DATA MINING 2
Neural Networks (Perceptron)

Riccardo Guidotti

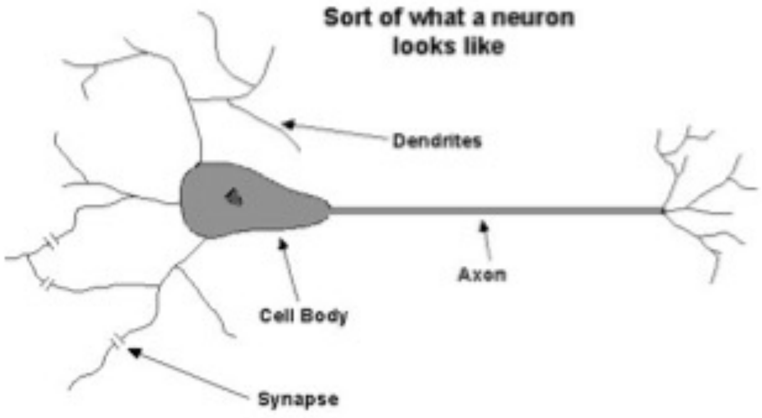
a.a. 2020/2021

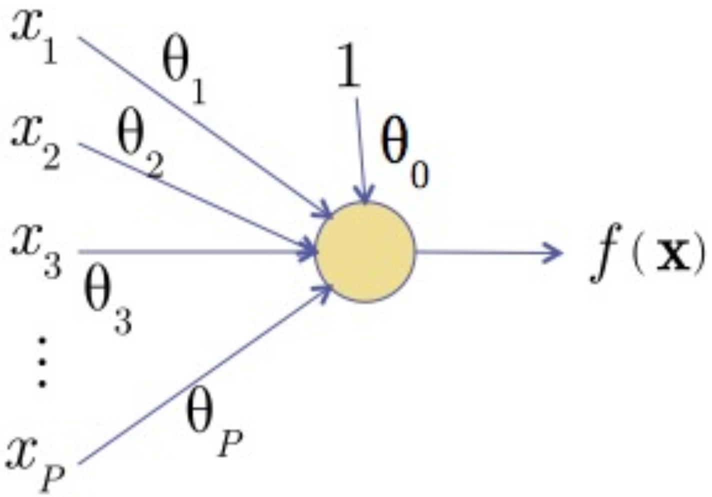
Slides edited from a set of slides titled "Introduction to Machine Learning and Neural Networks" by Davide Bacciu

UNIVERSITÀ DI PISA

The Neuron Metaphor

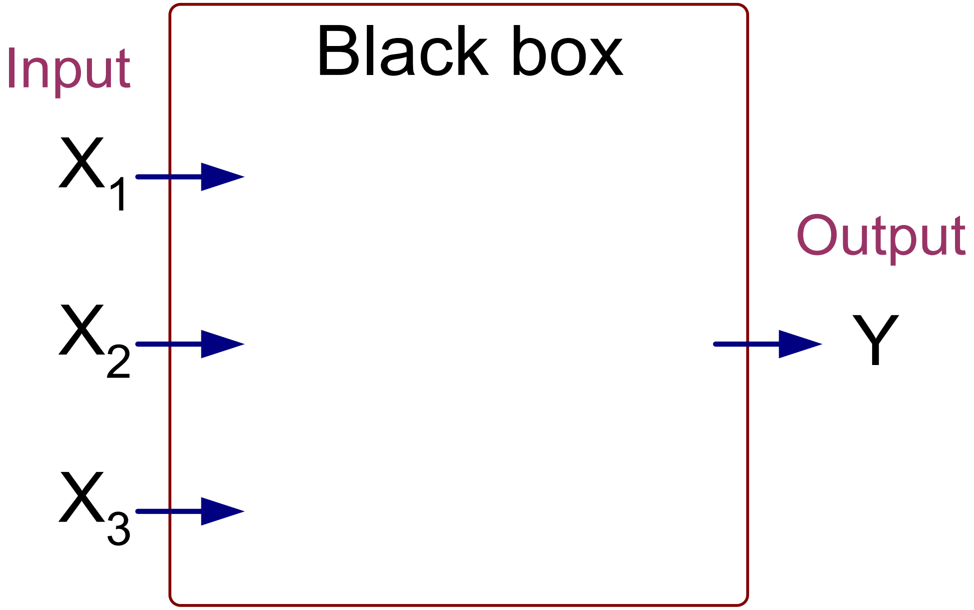
- Neurons
  - accept information from multiple inputs,
  - transmit information to other neurons.
- Multiply inputs by weights along edges
- Apply some function to the set of inputs at each node





Artificial Neural Networks (ANN)

