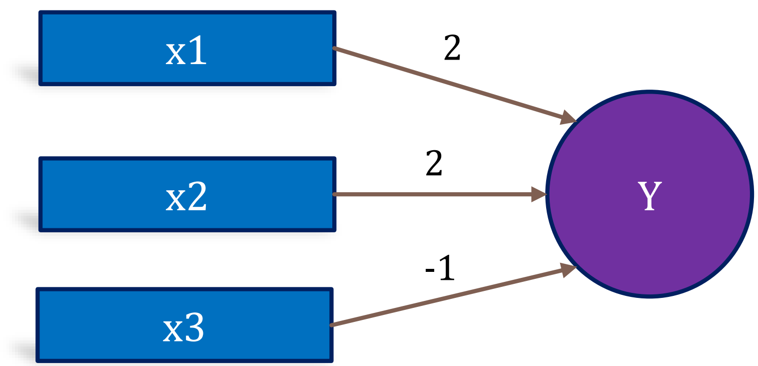
- All input signals arrive at the same time
- Each input signal can have a weight w_i to show its strength
  - The value of positive weights are equal
- Each neuron has a thresholding parameter θ
  - Neuron is activated if $net \geq \theta$
- To calculate the threshold, if n is the number of signals with positive weights, w is the value of the positive weights, and p is the number of signals with negative weights :

$$\theta > nw - p$$

McCulloch-Pitts Neuron

Extension – Weighted M-P Neuron

- Sample:



$$\theta > nw - p \quad \rightarrow \quad \theta > (2 * 2) - 1$$

Hebb Neural Network

- The major drawback of M-P: the capability of learning
  - They cannot develop capabilities for classification or recognition
- What is Hebbian Learning?
  - A mechanism to update weights between neurons
  - The process of repeatedly activating weakly connected neurons
    - Result: **providing stronger connections**

Hebb Neural Network

- The simplest Learning Rule for ANNs, AKA **Hebbian Learning Rule**
- Learning process is equal to changing the weights of the network
  - Initialization of weights

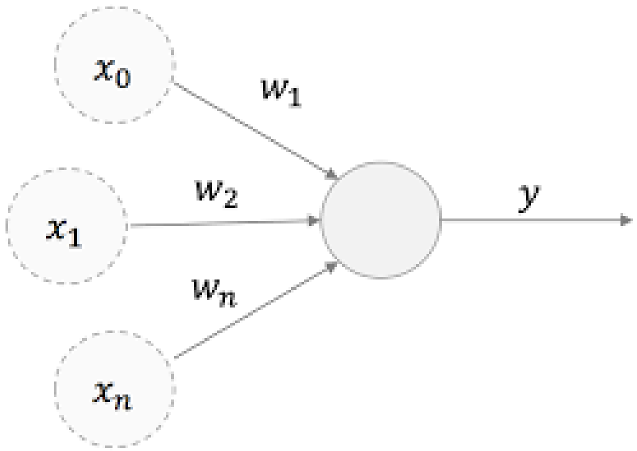
$$w_i = 0 \quad (i = 1 \; \dots n)$$

  - Calculation of the *net*
  - Calculation of AF's output

$$y = t \qquad (AKA\ target)$$

  - Updating weights

$$w_i\,(new) = w_i\,(old) + x_i y$$
$$b(new) = b(old) + y$$



Hebb Neural Network

Sample: AND function with different inputs

- We know that:

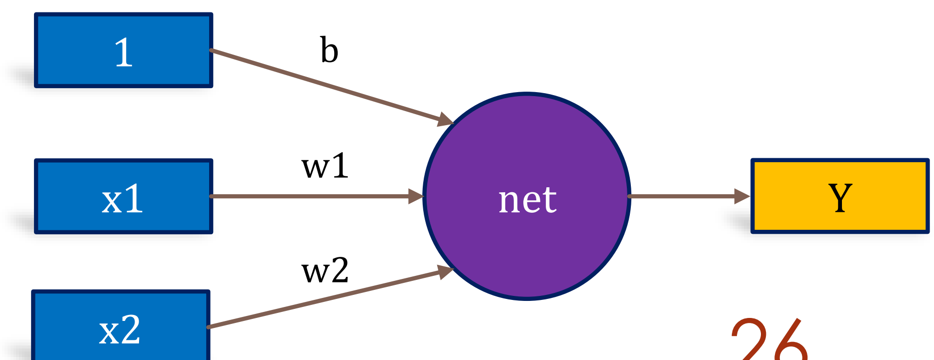
$$y_{in} = 1.b + w_1 x_1 + w_2 x_2$$

Input			Target
X1	**X2**	**bias**	**t**
1	1	1	1
1	0	1	0
0	1	1	0
0	0	1	0

- Now, lets define Δw and Δb:

$$w(new) = w(old) + \Delta w$$

$$\Delta w_i = x_i t \quad and \quad \Delta b = t$$



Hebb Neural Network

Sample: AND function with **binary** inputs

$Initiallize\ weights$: (0,0,0)

