school of ai

Fundamentals of Artificial Neural Networks

2/3

Agenda

Recap last session on perceptrons

Learning in a perceptron

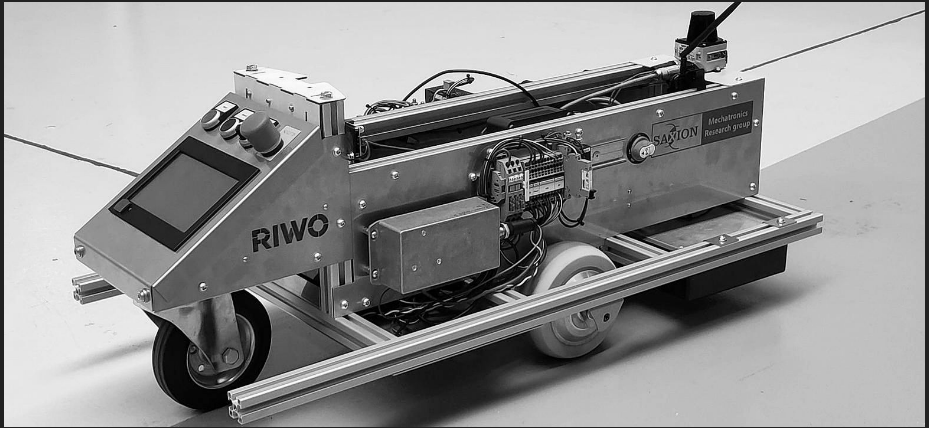
“Normal” ANNs

Wilco Bonestroo

Researcher at Mechatronics research group at Saxion

Software and AI for robots

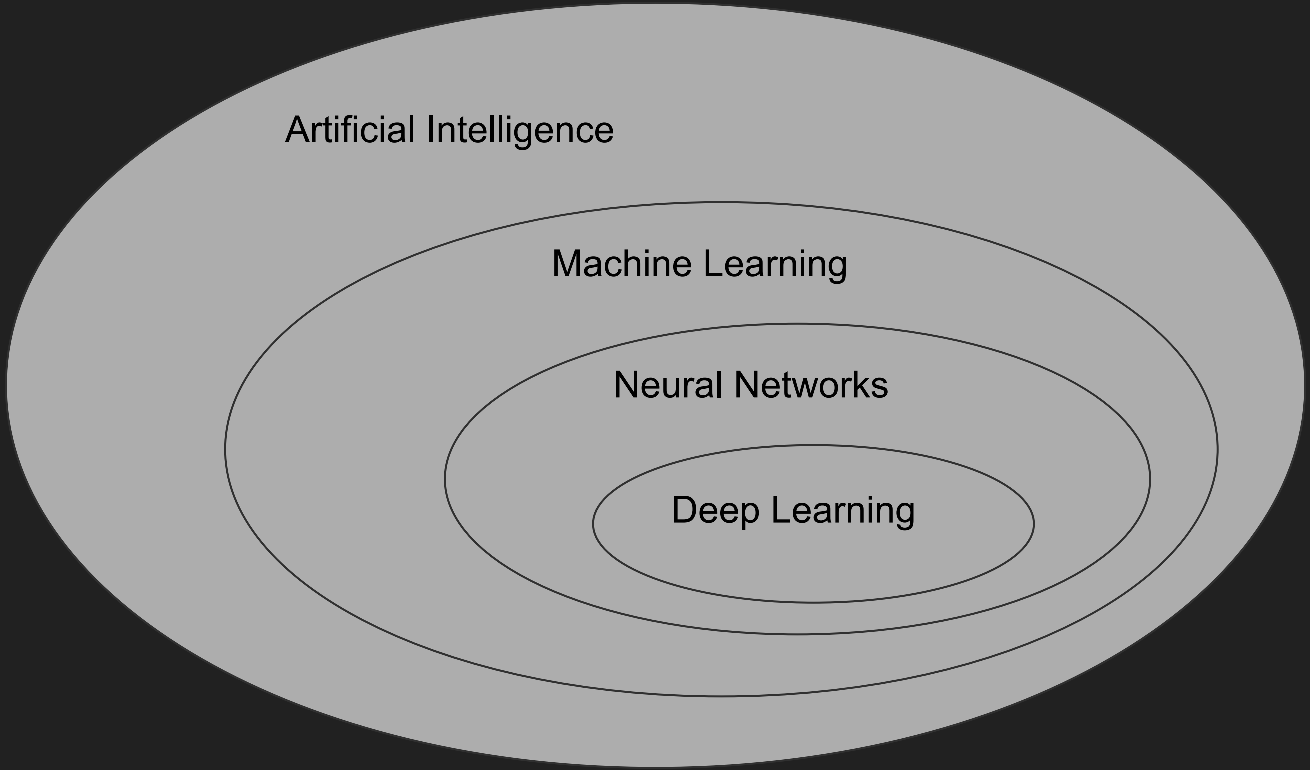