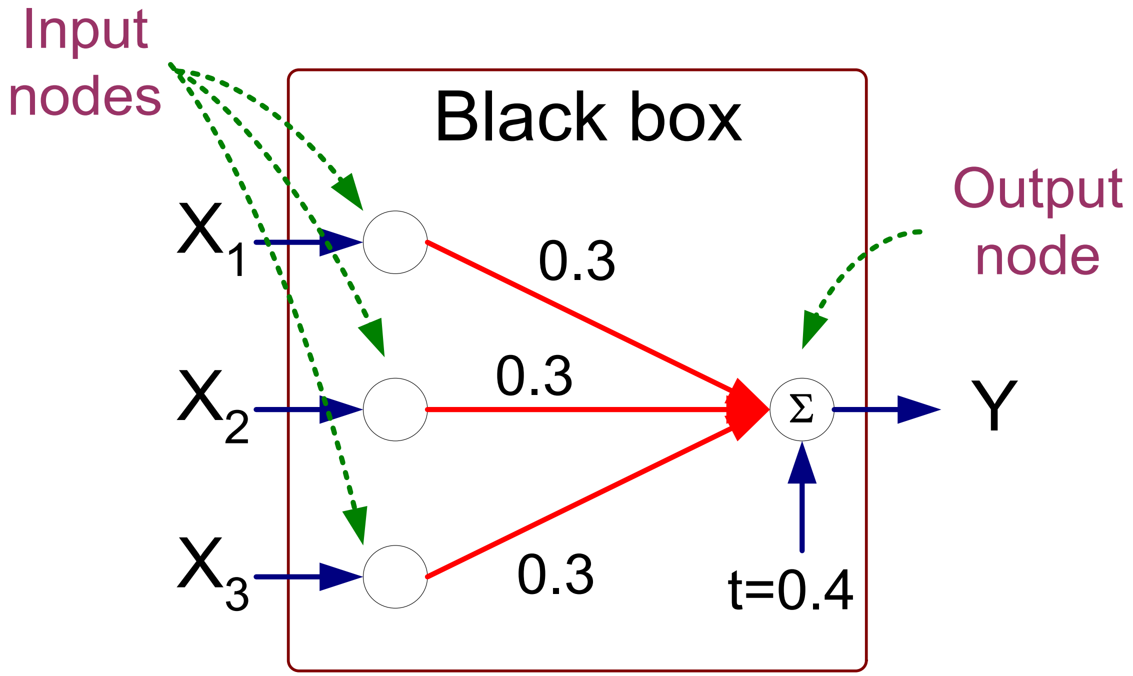
X_1	X_2	X_3	Y
1	0	0	-1
1	0	1	1
1	1	0	1
1	1	1	1
0	0	1	-1
0	1	0	-1
0	1	1	1
0	0	0	-1



Output Y is 1 if at least two of the three inputs are equal to 1.

Artificial Neural Networks (ANN)

X_1	X_2	X_3	Y
1	0	0	-1
1	0	1	1
1	1	0	1
1	1	1	1
0	0	1	-1
0	1	0	-1
0	1	1	1
0	0	0	-1



$$Y = sign(0.3X_1 + 0.3X_2 + 0.3X_3 - 0.4)$$

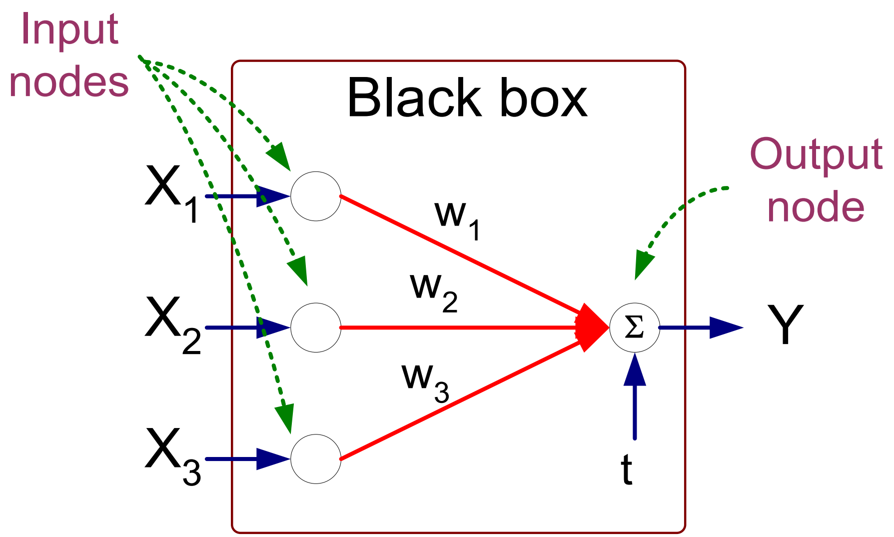
$$\text{where } sign(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ -1 & \text{if } x < 0 \end{cases}$$

Artificial Neural Networks (ANN)

- Model is an assembly of inter-connected nodes and weighted links

- Output node sums up each of its input value according to the weights of its links

- Compare output node against some threshold t (also named bias b)



$$Y = sign(\sum_{i=1}^{d} w_i X_i - t)$$

$$= sign(\sum_{i=0}^{d} w_i X_i)$$

Characterizing the Artificial Neuron

- Input/Output signal may be.
  - Real value.
  - Unipolar {0, 1}.
  - Bipolar {-1, +1}.
- **Weight** (w or sigma): ϑ_{ij} – strength of connection from unit *j* to unit *i*
- Learning amounts to **adjusting the weights** ϑ_{ij} by means of an **optimization algorithm** aiming to minimize a cost function, i.e., as in biological systems training a perceptron model amounts to adapting the weights of the links until they fit the input output relationships of the underlying data.