$\Delta w_i = x_i t$ $\Delta b = t$ $w(new) = w(old) + \Delta w$

Input			Target	Weight Changes			Weights		
X1	X2	1	t	Δw1	Δw2	Δb	W1	W2	b
1	1	1	1						
1	0	1	0						
0	1	1	0						
0	0	1	0						

We know the values of t

Hebb Neural Network

University of Guilan

Sample: AND function with binary inputs

$Initiallize\ weights$: (0,0,0)

$\Delta w_i = x_i t$ $\Delta b = t$ $w(new) = w(old) + \Delta w$

Input			Target	Weight Changes			Weights		
X1	X2	1	t	Δw1	Δw2	Δb	W1	W2	b
1	1	1	1	1	1	1			
1	0	1	0						
0	1	1	0						
0	0	1	0						

We know the values of t

Hebb Neural Network

University of Guilan

Sample: AND function with **binary** inputs

$Initiallize\ weights$: (0,0,0)

$\Delta w_i = x_i t$ $\Delta b = t$ $w(new) = w(old) + \Delta w$

Input			Target	Weight Changes			Weights		
X1	X2	1	t	Δw1	Δw2	Δb	W1	W2	b
1	1	1	1	1	1	1	1	1	1
1	0	1	0						
0	1	1	0						
0	0	1	0						

We know the values of t

Hebb Neural Network

Sample: AND function with binary inputs

$Initiallize\ weights$: (0,0,0)

$\Delta w_i = x_i t$ $\Delta b = t$ $w(new) = w(old) + \Delta w$

Input			Target	Weight Changes			Weights		
X1	X2	1	t	Δw1	Δw2	Δb	W1	W2	b
1	1	1	1	1	1	1	1	1	1
1	0	1	0	0	0	0			
0	1	1	0						
0	0	1	0						

We know the values of t

Hebb Neural Network

University of Guilan

Sample: AND function with binary inputs

$Initiallize\ weights$: (0,0,0)

$\Delta w_i = x_i t$ $\Delta b = t$ $w(new) = w(old) + \Delta w$

Input			Target	Weight Changes			Weights		
X1	X2	1	t	Δw1	Δw2	Δb	W1	W2	b
1	1	1	1	1	1	1	1	1	1
1	0	1	0	0	0	0	1	1	1
0	1	1	0						
0	0	1	0						

We know the values of t

Hebb Neural Network

Sample: AND function with **binary** inputs

$Initiallize\ weights$: (0,0,0)

$\Delta w_i = x_i t$ $\Delta b = t$ $w(new) = w(old) + \Delta w$

Input			Target	Weight Changes			Weights		
X1	X2	1	t	Δw1	Δw2	Δb	W1	W2	b
1	1	1	1	1	1	1	1	1	1
1	0	1	0	0	0	0	1	1	1
0	1	1	0	0	0	0	1	1	1
0	0	1	0	0	0	0	1	1	1

We know the values of t

Hebb Neural Network

Sample: AND function with binary inputs

$Initiallize\ weights$: (0,0,0)

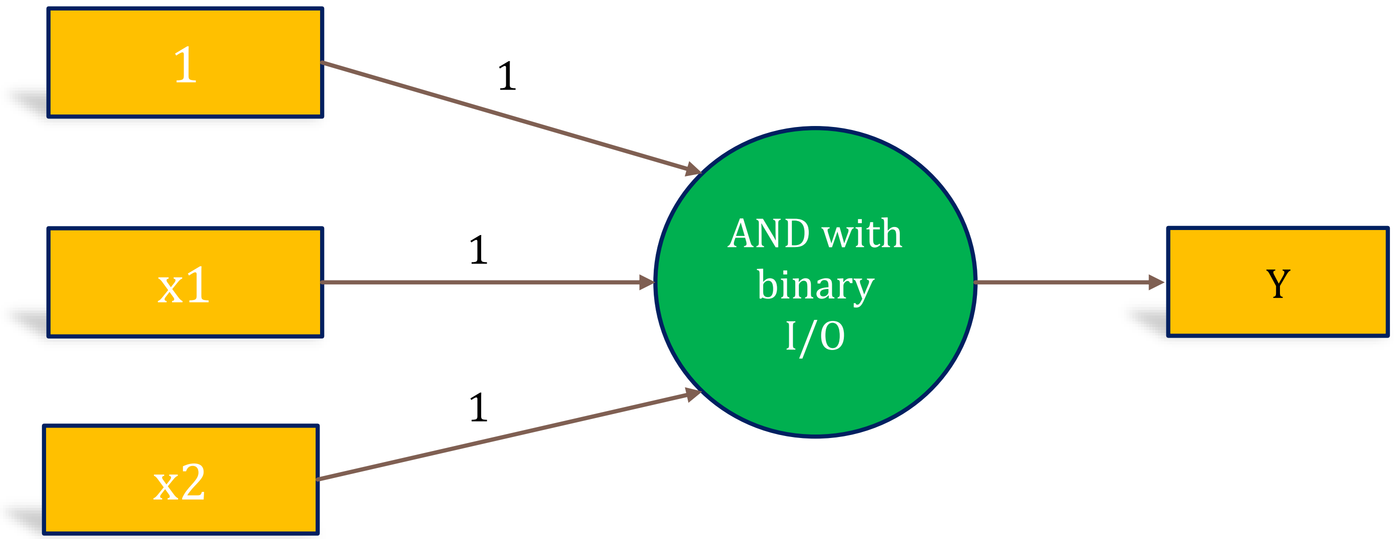
$\Delta w_i = x_i t$ $\Delta b = t$ $w(new) = w(old) + \Delta w$

