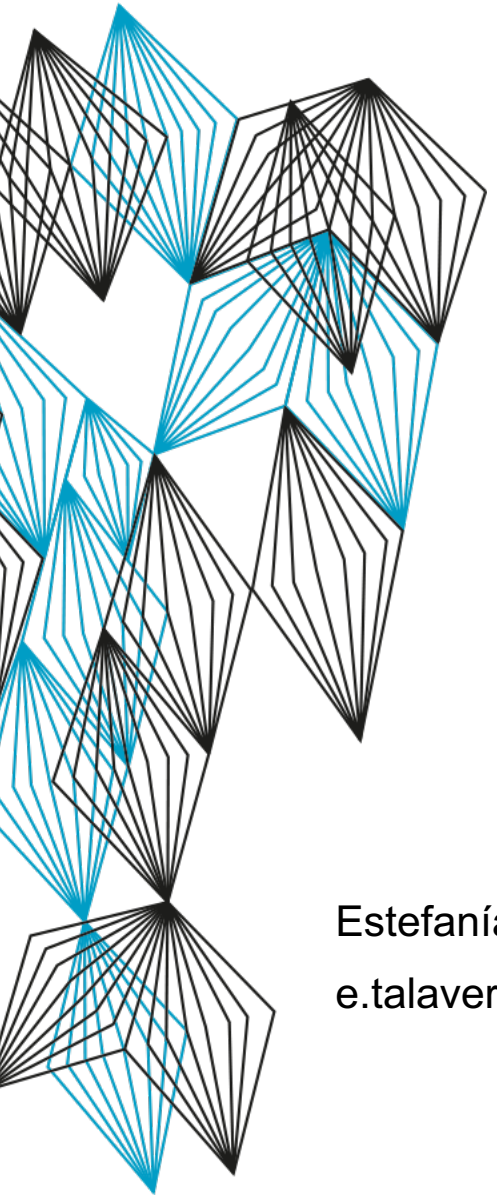


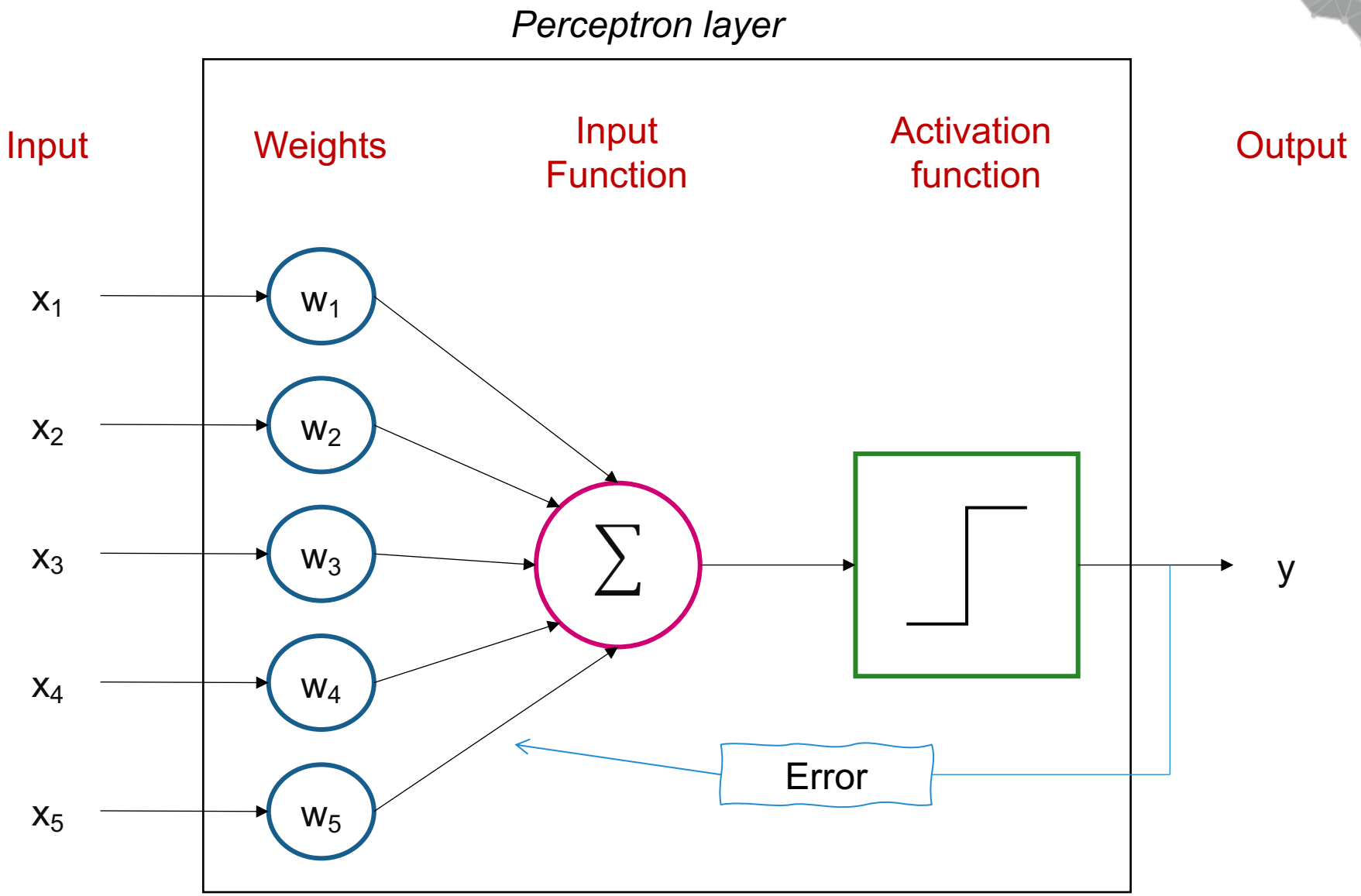
ARTIFICIAL INTELLIGENCE & CYBER SECURITY

NEURAL NETWORKS

Perceptron

Estefanía Talavera Martínez
e.talaveramartinez@utwente.nl

PERCEPTRON


Perceptron layer
Input
Weights
Input Function
Activation function
Output
x1
x2
x3
x4
x5
w1
w2
w3
w4
w5
∑
y
Error

PERCEPTRON LEARNING ALGORITHM

LINEAR PERCEPTRON

- Inspired by the brain
- Model of neurons:
  - Binary signals (transmit / don't transmit)
  - Multiple inputs, one output
  - Send a signal to the output if inputs are sufficiently activated

- Activation:

$$f(\mathbf{x}; \mathbf{w}) = h(\mathbf{w} \cdot \mathbf{x}), \text{ where } h(a) = \begin{cases} +1 & \text{if } a > 0 \\ -1 & \text{otherwise} \end{cases}$$

  is a (non-linear) step function.

- Training data set $\{\mathbf{x}_n, t_n\}$ uses the target coding scheme:

$$t_n = \begin{cases} +1 & \text{if } \mathbf{x}_n \text{ belongs to } \mathcal{C}_1 \\ -1 & \text{otherwise} \end{cases}$$

PERCEPTRON LEARNING ALGORITHM

LINEAR PERCEPTRON

- Inspired by the brain
- Model of neurons:
  - Binary signals (transmit / don't transmit)
  - Multiple inputs, one output
  - Send a sig uts are suffi
- Activation:

Function variables

Function parameters

Inner product
$\mathbf{x} \cdot \mathbf{w} = \sum_i x_i w_i$

$$f(\mathbf{x}; \mathbf{w}) = h(\mathbf{w} \cdot \mathbf{x}), \text{ where } h(a) = \begin{cases} +1 & \text{if } a > 0 \\ -1 & \text{otherwise} \end{cases}$$

is a (non-linear) step function.

- Training data set $\{\mathbf{x}_n, t_n\}$ uses the target coding scheme:

$$t_n = \begin{cases} +1 & \text{if } \mathbf{x}_n \text{ belongs to } \mathcal{C}_1 \\ -1 & \text{otherwise} \end{cases}$$

PERCEPTRON LEARNING ALGORITHM

TRAINING THE LINEAR PERCEPTRON

Classification is correct iff (if and only if):

- $t_n > 0$ and $\mathbf{w} \cdot x > 0$ (because then $f(\mathbf{w} \cdot x) > 0$), or
- $t_n < 0$ and $\mathbf{w} \cdot x < 0$

Therefore, classification is correct if

$$\mathbf{w} \cdot x_n t_n > 0$$

We want to minimise the misclassification rate