Characterizing the Artificial Neuron

- The bias *b* is a constant that can be written as $\vartheta_{i0}x_0$ with $x_0 = 1$ and ϑ_{i0} = b such that

$$net_i = \sum_{j=0}^{n} \vartheta_{ij} x_j$$

- The function $f(net_i(x))$ is the unit's **activation function**. In the simplest case, *f* is the identity function, and the unit's output is just its net input. This is called a ***linear unit.*** Otherwise we can have a ***sign unit,*** or a ***logistic unit.***

The Perceptron Classifier

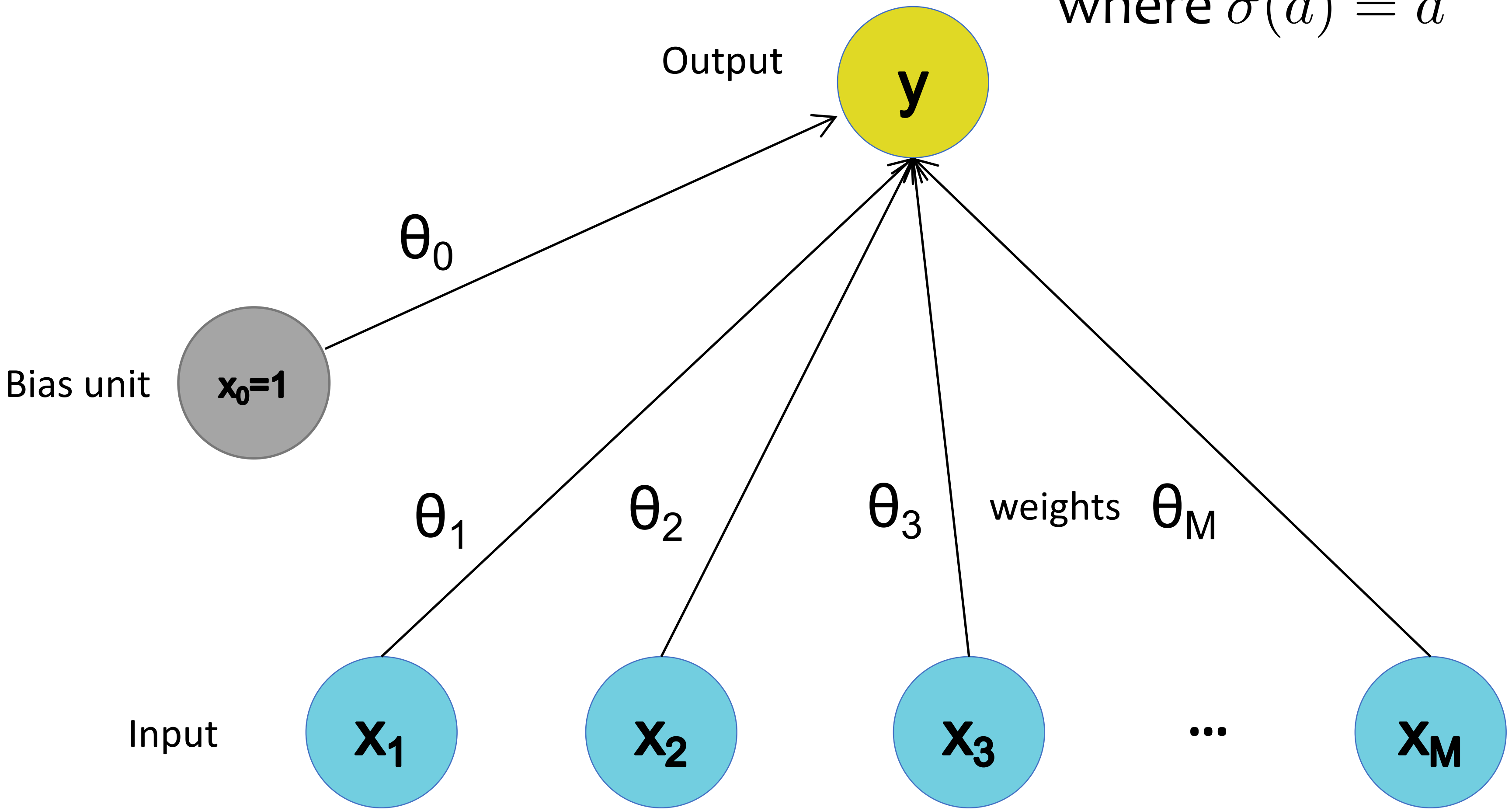
A Simple Linear Neuron

$$y = h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \sigma(\boldsymbol{\theta}^T \boldsymbol{x})$$