Input			Target	Weight Changes			Weights		
X1	X2	1	t	Δw1	Δw2	Δb	W1	W2	b
1	1	1	1	1	1	1	1	1	1
1	0	1	0	0	0	0	1	1	1
0	1	1	0	0	0	0	1	1	1
0	0	1	0	0	0	0	1	1	1

The output weights do not change! So, we cannot solve binary AND with Hebb

Hebb Neural Network

Sample: AND function with binary inputs



Hebb Neural Network

Sample: AND function with binary inputs and bipolar outputs

$Initiallize\ weights: (0,0,0)$

$\Delta w_i = x_i t$ $\Delta b = t$ $w(new) = w(old) + \Delta w$

Input			Target	Weight Changes			Weights		
X1	X2	1	t	Δw1	Δw2	Δb	W1	W2	b
1	1	1	1	1	1	1	1	1	1
1	0	1	-1	-1	0	-1	0	1	0
0	1	1	-1	0	-1	-1	0	0	-1
0	0	1	-1	0	0	-1	0	0	-2

Hebb Neural Network

Sample: AND function with binary inputs and bipolar outputs

$Initiallize\ weights$: (0,0,0)

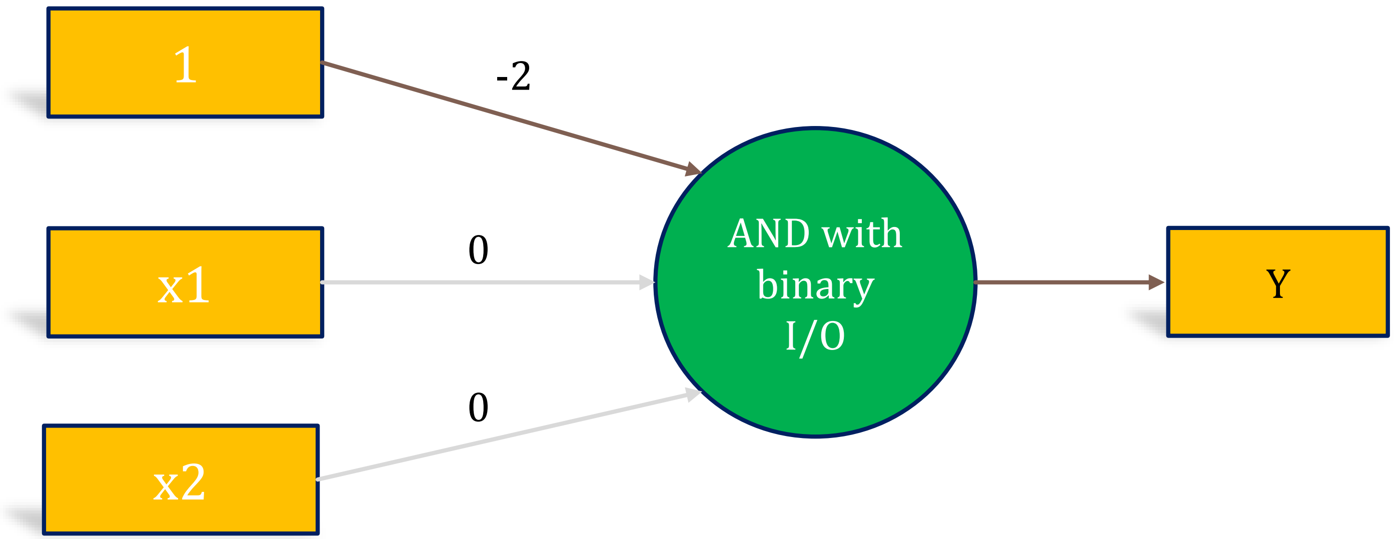
$\Delta w_i = x_i t$ $\Delta b = t$ $w(new) = w(old) + \Delta w$

Input			Target	Weight Changes			Weights		
X1	X2	1	t	Δw1	Δw2	Δb	W1	W2	b
1	1	1	1	1	1	1	1	1	1
1	0	1	-1	-1	0	-1	0	1	0
0	1	1	-1	0	-1	-1	0	0	-1
0	0	1	-1	0	0	-1	0	0	-2

The two signal nodes become inactive ...