Autonomous mobile robots



Research group

- Unmanned Robotics Systems
- Smart Industrial Systems

The bigger picture



ANN concepts

Node

Link

Input

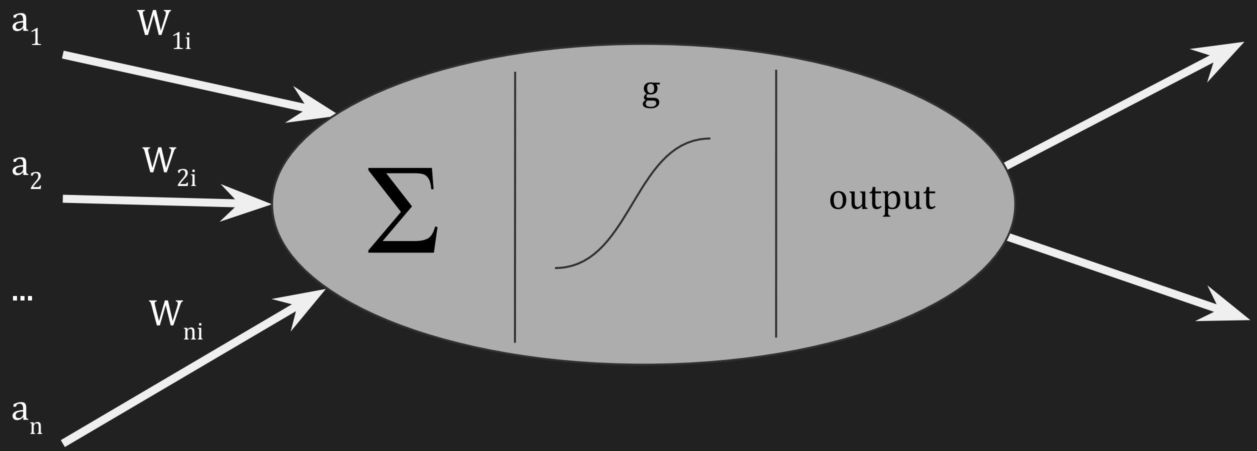
Weight

Activation function

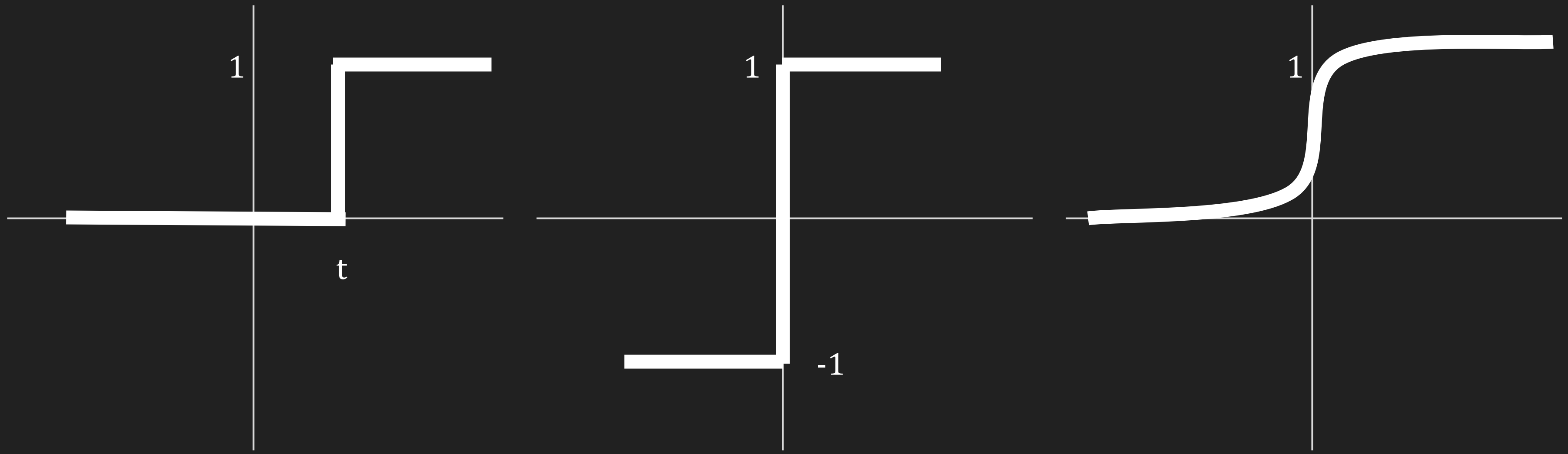
Output

Layer

“Knowledge”