net

where $\sigma(a) = a$

Linear activation function

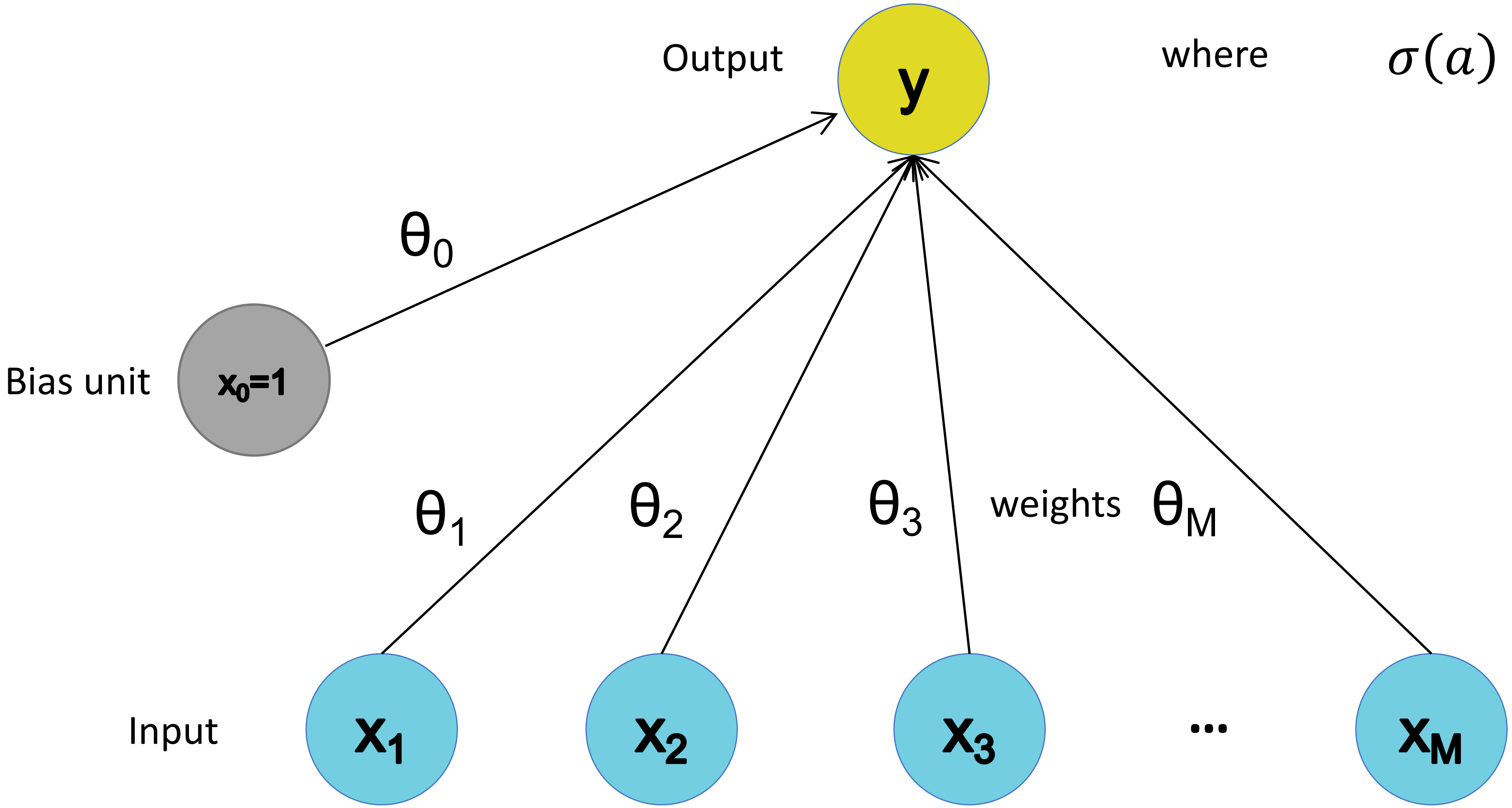


Linear Threshold Unit (a.k.a. Perceptron)

$$y = h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \sigma(\underbrace{\boldsymbol{\theta}^T \boldsymbol{x}}_{\text{net}})$$