Hebb Neural Network

Sample: AND function with binary inputs and bipolar outputs



Hebb Neural Network

Sample: AND function with bipolar inputs and outputs

$Initiallize\ weights$: (0,0,0)

$\Delta w_i = x_i t$ $\Delta b = t$ $w(new) = w(old) + \Delta w$

Input			Target	Weight Changes			Weights		
X1	X2	1	t	Δw1	Δw2	Δb	W1	W2	b
1	1	1	1	1	1	1	1	1	1
1	-1	1	-1	-1	1	-1	0	2	0
-1	1	1	-1	1	-1	-1	1	1	-1
-1	-1	1	-1	1	1	-1	2	2	-2

Hebb Neural Network

Sample: AND function with bipolar inputs and outputs

$Initiallize\ weights$: (0,0,0)

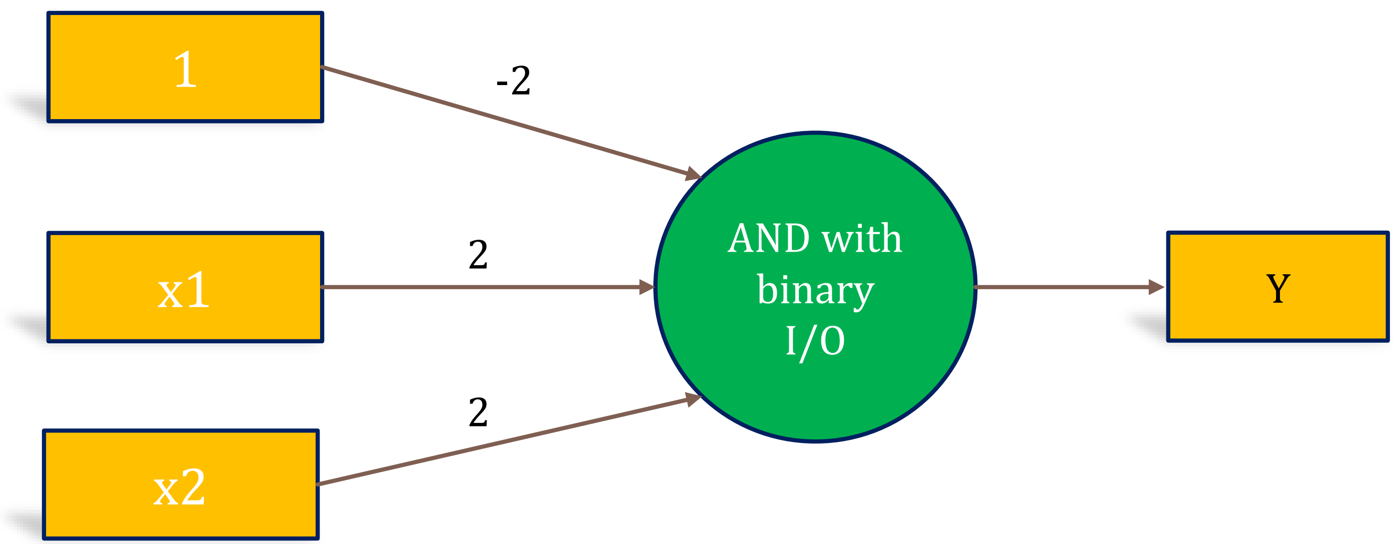
$\Delta w_i = x_i t$ $\Delta b = t$ $w(new) = w(old) + \Delta w$

Input			Target	Weight Changes			Weights		
X1	X2	1	t	Δw1	Δw2	Δb	W1	W2	b
1	1	1	1	1	1	1	1	1	1
1	-1	1	-1	-1	1	-1	0	2	0
-1	1	1	-1	1	-1	-1	1	1	-1
-1	-1	1	-1	1	1	-1	2	2	-2

Now it learns!