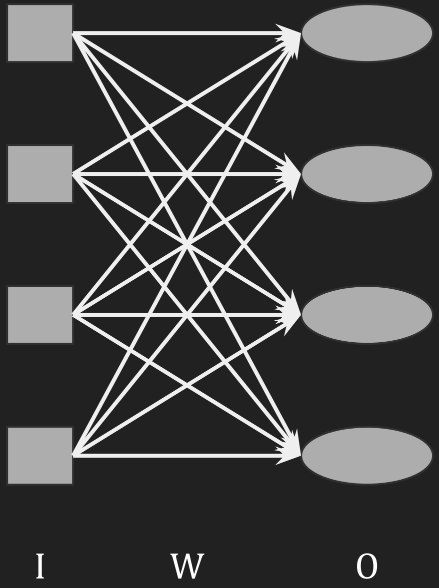
Common Activation Functions

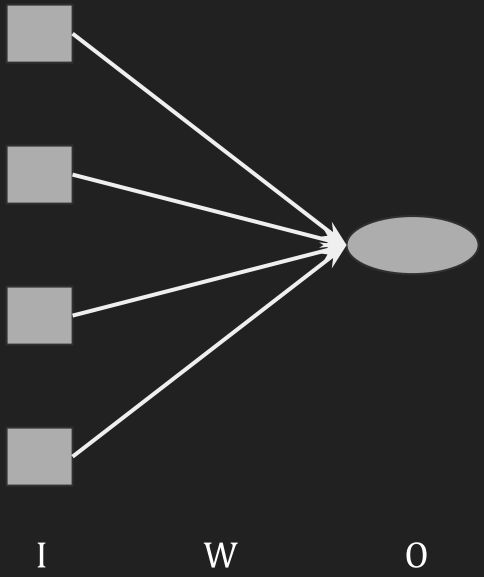


Step function Sign function Sigmoid function

Perceptrons

No hidden layers

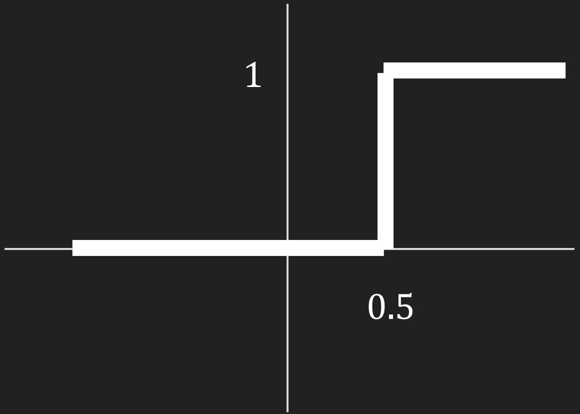




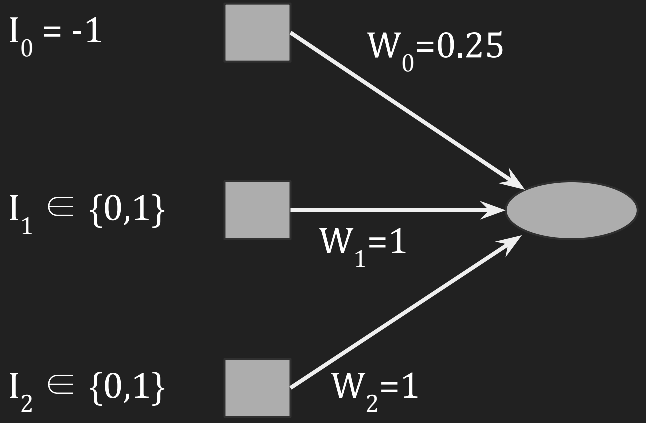
$$O = \mathrm{Step}_0\left(\sum W_j I_j\right) = \mathrm{Step}_0(\mathbf{W} \cdot \mathbf{I})$$