$$E(\boldsymbol{w}) = -\sum_{n \in \mathcal{M}} \mathbf{w} \cdot x_n t_n$$

where $\mathcal{M} = \{ x_i \mid \mathbf{w} \cdot x_i t_i \leq 0 \}$ is the set of misclassified samples.

PERCEPTRON LEARNING ALGORITHM

LEARNING ALGORITHM

- Training algorithm:

  cycle through the training patterns and if misclassified, update w as follows

Weights at iteration *i-1*

Weights at iteration *i*

$$\mathbf{w}^{(i)} = \mathbf{w}^{(i-1)} + x_n t_n$$

$\pm$ datapoint *n,* depending on the target class

- This was revolutionary, because it seemed plausible that neurons could really work like this.

Perceptron Convergence Theorem

If the training set is linearly separable,
the algorithm is guaranteed to find a solution in a finite number of steps.

PERCEPTRON LEARNING ALGORITHM

PERCEPTRONS

- Output: step function of linear combination of inputs
- Step function y(·) ⇒ non-linear

$$y(x) = h(\mathbf{w} \cdot \mathbf{x})$$

- Multiple layers would make complex functions possible
  - non-linear functions of non-linear functions
- Training of single layer is problematic
  - Convergence
  - non-separable training data
  - Solution depends on initialisation
- Training of multiple layers would be next to impossible

PERCEPTRON LEARNING ALGORITHM

PROBLEMS WITH PERCEPTRON

- Training is slow, and does not generalise easily to more than 2 classes
- As long as the algorithm hasn't converged, there is no way to know whether the problem is not linearly separable, or just slow to converge.
- Because the set of misclassified training examples changes at each weight update, the algorithm is not guaranteed to reduce the overall error at each step. If the data is not linearly separable, no particularly good solution may be found.
- In general, many solutions exist and the final solution depends on the initial conditions, not on some optimality criterion.
- We can learn any linear function, but how do we deal with non-linear problems?

PERCEPTRON LEARNING ALGORITHM

NEURAL NETWORK

By using a differentiable activation function, we can make training much easier

For example: logistic activation function:

$$\sigma(a) = \frac{1}{1 + \exp(-a)}$$