Hebb Neural Network

Sample: AND function with bipolar inputs and outputs



Hebb Neural Network

Important conclusion

- The type of the inputs and outputs can make a problem solvable or not
- A very important hint in Hebbian Learning Rule:
  - This network is suitable for **bipolar** data
  - It can help us to recognize missing from wrong data

Perceptron

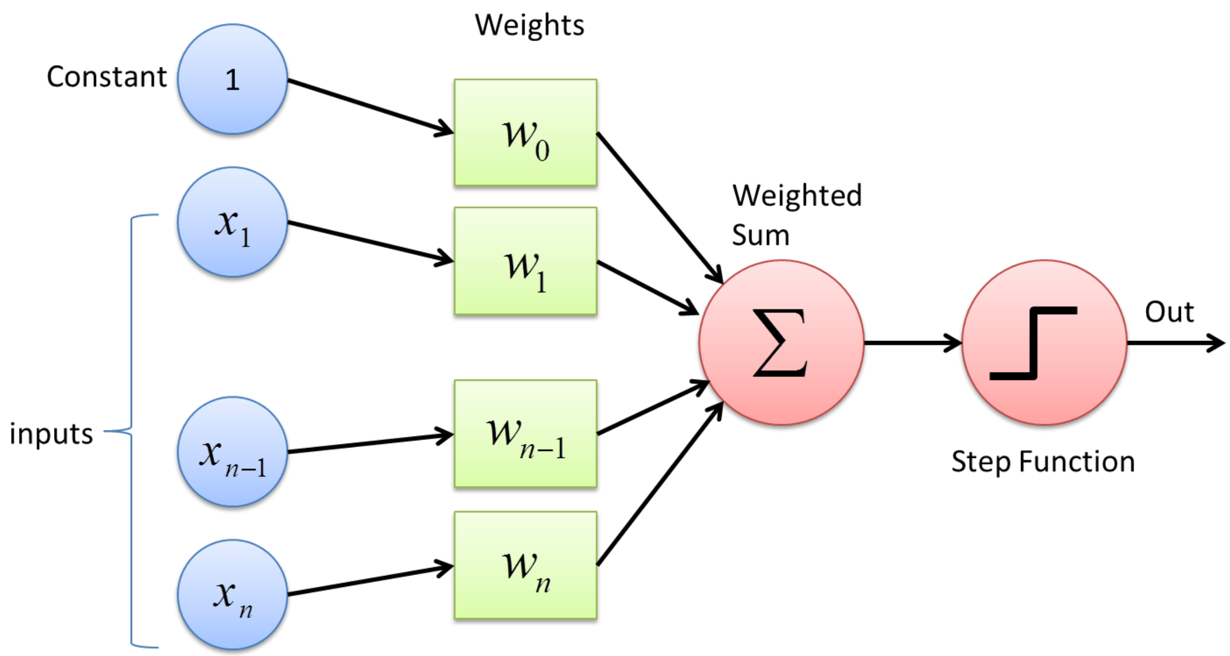
University of Guilan

Single Perceptron Networks

- In their basic type, single-layer feed-forward networks
- Here, weights can be **inhibitory**, **excitatory** or **zero** (-1, +1 or 0)
- The activation function is a binary step function
- A very common type of ANNs for Supervised Learning
- Better learning process comparing to Hebb NN
- We may need several **epochs** to finish the learning process

Perceptron

Single Perceptron Networks



Perceptron

University of Guilan

Perceptron Learning Algorithm

- Initialize weights, bias, and a learning rate $0 \leq \alpha \leq 1$

- Calculate the net value $\boldsymbol{y_{in}}$

$$y_{in} = b + \sum x_i w_i$$

- Apply the AF over the net input to obtain an output
  - We might need a threshold θ

$$y = \begin{cases} 1 & if\ y_{in} > \theta \\ 0 & if\ -\theta < y_{in} < \theta \\ -1 & if\ y_{in} < -\theta \end{cases}$$

Perceptron

Perceptron Learning Algorithm

- Now, compare the desired target value $\boldsymbol{t}$ and the calculated output $\boldsymbol{y}$

- If $\boldsymbol{y} \neq \boldsymbol{t}$ then update the weights and bias (as below):

$$w_i(new) = w_i(old) + \alpha x_i t \qquad\qquad b(new) = b(old) + \alpha t$$

- Else, no need to update:

$$w_i(new) = w_i(old) \qquad\qquad b(new) = b(old)$$

- Continue the iteration <u>until there is no weight change</u>

Perceptron

Sample: AND function with binary inputs and bipolar outputs

- Parameters (initialization): $weights(0,0,0)$ $\alpha = 1$ $\theta = 0.2$

Input			NET	Out	Target	Weight Changes			Weights		
X1	X2	1	y_{in}	y	t	Δw1	Δw2	Δb	W1	W2	b
1	1	1			1						
1	0	1			-1						
0	1	1			-1						
0	0	1			-1						

Perceptron

University of Guilan

Epoch #1

Sample: AND function with binary inputs and bipolar outputs

- Parameters (initialization): $weights(0,0,0)$ $\alpha = 1$ $\theta = 0.2$

$\sum x_i w_i$

Input			NET	Out	Target	Weight Changes			Weights		
X1	X2	1	yin	y	t	Δw1	Δw2	Δb	W1	W2	b
1	1	1	**0**		1						
1	0	1			-1						
0	1	1			-1						
0	0	1			-1						