$$I_1 * W_1 + I_2 * W_2 + \ldots$$

Represent boolean functions



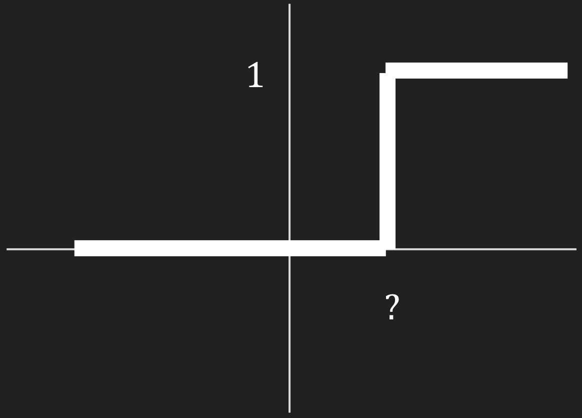
Step function



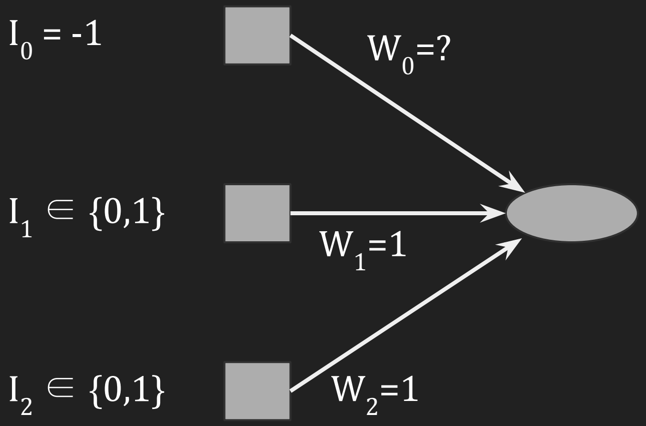
I_1	I_2	O
0	0	
0	1	
1	0	
1	1	

More convenient: $W \times I > 0$

Represent boolean functions



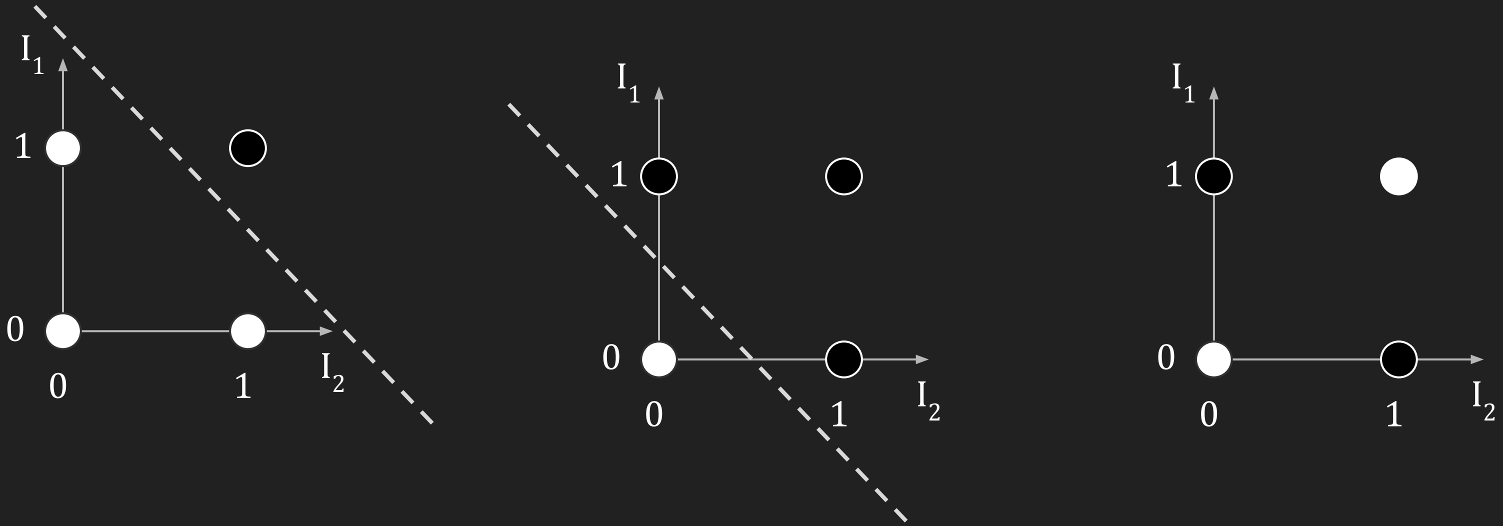
Step function



XOR

I_1	I_2	O
0	0	0
0	1	1
1	0	1
1	1	0

Linear Separable (two inputs / dimensions)



Learning in a perceptron

Find an algorithm that can “train” a network based on given examples.

Perceptron learning rule by Rosenblat (1960)

I_1	I_2	O
0	0	0
0	1	0
1	0	0
1	1	1

Generic Network Learning Algorithm

Initialise network (weights)

Repeat

  For each example

    Determine outcome of current network for example

    Compare outcome to expected value for example

    Update the network

  end

Until all examples are correctly classified

Generic Network Learning

Initialise network (weights)

Repeat

    For each example

        Determine outcome of current network for example

        Compare outcome to expected value for example

        Update the network

    end

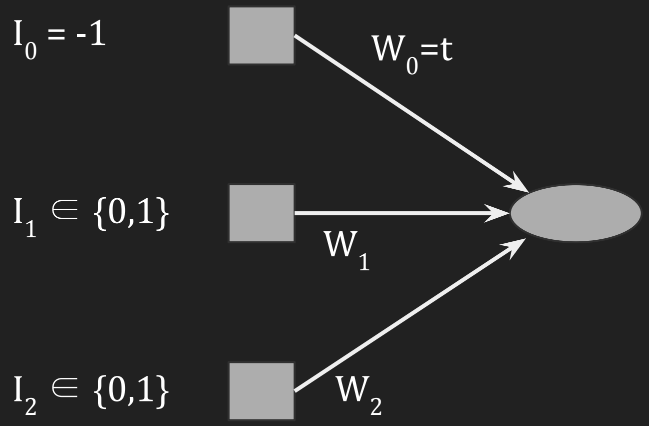
Until all examples are correctly classified

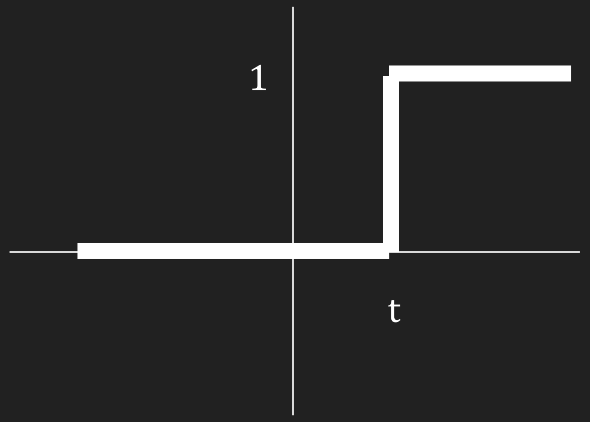
Examples (AND function)