Sign activation function

where $\sigma(a) = \begin{cases} +1 & a \geq 0 \\ -1 & a < 0 \end{cases}$



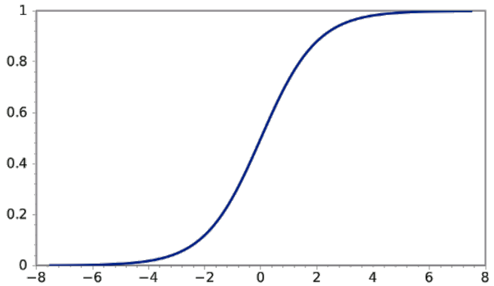
The Logistic Neuron

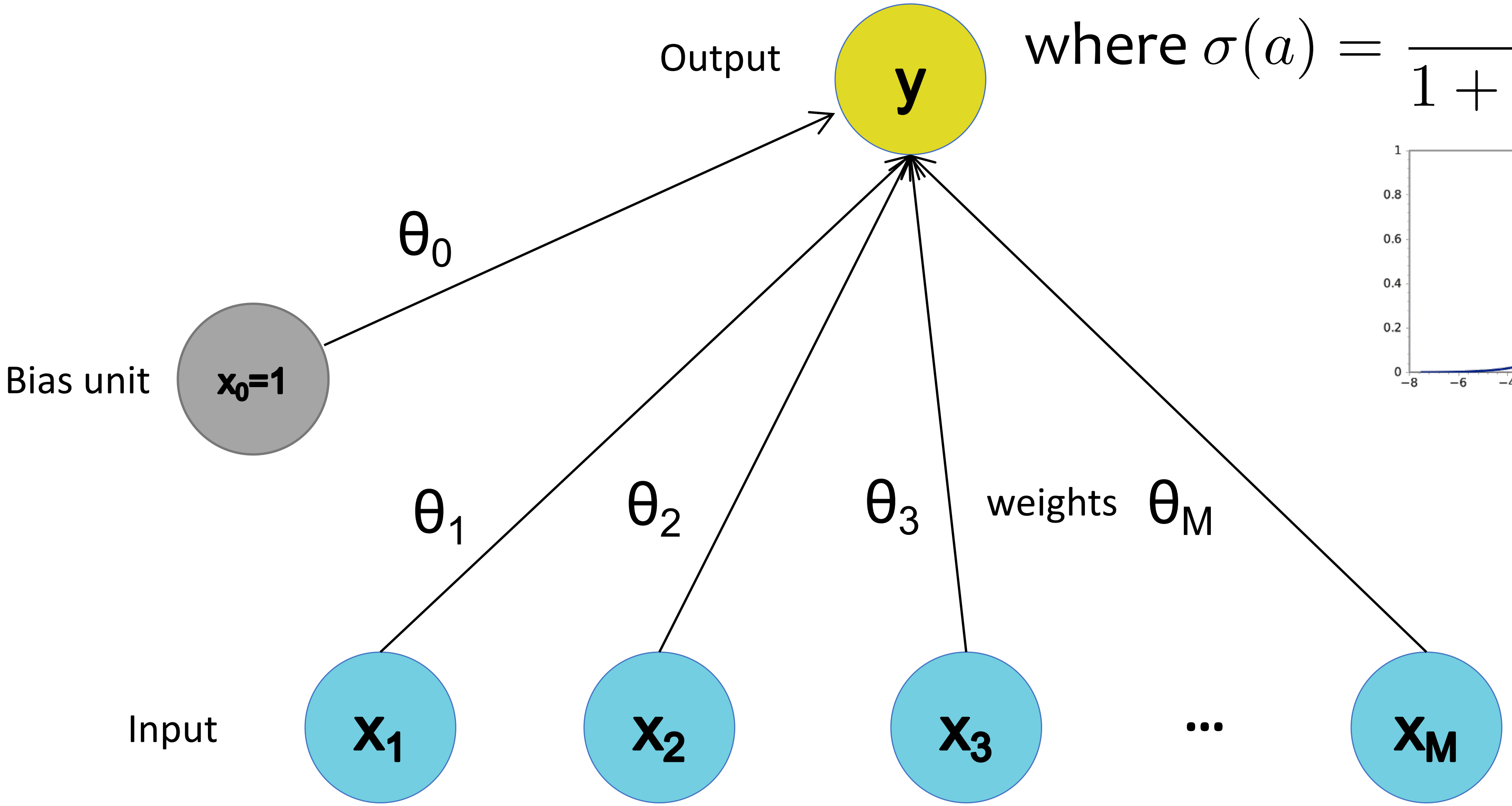
$$y = h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \sigma(\boldsymbol{\theta}^T \boldsymbol{x})$$

net

Sigmoid logistic activation function

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





Perceptron

- Single layer network
  - Contains only input and output nodes

- Activation function: f = sign(w•x)

- Applying model is straightforward

$$Y = sign(0.3X_1 + 0.3X_2 + 0.3X_3 - 0.4)$$

$$\text{where } sign(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ -1 & \text{if } x < 0 \end{cases}$$

  - $X_1 = 1$, $X_2 = 0$, $X_3 = 1$ => y = sign(0.2) = 1

Learning Iterative Procedure

- During the training phase the weight parameters are adjusted until the outputs of the perceptron become consistent with the true outputs of the training examples.
- Initialize the weights (w_0, w_1, …, w_m)
- Repeat
  - For each training example (x_i, y_i)
    - Compute $f(w^{(k)}, x_i)$
    - Update the weights: $w^{(k+1)} = w^{(k)} + \lambda \left[y_i - f(w^{(k)}, x_i) \right] x_i$
- Until stopping condition is met

Iteration index

Learning rate

Perceptron Learning Rule

- Weight update formula:

$$w^{(k+1)} = w^{(k)} + \lambda\left[y_i - f(w^{(k)}, x_i)\right]x_i \;\; ; \;\; \lambda : \text{learning rate}$$

- Intuition:
  - Update weight based on error: $e = \left[y_i - f(w^{(k)}, x_i)\right]$
  - If y=f(x,w), e=0: no update needed
  - If y>f(x,w), e=2: weight must be increased so that f(x,w) will increase
  - If y<f(x,w), e=-2: weight must be decreased so that f(x,w) will decrease

The Learning Rate

- Is a parameter with value between 0 and 1 used to control the amount of adjustment made in each iteration.
- If is close to 0 the new weight is mostly influenced by the value of the old weight.
- If it is close to 1, then the new weight is mostly influenced by the current adjustment.
- The learning rate can be adaptive: initially moderately large and the gradually decreases in subsequent iterations.

Example of Perceptron Learning

$$w^{(k+1)} = w^{(k)} + \lambda\left[y_i - f(w^{(k)}, x_i)\right]x_i$$

$$Y = sign(\sum_{i=0}^{d} w_i X_i)$$

$$\lambda = 0.1$$

X_1	X_2	X_3	Y
1	0	0	-1
1	0	1	1
1	1	0	1
1	1	1	1
0	0	1	-1
0	1	0	-1
0	1	1	1
0	0	0	-1

	w_0	w_1	w_2	w_3
0	0	0	0	0
1	-0.2	-0.2	0	0
2	0	0	0	0.2
3	0	0	0	0.2
4	0	0	0	0.2
5	-0.2	0	0	0
6	-0.2	0	0	0
7	0	0	0.2	0.2
8	-0.2	0	0.2	0.2

Epoch	w_0	w_1	w_2	w_3
0	0	0	0	0
1	-0.2	0	0.2	0.2
2	-0.2	0	0.4	0.2
3	-0.4	0	0.4	0.2
4	-0.4	0.2	0.4	0.4
5	-0.6	0.2	0.4	0.2
6	-0.6	0.4	0.4	0.2