Perceptron

Epoch #1

Sample: AND function with binary inputs and bipolar outputs

- Parameters (initialization): $weights(0,0,0)$ $\alpha = 1$ $\theta = 0.2$

$\sum x_i w_i$

Input			NET	Out	Target	Weight Changes			Weights		
X1	X2	1	yin	y	t	Δw1	Δw2	Δb	W1	W2	b
1	1	1	0	**0**	1						
1	0	1			-1						
0	1	1			-1						
0	0	1			-1						

$$y = \begin{cases} 1 & if\ y_{in} > \theta \\ 0 & if\ -\theta < y_{in} < \theta \\ -1 & if\ y_{in} < -\theta \end{cases}$$

Perceptron

University of Guilan

Epoch #1

Sample: AND function with binary inputs and bipolar outputs

- Parameters (initialization): $weights(0,0,0)$ $\alpha = 1$ $\theta = 0.2$

$\sum x_i w_i$

Input			NET	Out	Target	Weight Changes			Weights		
X1	X2	1	yin	y	t	Δw1	Δw2	Δb	W1	W2	b
1	1	1	0	0	1						
1	0	1			-1						
0	1	1			-1						
0	0	1			-1						

Not equal! So we need to update weights

$$y = \begin{cases} 1 & if\ y_{in} > \theta \\ 0 & if\ -\theta < y_{in} < \theta \\ -1 & if\ y_{in} < -\theta \end{cases}$$

Perceptron

Epoch #1

Sample: AND function with binary inputs and bipolar outputs

- Parameters (initialization): $weights(0,0,0)$ $\alpha = 1$ $\theta = 0.2$

$\sum x_i w_i$

$\Delta w_i = \alpha x_i t$

Input			NET	Out	Target	Weight Changes			Weights		
X1	X2	1	yin	y	t	Δw1	Δw2	Δb	W1	W2	b
1	1	1	0	0	1	1	1	1			
1	0	1			-1						
0	1	1			-1						
0	0	1			-1						

$$y = \begin{cases} 1 & if\ y_{in} > \theta \\ 0 & if\ -\theta < y_{in} < \theta \\ -1 & if\ y_{in} < -\theta \end{cases}$$

Perceptron

Epoch #1

Sample: AND function with binary inputs and bipolar outputs

- Parameters (initialization): $weights(0,0,0)$ $\alpha = 1$ $\theta = 0.2$

$\sum x_i w_i$ $\Delta w_i = \alpha x_i t$ $w_i(new) = w_i(old) + \alpha x_i t$

Input			NET	Out	Target	Weight Changes			Weights		
X1	X2	1	yin	y	t	Δw1	Δw2	Δb	W1	W2	b
1	1	1	0	0	1	1	1	1	1	1	1
1	0	1			-1						
0	1	1			-1						
0	0	1			-1						

$$y = \begin{cases} 1 & if\ y_{in} > \theta \\ 0 & if\ -\theta < y_{in} < \theta \\ -1 & if\ y_{in} < -\theta \end{cases}$$

Perceptron

University of Guilan

Epoch #1

Sample: AND function with binary inputs and bipolar outputs

- Parameters (initialization): $weights(0,0,0)$ $\alpha = 1$ $\theta = 0.2$

$\sum x_i w_i$ $\Delta w_i = \alpha x_i t$ $w_i(new) = w_i(old) + \alpha x_i t$

Input			NET	Out	Target	Weight Changes			Weights		
X1	X2	1	yin	y	t	Δw1	Δw2	Δb	W1	W2	b
1	1	1	0	0	1	1	1	1	1	1	1
1	0	1	2	1	-1	-1	0	-1	0	1	0
0	1	1			-1						
0	0	1			-1						

$$y = \begin{cases} 1 & if\ y_{in} > \theta \\ 0 & if\ -\theta < y_{in} < \theta \\ -1 & if\ y_{in} < -\theta \end{cases}$$

Perceptron

Epoch #1

Sample: AND function with binary inputs and bipolar outputs

- Parameters (initialization): $weights(0,0,0)$ $\alpha = 1$ $\theta = 0.2$

$\sum x_i w_i$ $\Delta w_i = \alpha x_i t$ $w_i(new) = w_i(old) + \alpha x_i t$

Input			NET	Out	Target	Weight Changes			Weights		
X1	X2	1	yin	y	t	Δw1	Δw2	Δb	W1	W2	b
1	1	1	0	0	1	1	1	1	1	1	1
1	0	1	2	1	-1	-1	0	-1	0	1	0
0	1	1	1	1	-1	0	-1	-1	0	0	-1
0	0	1			-1						

$$y = \begin{cases} 1 & if\ y_{in} > \theta \\ 0 & if\ -\theta < y_{in} < \theta \\ -1 & if\ y_{in} < -\theta \end{cases}$$