W_0	W_1	W_2
0	0	0

I_0	I_1	I_2
-1	1	1

Output	Expected





Step function

Examples (AND function)

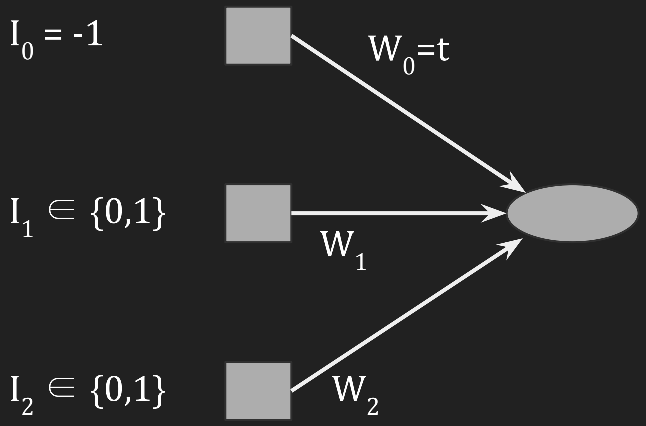
W_0	W_1	W_2
0	0	0

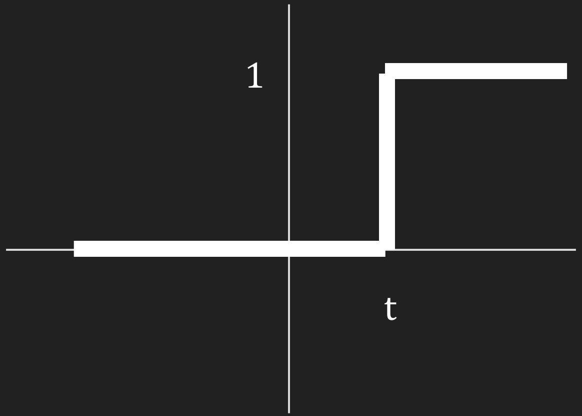
I_0	I_1	I_2
-1	1	1

Output	Expected
0	1

If the output should be **higher**, we should **add** the values of I_j to W_j

Short notation: W becomes $W + \alpha \times I$





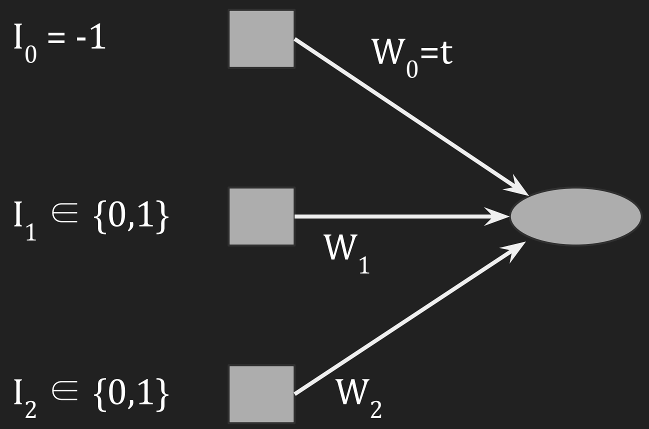
Step function

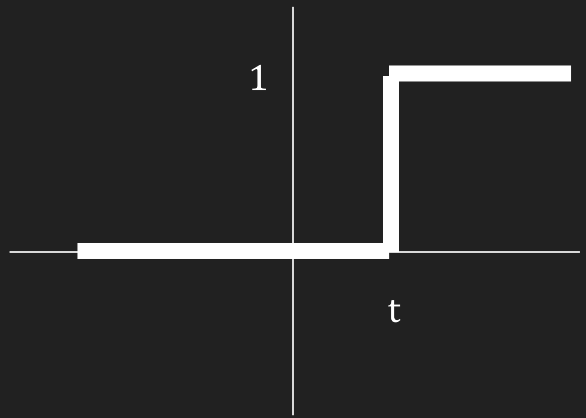
Examples (AND function)

W_0	W_1	W_2
-1	0	0

I_0	I_1	I_2
-1	0	1

Output	Expected





Step function

Examples (AND function)