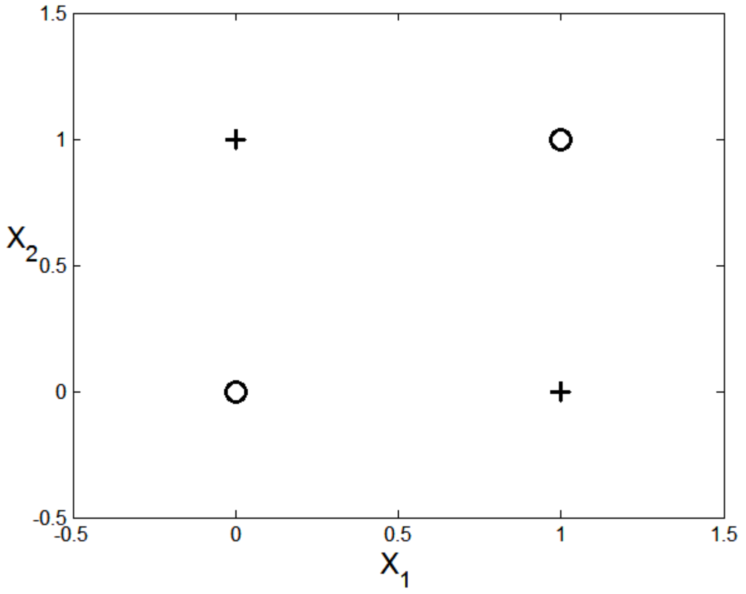
Nonlinearly Separable Data

- Since *f(w,x)* is a linear combination of input variables, decision boundary is linear.
- For nonlinearly separable problems, the perceptron fails because no linear hyperplane can separate the data perfectly.
- An example of nonlinearly separable data is the XOR function.

XOR Data

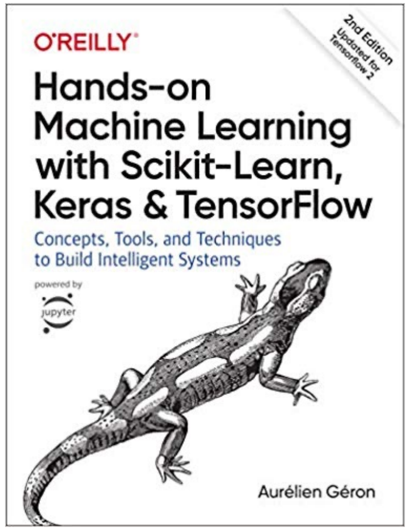
x_1	x_2	y
0	0	-1
1	0	1
0	1	1
1	1	-1

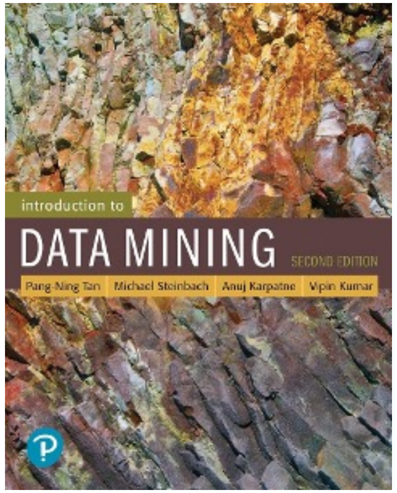
$$y = x_1 \oplus x_2$$

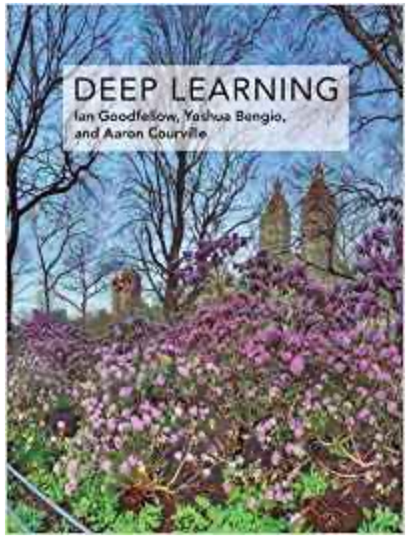


References

- Artificial Neural Network. Chapter 5.4 and 5.5. Introduction to Data Mining.
- Hands-on Machine Learning with Scikit-Learn, Keras & Tensorflow. A practical handbook to start wrestling with Machine Learning models (2nd ed).
- Deep Learning. Ian Goodfellow, Yoshua Bengio, and Aaron Courville. The reference book for deep learning models.



