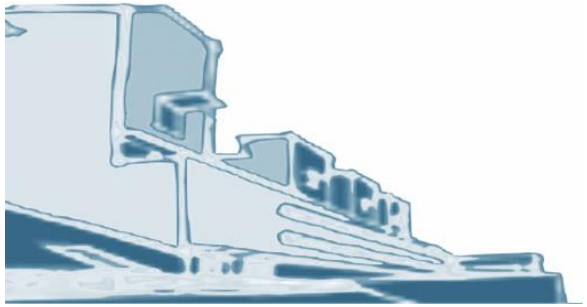
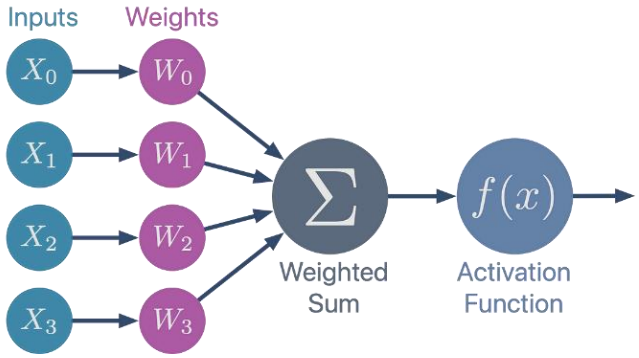
AC – Aprendizagem Computacional / Machine Learning

P4a – Perceptron

Jorge Henriques
jh@dei.uc.pt

Departamento de Engenharia Informática
Faculdade de Ciências e Tecnologia





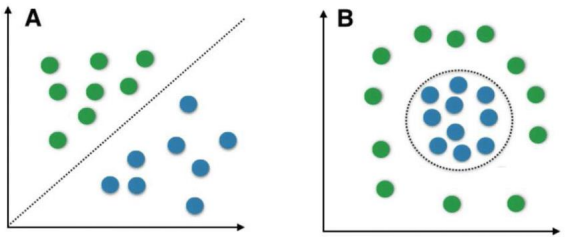
1 2 9 0
UNIVERSIDADE D COIMBRA

dei engenharia informática

Contents

Objectives – Network with one Layers of neurons

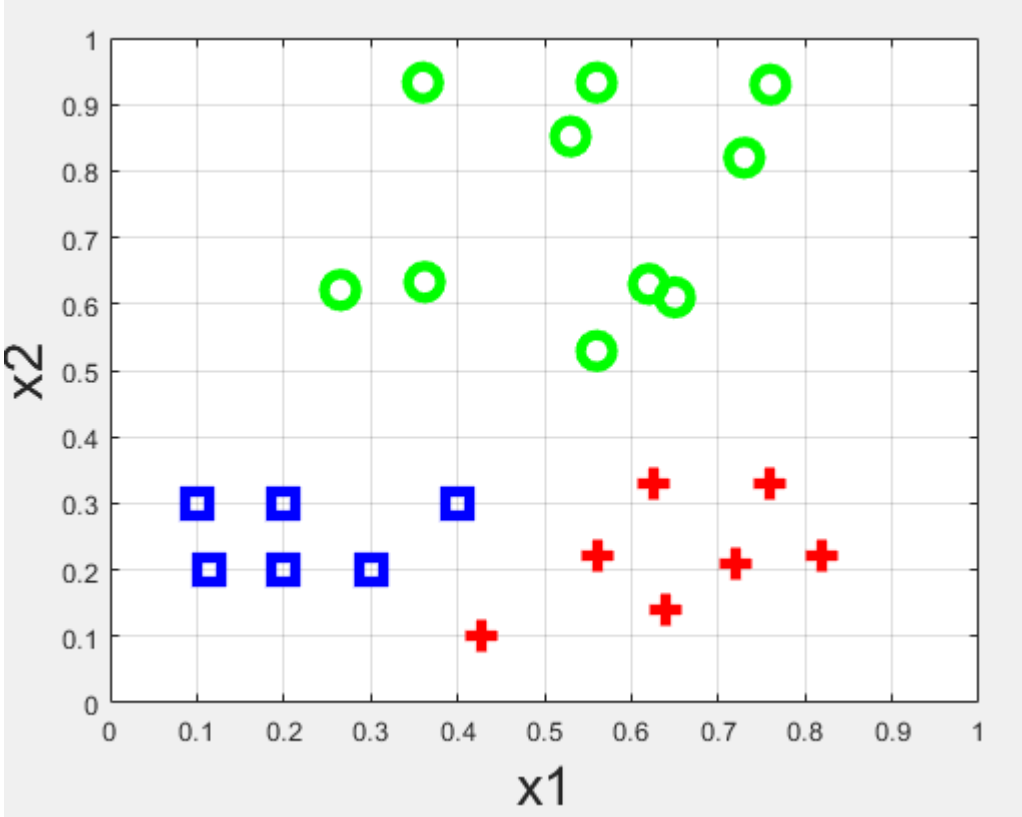
- Concepts
- Datasets/problems
  - Linearly separable
  - Not linearly separable