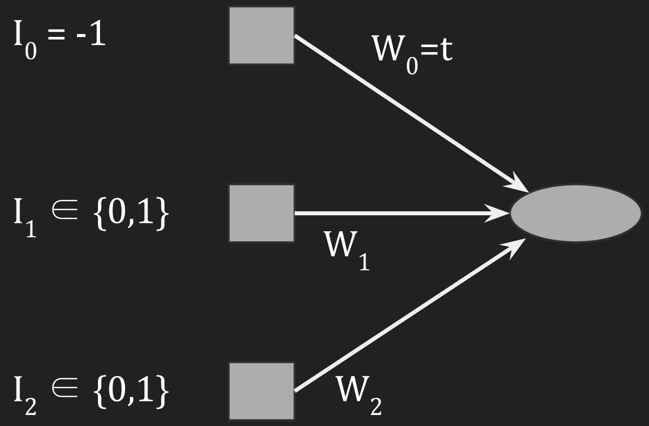
W_0	W_1	W_2
-1	0	0

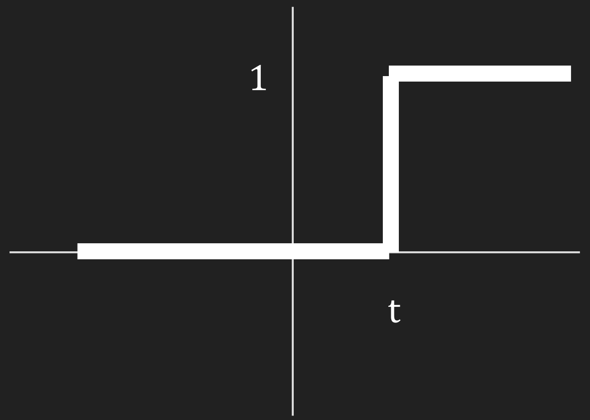
I_0	I_1	I_2
-1	0	1

Output	Expected
1	0

If the output should be lower, we should subtract the values of I_j from W_j

Short notation: W becomes $W - \alpha \times I$





Step function

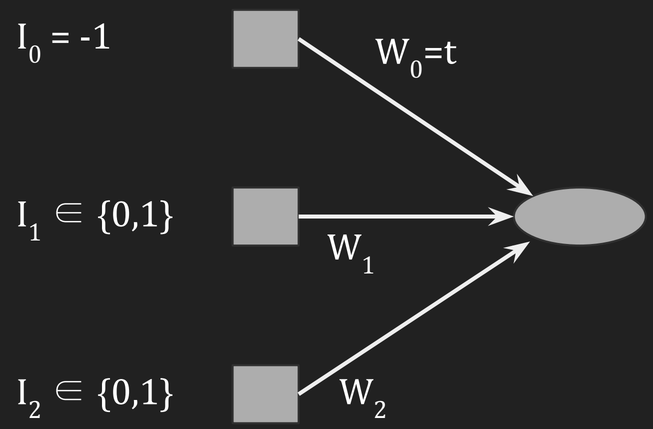
The actual output is O

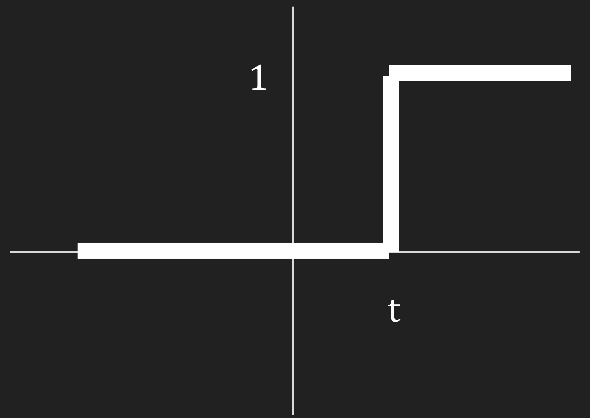
The expected output is T

The error Err = T - O

Learning Rule:

W_j becomes $W_j + \alpha \times I_j \times Err$





Step function

OR function

W_0	W_1	W_2
0	0	0

I_0	I_1	I_2
-1	0	0

Output	Expected	

OR function

W_0	W_1	W_2
0	0	0

I_0	I_1	I_2
-1	0	0

Output	Expected	
0	0	👍

OR function

W_0	W_1	W_2
0	0	0
0	0	0

I_0	I_1	I_2
-1	0	0
-1	0	1

Output	Expected	
0	0	👍

OR function

W_0	W_1	W_2
0	0	0
0	0	0

I_0	I_1	I_2
-1	0	0
-1	0	1

Output	Expected	
0	0	👍
0	1	😡

OR function

W_0	W_1	W_2
0	0	0
0	0	0
-1	0	1

I_0	I_1	I_2
-1	0	0
-1	0	1

Output	Expected	
0	0	👍
0	1	😡

OR function

W_0	W_1	W_2
0	0	0
0	0	0
-1	0	1

I_0	I_1	I_2
-1	0	0
-1	0	1
-1	0	1

Output	Expected	
0	0	👍
0	1	😡

OR function

W_0	W_1	W_2
0	0	0
0	0	0
-1	0	1

I_0	I_1	I_2
-1	0	0
-1	0	1
-1	0	1

Output	Expected	
0	0	👍
0	1	😡
1	1	👍

OR function

W_0	W_1	W_2
0	0	0
0	0	0
-1	0	1
-1	0	1

I_0	I_1	I_2
-1	0	0
-1	0	1
-1	0	1
-1	1	0

Output	Expected	
0	0	👍
0	1	😡
1	1	👍

OR function

W_0	W_1	W_2
0	0	0
0	0	0
-1	0	1
-1	0	1

I_0	I_1	I_2
-1	0	0
-1	0	1
-1	0	1
-1	1	0

Output	Expected	
0	0	👍
0	1	😡
1	1	👍
1	1	👍

OR function

W_0	W_1	W_2
0	0	0
0	0	0
-1	0	1
-1	0	1
-1	0	1

I_0	I_1	I_2
-1	0	0
-1	0	1
-1	0	1
-1	1	0
-1	1	1

Output	Expected	
0	0	👍
0	1	😠
1	1	👍
1	1	👍

OR function

W_0	W_1	W_2
0	0	0
0	0	0
-1	0	1
-1	0	1
-1	0	1

I_0	I_1	I_2
-1	0	0
-1	0	1
-1	0	1
-1	1	0
-1	1	1

Output	Expected	
0	0	👍
0	1	😡
1	1	👍
1	1	👍
1	1	👍

OR function

W_0	W_1	W_2
0	0	0
0	0	0
-1	0	1
-1	0	1
-1	0	1
-1	0	1

I_0	I_1	I_2
-1	0	0
-1	0	1
-1	0	1
-1	1	0
-1	1	1
-1	0	0

Output	Expected	
0	0	👍
0	1	😡
1	1	👍
1	1	👍
1	1	👍

OR function

W_0	W_1	W_2
0	0	0
0	0	0
-1	0	1
-1	0	1
-1	0	1
-1	0	1

I_0	I_1	I_2
-1	0	0
-1	0	1
-1	0	1
-1	1	0
-1	1	1
-1	0	0

Output	Expected	
0	0	👍
0	1	😡
1	1	👍
1	1	👍
1	1	👍
1	0	😡

OR function

W_0	W_1	W_2
0	0	0
0	0	0
-1	0	1
-1	0	1
-1	0	1
-1	0	1
0	0	1

I_0	I_1	I_2
-1	0	0
-1	0	1
-1	0	1
-1	1	0
-1	1	1
-1	0	0
-1	0	0

Output	Expected	
0	0	👍
0	1	😡
1	1	👍
1	1	👍
1	1	👍
1	0	😡

OR function

W_0	W_1	W_2
0	0	0
0	0	0
-1	0	1
-1	0	1
-1	0	1
-1	0	1
0	0	1

I_0	I_1	I_2
-1	0	0
-1	0	1
-1	0	1
-1	1	0
-1	1	1
-1	0	0
-1	0	0

Output	Expected	
0	0	👍
0	1	😡
1	1	👍
1	1	👍
1	1	👍
1	0	😡
0	0	👍

OR function

W_0	W_1	W_2
0	0	1

I_0	I_1	I_2
-1	0	0

Output	Expected	
0	0	👍

OR function

W_0	W_1	W_2
0	0	1
0	0	1

I_0	I_1	I_2
-1	0	0
-1	0	1

Output	Expected	
0	0	👍

OR function

W_0	W_1	W_2
0	0	1
0	0	1

I_0	I_1	I_2
-1	0	0
-1	0	1

Output	Expected	
0	0	👍
1	1	👍

OR function

W_0	W_1	W_2
0	0	1
0	0	1
0	0	1

I_0	I_1	I_2
-1	0	0
-1	0	1
-1	1	0

Output	Expected	
0	0	👍
1	1	👍

OR function

W_0	W_1	W_2
0	0	1
0	0	1
0	0	1

I_0	I_1	I_2
-1	0	0
-1	0	1
-1	1	0

Output	Expected	
0	0	👍
1	1	👍
0	1	😡

OR function

W_0	W_1	W_2
0	0	1
0	0	1
0	0	1
-1	1	1

I_0	I_1	I_2
-1	0	0
-1	0	1
-1	1	0
-1	1	0

Output	Expected	
0	0	👍
1	1	👍