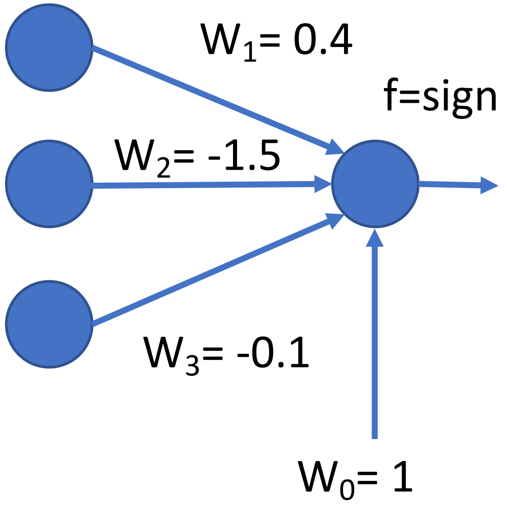




Exercises - Perceptron

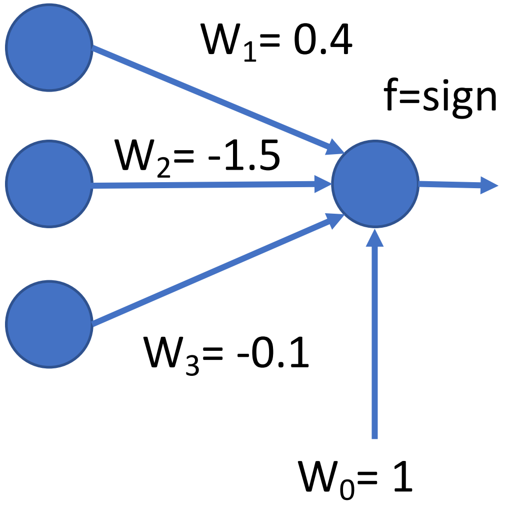
Predict with Perceptron

id	X_1	X_2	X_3	Y
1	0	0	0	
2	1	1	1	
3	1	0	1	
4	0	2	0	



Predict with Perceptron - Solution

id	X_1	X_2	X_3	Y
1	0	0	0	1
2	1	1	1	-1
3	1	0	1	1
4	0	2	0	-1



Y_1 = sign(1 + 0.4 * 0 + -1.5 * 0 + -0.1 * 0) = sign(1) = 1

Y_2 = sign(1 + 0.4 * 1 + -1.5 * 1 + -0.1 * 1) = sign(-0.2) = -1

Y_3 = sign(1 + 0.4 * 1 + -1.5 * 0 + -0.1 * 0) = sign(1.3) = 1

Y_4 = sign(1 + 0.4 * 0 + -1.5 * 2 + -0.1 * 0) = sign(-2) = -1

Train Linear Perceptron

id	X_1	X_2	Y
a	0	1	-1
b	0	0	1
c	1	2	-1

Lambda = 0.3

f = sign

it	W_0	W_1	W_2	X.W	f(X.W)	error	$delta_0$	$delta_1$	$delta_2$
1	-1	0	0						
2									
3									
4									
5									
6									
7									
8									
9									
10									
11									
12									

Train Linear Perceptron

id	X_1	X_2	Y
a	0	1	-1
b	0	0	1
c	1	2	-1

Lambda = 0.3

f = sign

it	W_0	W_1	W_2	X.W	f(X.W)	error	$delta_0$	$delta_1$	$delta_2$
1	-1	0	0	-1					
2									
3									
4									
5									
6									
7									
8									
9									
10									
11									
12									

Train Linear Perceptron

id	X_1	X_2	Y
a	0	1	-1
b	0	0	1
c	1	2	-1

Lambda = 0.3

f = sign

it	W_0	W_1	W_2	X.W	f(X.W)	error	$delta_0$	$delta_1$	$delta_2$
1	-1	0	0	-1	-1				
2									
3									
4									
5									
6									
7									
8									
9									
10									
11									
12									

Train Linear Perceptron

id	X_1	X_2	Y
a	0	1	-1
b	0	0	1
c	1	2	-1

Lambda = 0.3

f = sign

it	W_0	W_1	W_2	X.W	f(X.W)	error	$delta_0$	$delta_1$	$delta_2$
1	-1	0	0	-1	-1	0			
2									
3									
4									
5									
6									
7									
8									
9									
10									
11									
12									

Train Linear Perceptron

id	X_1	X_2	Y
a	0	1	-1
b	0	0	1
c	1	2	-1

Lambda = 0.3

f = sign

it	W_0	W_1	W_2	X.W	f(X.W)	error	$delta_0$	$delta_1$	$delta_2$
1	-1	0	0	-1	-1	0	0		
2									
3									
4									
5									
6									
7									
8									
9									
10									
11									
12									

Train Linear Perceptron

id	X_1	X_2	Y
a	0	1	-1
b	0	0	1
c	1	2	-1

Lambda = 0.3

f = sign

it	W_0	W_1	W_2	X.W	f(X.W)	error	$delta_0$	$delta_1$	$delta_2$
1	-1	0	0	-1	-1	0	0	0	0
2									
3									
4									
5									
6									
7									
8									
9									
10									
11									
12									

Train Linear Perceptron

id	X_1	X_2	Y
a	0	1	-1
b	0	0	1
c	1	2	-1

Lambda = 0.3

f = sign

it	W_0	W_1	W_2	X.W	f(X.W)	error	$delta_0$	$delta_1$	$delta_2$
1	-1	0	0	-1	-1	0	0	0	0
2									
3									
4									
5									
6									
7									
8									
9									
10									
11									
12									

Train Linear Perceptron

id	X_1	X_2	Y
a	0	1	-1
b	0	0	1
c	1	2	-1

Lambda = 0.3

f = sign

it	W_0	W_1	W_2	X.W	f(X.W)	error	$delta_0$	$delta_1$	$delta_2$
1	-1	0	0	-1	-1	0	0	0	0
2	-1								
3									
4									
5									
6									
7									
8									
9									
10									
11									
12									

Train Linear Perceptron

id	X_1	X_2	Y
a	0	1	-1
b	0	0	1
c	1	2	-1

Lambda = 0.3

f = sign

it	W_0	W_1	W_2	X.W	f(X.W)	error	$delta_0$	$delta_1$	$delta_2$
1	-1	0	0	-1	-1	0	0	0	0
2	-1	0							
3									
4									
5									
6									
7									
8									
9									
10									
11									
12									