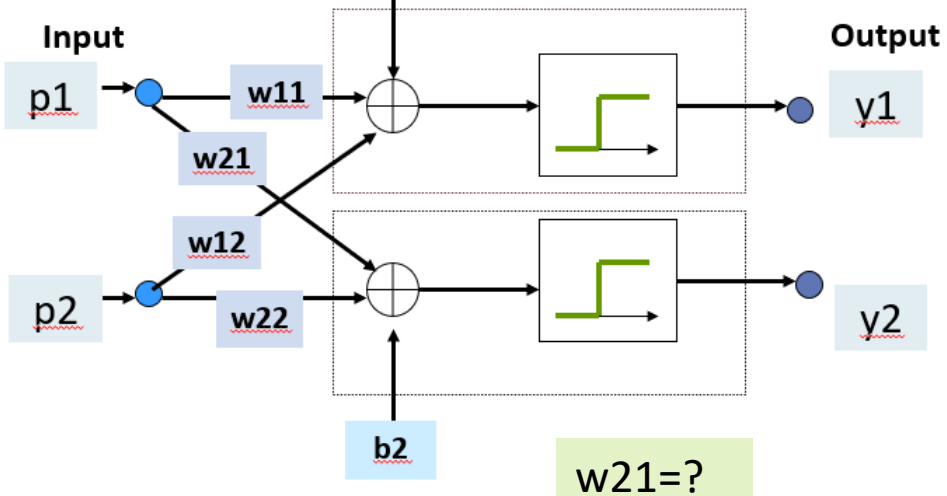
- Structures
  - Perceptron | Implement perceptron training rule
    By hand and automatically
  - Adaline | use of pseudo-inverse
  - Sigmoidal activation | Implement Recursive Least squares

Contents

- Train a perceptron network (one layer)
  - Define the structure (number of perceptrons)
  - Train the network (parameters w, b)



- Training a perceptron network



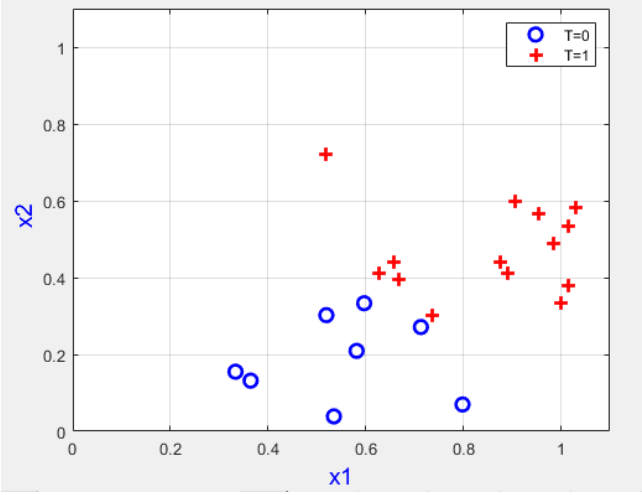
w11=?
w12=?
b1=?

w21=?
w22=?
b2=?

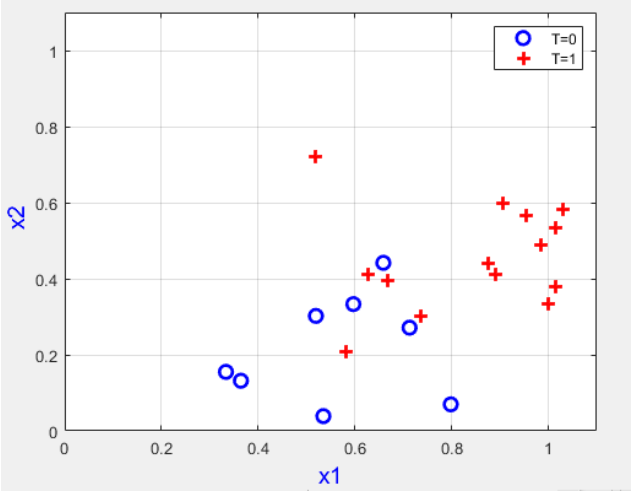
If y1=1 AND y2=1 Y=?
If y1=1 AND y2=0 Y=?
....