0	1	😡

OR function

W_0	W_1	W_2
0	0	1
0	0	1
0	0	1
-1	1	1

I_0	I_1	I_2
-1	0	0
-1	0	1
-1	1	0
-1	1	0

Output	Expected	
0	0	👍
1	1	👍
0	1	😡
1	1	👍

OR function

W_0	W_1	W_2
0	0	1
0	0	1
0	0	1
-1	1	1
-1	1	1

I_0	I_1	I_2
-1	0	0
-1	0	1
-1	1	0
-1	1	0
-1	1	1

Output	Expected	
0	0	👍
1	1	👍
0	1	😡
1	1	👍

OR function

W_0	W_1	W_2
0	0	1
0	0	1
0	0	1
-1	1	1
-1	1	1

I_0	I_1	I_2
-1	0	0
-1	0	1
-1	1	0
-1	1	0
-1	1	1

Output	Expected	
0	0	👍
1	1	👍
0	1	😡
1	1	👍
1	1	👍

OR function

W_0	W_1	W_2
0	0	1
0	0	1
0	0	1
-1	1	1
-1	1	1
-1	1	1

I_0	I_1	I_2
-1	0	0
-1	0	1
-1	1	0
-1	1	0
-1	1	1
-1	0	0

Output	Expected	
0	0	👍
1	1	👍
0	1	😡
1	1	👍
1	1	👍

OR function

W_0	W_1	W_2
0	0	1
0	0	1
0	0	1
-1	1	1
-1	1	1
-1	1	1

I_0	I_1	I_2
-1	0	0
-1	0	1
-1	1	0
-1	1	0
-1	1	1
-1	0	0

Output	Expected	
0	0	👍
1	1	👍
0	1	😡
1	1	👍
1	1	👍
1	0	😡

OR function

W_0	W_1	W_2
-1	1	1

I_0	I_1	I_2
-1	0	0

Output	Expected	
1	0	😡

OR function

W_0	W_1	W_2
-1	1	1
0	1	1

I_0	I_1	I_2
-1	0	0
-1	0	0

Output	Expected	
1	0	😡

OR function

W_0	W_1	W_2
-1	1	1
0	1	1

I_0	I_1	I_2
-1	0	0
-1	0	0

Output	Expected	
1	0	😡
0	0	😃

OR function

W_0	W_1	W_2
-1	1	1
0	1	1
0	1	1

I_0	I_1	I_2
-1	0	0
-1	0	0
-1	0	1

Output	Expected	
1	0	😡
0	0	😀

OR function

W_0	W_1	W_2
-1	1	1
0	1	1
0	1	1

I_0	I_1	I_2
-1	0	0
-1	0	0
-1	0	1

Output	Expected	
1	0	😡
0	0	😃
1	1	😃

OR function

W_0	W_1	W_2
-1	1	1
0	1	1
0	1	1
0	1	1

I_0	I_1	I_2
-1	0	0
-1	0	0
-1	0	1
-1	1	0

Output	Expected	
1	0	😡
0	0	😃
1	1	😃

OR function

W_0	W_1	W_2
-1	1	1
0	1	1
0	1	1
0	1	1

I_0	I_1	I_2
-1	0	0
-1	0	0
-1	0	1
-1	1	0

Output	Expected	
1	0	😡
0	0	😃
1	1	😃
1	1	😃

OR function

W_0	W_1	W_2
-1	1	1
0	1	1
0	1	1
0	1	1
0	1	1

I_0	I_1	I_2
-1	0	0
-1	0	0
-1	0	1
-1	1	0
-1	1	1

Output	Expected	
1	0	😡
0	0	😀
1	1	😀
1	1	😀

OR function

W_0	W_1	W_2
-1	1	1
0	1	1
0	1	1
0	1	1
0	1	1

I_0	I_1	I_2
-1	0	0
-1	0	0
-1	0	1
-1	1	0
-1	1	1

Output	Expected	
1	0	😡
0	0	😀
1	1	😀
1	1	😀
1	1	😀

OR function

W_0	W_1	W_2
-1	1	1
0	1	1
0	1	1
0	1	1
0	1	1

I_0	I_1	I_2
-1	0	0
-1	0	0
-1	0	1
-1	1	0
-1	1	1

Output	Expected	
1	0	😡
0	0	😀
1	1	😀
1	1	😀
1	1	😀

😀

For **linear separable** functions this algorithm **always** converges!

Most functions are not linearly separable :(

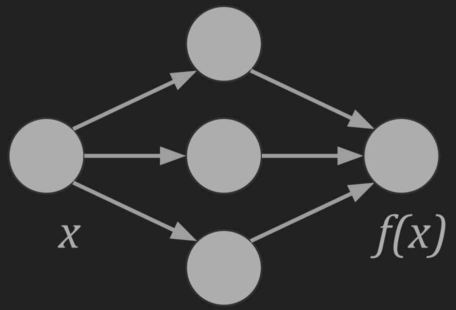
This is a sort of gradient descent, but not backpropagation!

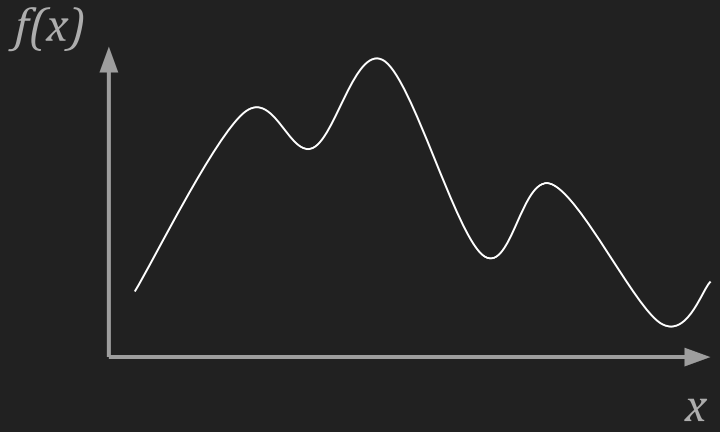
The approach we saw does not work for “normal” ANN with many layers and different activation functions...

How to represent other functions?

Add layers

“With one (sufficiently large) layer of hidden units, it is possible to represent any continuous function of the inputs.”





Resources

Book: Artificial Intelligence: A Modern Approach

Auteur: Stuart Russell