Train Linear Perceptron

id	X_1	X_2	Y
a	0	1	-1
b	0	0	1
c	1	2	-1

Lambda = 0.3

f = sign

it	W_0	W_1	W_2	X.W	f(X.W)	error	$delta_0$	$delta_1$	$delta_2$
1	-1	0	0	-1	-1	0	0	0	0
2	-1	0	0						
3									
4									
5									
6									
7									
8									
9									
10									
11									
12									

Train Linear Perceptron

id	X_1	X_2	Y
a	0	1	-1
b	0	0	1
c	1	2	-1

Lambda = 0.3

f = sign

it	W_0	W_1	W_2	X.W	f(X.W)	error	$delta_0$	$delta_1$	$delta_2$
1	-1	0	0	-1	-1	0	0	0	0
2	-1	0	0	-1					
3									
4									
5									
6									
7									
8									
9									
10									
11									
12									

Train Linear Perceptron

id	X_1	X_2	Y
a	0	1	-1
b	0	0	1
c	1	2	-1

Lambda = 0.3

f = sign

it	W_0	W_1	W_2	X.W	f(X.W)	error	$delta_0$	$delta_1$	$delta_2$
1	-1	0	0	-1	-1	0	0	0	0
2	-1	0	0	-1	-1				
3									
4									
5									
6									
7									
8									
9									
10									
11									
12									

Train Linear Perceptron

id	X_1	X_2	Y
a	0	1	-1
b	0	0	1
c	1	2	-1

Lambda = 0.3

f = sign

it	W_0	W_1	W_2	X.W	f(X.W)	error	$delta_0$	$delta_1$	$delta_2$
1	-1	0	0	-1	-1	0	0	0	0
2	-1	0	0	-1	-1	2			
3									
4									
5									
6									
7									
8									
9									
10									
11									
12									

Train Linear Perceptron

id	X_1	X_2	Y
a	0	1	-1
b	0	0	1
c	1	2	-1

Lambda = 0.3

f = sign

it	W_0	W_1	W_2	X.W	f(X.W)	error	$delta_0$	$delta_1$	$delta_2$
1	-1	0	0	-1	-1	0	0	0	0
2	-1	0	0	-1	-1	2	0.6		
3									
4									
5									
6									
7									
8									
9									
10									
11									
12									

Train Linear Perceptron

id	X_1	X_2	Y
a	0	1	-1
b	0	0	1
c	1	2	-1

Lambda = 0.3

f = sign

it	W_0	W_1	W_2	X.W	f(X.W)	error	$delta_0$	$delta_1$	$delta_2$
1	-1	0	0	-1	-1	0	0	0	0
2	-1	0	0	-1	-1	2	0.6	0	
3									
4									
5									
6									
7									
8									
9									
10									
11									
12									

Train Linear Perceptron

id	X_1	X_2	Y
a	0	1	-1
b	0	0	1
c	1	2	-1

Lambda = 0.3

f = sign

it	W_0	W_1	W_2	X.W	f(X.W)	error	$delta_0$	$delta_1$	$delta_2$
1	-1	0	0	-1	-1	0	0	0	0
2	-1	0	0	-1	-1	2	0.6	0	0
3									
4									
5									
6									
7									
8									
9									
10									
11									
12									

Train Linear Perceptron

id	X_1	X_2	Y
a	0	1	-1
b	0	0	1
c	1	2	-1

Lambda = 0.3

f = sign

it	W_0	W_1	W_2	X.W	f(X.W)	error	$delta_0$	$delta_1$	$delta_2$
1	-1	0	0	-1	-1	0	0	0	0
2	-1	0	0	-1	-1	2	0.6	0	0
3	-0.4								
4									
5									
6									
7									
8									
9									
10									
11									
12									

Train Linear Perceptron

id	X_1	X_2	Y
a	0	1	-1
b	0	0	1
c	1	2	-1

Lambda = 0.3

f = sign

it	W_0	W_1	W_2	X.W	f(X.W)	error	$delta_0$	$delta_1$	$delta_2$
1	-1	0	0	-1	-1	0	0	0	0
2	-1	0	0	-1	-1	2	0.6	0	0
3	-0.4	0	0						
4									
5									
6									
7									
8									
9									
10									
11									
12									

Train Linear Perceptron - Solution

id	X_1	X_2	Y
a	0	1	-1
b	0	0	1
c	1	2	-1

Lambda = 0.3

f = sign

it	W_0	W_1	W_2	X.W	f(X.W)	error	$delta_0$	$delta_1$	$delta_2$
1	-1	0	0	-1	-1	0	0	0	0
2	-1	0	0	-1	-1	2	0,6	0	0
3	-0,4	0	0	-0,4	-1	0	0	0	0
4	-0,4	0	0	-0,4	-1	0	0	0	0
5	-0,4	0	0	-0,4	-1	2	0,6	0	0
6	0,2	0	0	0,2	1	-2	-0,6	-0,6	-1,2
7	-0,4	-0,6	-1,2	-1,6	-1	0	0	0	0
8	-0,4	-0,6	-1,2	-0,4	-1	2	0,6	0	0
9	0,2	-0,6	-1,2	-2,8	-1	0	0	0	0
10	0,2	-0,6	-1,2	-1	-1	0	0	0	0
11	0,2	-0,6	-1,2	0,2	1	0	0	0	0
12	0,2	-0,6	-1,2	-2,8	-1	0	0	0	0