Perceptron

Epoch #1

Sample: AND function with binary inputs and bipolar outputs

- Parameters (initialization): $weights(0,0,0)$ $\alpha = 1$ $\theta = 0.2$

$\sum x_i w_i$ $\Delta w_i = \alpha x_i t$ $w_i(new) = w_i(old) + \alpha x_i t$

Input			NET	Out	Target	Weight Changes			Weights		
X1	X2	1	yin	y	t	Δw1	Δw2	Δb	W1	W2	b
1	1	1	0	0	1	1	1	1	1	1	1
1	0	1	2	1	-1	-1			0	1	0
0	1	1	1	1	-1	0			0	0	-1
0	0	1	-1	**-1**	**-1**						

Equal! So no need to update weights

$$y = \begin{cases} 1 & if\ y_{in} > \theta \\ 0 & if\ -\theta < y_{in} < \theta \\ -1 & if\ y_{in} < -\theta \end{cases}$$

Perceptron

Epoch #1

Sample: AND function with binary inputs and bipolar outputs

- Parameters (initialization): $weights(0,0,0)$ $\alpha = 1$ $\theta = 0.2$

$\sum x_i w_i$ $\Delta w_i = \alpha x_i t$ $w_i(new) = w_i(old) + \alpha x_i t$

Input			NET	Out	Target	Weight Changes			Weights		
X1	X2	1	yin	y	t	Δw1	Δw2	Δb	W1	W2	b
1	1	1	0	0	1	1	1	1	1	1	1
1	0	1	2	1	-1	-1	0	-1	0	1	0
0	1	1	1	1	-1	0	-1	-1	0	0	-1
0	0	1	-1	**-1**	**-1**	0	0	0	0	0	-1

$$y = \begin{cases} 1 & if\ y_{in} > \theta \\ 0 & if\ -\theta < y_{in} < \theta \\ -1 & if\ y_{in} < -\theta \end{cases}$$

Perceptron

Epoch #1

Sample: AND function with binary inputs and bipolar outputs

- Parameters (initialization): $weights(0,0,0)$ $\alpha = 1$ $\theta = 0.2$

$\sum x_i w_i$ $\Delta w_i = \alpha x_i t$ $w_i(new) = w_i(old) + \alpha x_i t$

Input			NET	Out	Target	Weight Changes			Weights		
X1	X2	1	yin	y	t	Δw1	Δw2	Δb	W1	W2	b
1	1	1	0	0	1	1	1	1	1	1	1
1	0	1	2	1	-1	-1	0	-1	0	1	0
0	1	1	1	1	-1	0	-1	-1	0	0	-1
0	0	1	-1	-1	-1	0	0	0	0	0	-1

$$y = \begin{cases} 1 & if\ y_{in} > \theta \\ 0 & if\ -\theta < y_{in} < \theta \\ -1 & if\ y_{in} < -\theta \end{cases}$$

Init weights of the next epoch

Perceptron

Epoch #2

Sample: AND function with binary inputs and bipolar outputs

- Parameters (initialization): $weights(0,0,-1)$ $\alpha = 1$ $\theta = 0.2$

$\sum x_i w_i$ $\Delta w_i = \alpha x_i t$ $w_i(new) = w_i(old) + \alpha x_i t$

Input			NET	Out	Target	Weight Changes			Weights		
X1	X2	1	yin	y	t	Δw1	Δw2	Δb	W1	W2	b
1	1	1			1						
1	0	1			-1						
0	1	1			-1						
0	0	1			-1						

$$y = \begin{cases} 1 & if\ y_{in} > \theta \\ 0 & if\ -\theta < y_{in} < \theta \\ -1 & if\ y_{in} < -\theta \end{cases}$$

Perceptron

Epoch #2

Sample: AND function with binary inputs and bipolar outputs

- Parameters (initialization): $weights(0,0,-1)$ $\alpha = 1$ $\theta = 0.2$

$\sum x_i w_i$ $\Delta w_i = \alpha x_i t$ $w_i(new) = w_i(old) + \alpha x_i t$

Input			NET	Out	Target	Weight Changes			Weights		
X1	X2	1	yin	y	t	Δw1	Δw2	Δb	W1	W2	b
1	1	1	-1	-1	1	1	1	1	1	1	0
1	0	1	1	1	-1	-1	0	-1	0	1	-1
0	1	1	0	0	-1	0	-1	-1	0	0	-2
0	0	1	-2	**-1**	**-1**	0	0	0	0	0	-2

$$y = \begin{cases} 1 & if\ y_{in} > \theta \\ 0 & if\ -\theta < y_{in} < \theta \\ -1 & if\ y_{in} < -\theta \end{cases}$$

Init weights of the next epoch

Perceptron

Epoch #3

Sample: AND function with binary inputs and bipolar outputs

- Parameters (initialization): $weights(0,0,-2)$ $\alpha = 1$ $\theta = 0.2$

$\sum x_i w_i$ $\Delta w_i = \alpha x_i t$ $w_i(new) = w_i(old) + \alpha x_i t$

Input			NET	Out	Target	Weight Changes			Weights		
X1	X2	1	yin	y	t	Δw1	Δw2	Δb	W1	W2	b
1	1	1			1						
1	0	1			-1						
0	1	1			-1						
0	0	1			-1						

$$y = \begin{cases} 1 & if\ y_{in} > \theta \\ 0 & if\ -\theta < y_{in} < \theta \\ -1 & if\ y_{in} < -\theta \end{cases}$$

Perceptron

Epoch #3

Sample: AND function with binary inputs and bipolar outputs

- Parameters (initialization): $weights(0,0,-2)$ $\alpha = 1$ $\theta = 0.2$

$\sum x_i w_i$ $\Delta w_i = \alpha x_i t$ $w_i(new) = w_i(old) + \alpha x_i t$

Input			NET	Out	Target	Weight Changes			Weights		
X1	X2	1	yin	y	t	Δw1	Δw2	Δb	W1	W2	b
1	1	1	-2	-1	1	1	1	1	1	1	-1
1	0	1	0	0	-1	-1	0	-1	0	1	-2
0	1	1	-1	**-1**	**-1**	0	0	0	0	1	-2
0	0	1	-2	**-1**	**-1**	0	0	0	0	1	-2

$$y = \begin{cases} 1 & if\ y_{in} > \theta \\ 0 & if\ -\theta < y_{in} < \theta \\ -1 & if\ y_{in} < -\theta \end{cases}$$

Perceptron

University of Guilan

Perceptron

Epoch #10

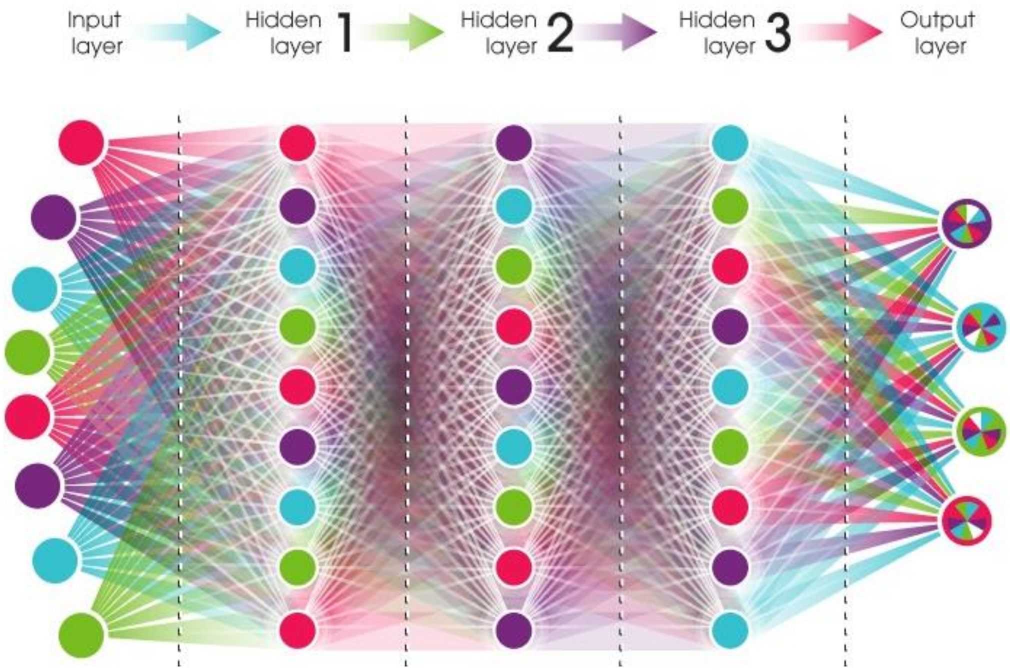
Sample: AND function with binary inputs and bipolar outputs

Input			NET	Out	Target	Weight Changes			Weights		
X1	X2	1	y_{in}	y	t	Δw1	Δw2	Δb	W1	W2	b
1	1	1	1	**1**	**1**	0	0	0	2	3	-4
1	0	1	-2	**-1**	**-1**	0	0	0	2	3	-4
0	1	1	-1	**-1**	**-1**	0	0	0	2	3	-4
0	0	1	-4	**-1**	**-1**	0	0	0	2	3	-4

Termination Condition ☺

What’s Next?

- Deep Neural Networks (DNNs)



Questions?