Contents

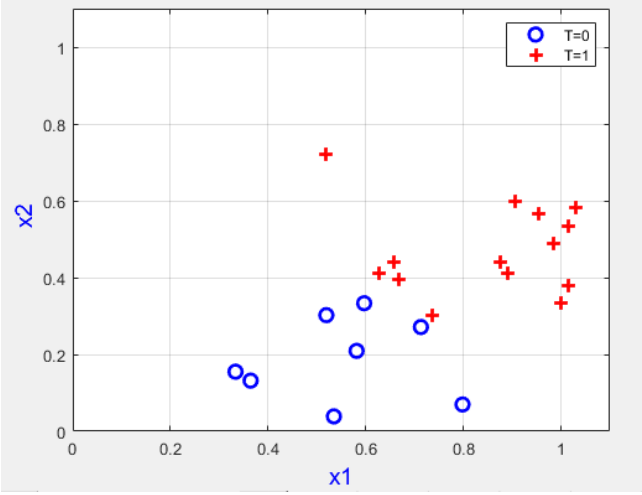
- **Datasets**
  - Dataset 1 – linearly separable
  - P4_data1.csv

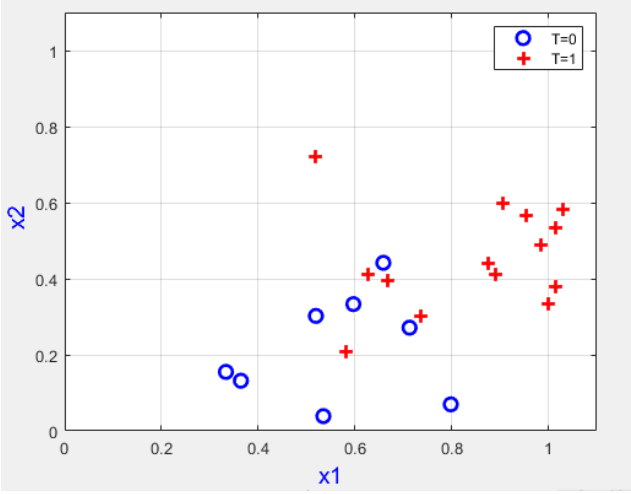


  - Dataset 2 – non-linearly separable
  - P4_data2.csv



- **Three algorithms**
  - 1| Perceptron
    - Perceptron Training rule (on-line)
  - 2| Adaline
    - Pseudo-inverse (off-line)
  - 3| Sigmoidal
    - RLMSE (on-line)





- **DATA**
  - X={x1,x2}

0.8147	0.1270	0.6324	0.2785	0.9575	0.1576	0.9572	0.8003	0.4218	0.7922
0.9058	0.9134	0.0975	0.5469	0.9649	0.9706	0.4854	0.1419	0.9157	0.9595

(R=2,N=10)

  - $x_0 = \begin{matrix} 0.8147 \\ 0.9058 \end{matrix}$ xi = ...

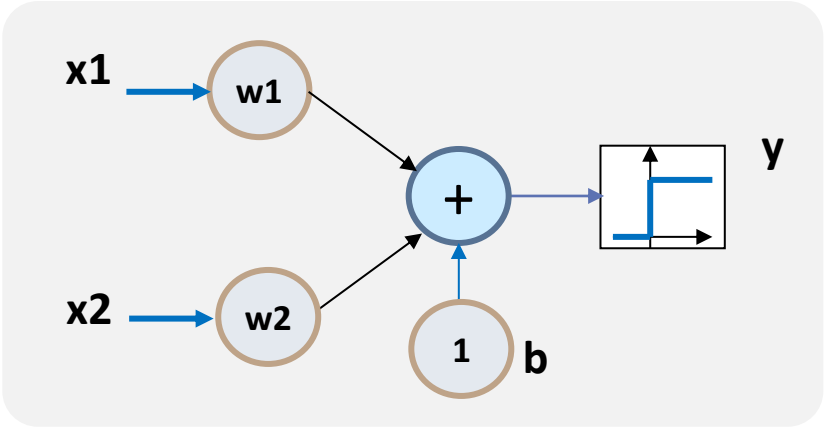
  - T

1 0 1 0 1 0 1 1 0 1

(1,N=10)

1| Perceptron learning rule (iterative)

- Activation function: hardlim



$$\theta = \underset{(1,3)}{\begin{bmatrix} w & b \end{bmatrix}} \qquad z = \underset{(3,1)}{\begin{bmatrix} x_i \\ 1 \end{bmatrix}}$$

$$y = \text{hardlim}(\theta z)$$

$$W^{(new)} = W^{(old)} + E\ X^T$$

(1,2) (1,N) (N,2)

$$b^{(new)} = b^{(old)} + E\ \mathrm{I}$$

(1,1) (1,N)(N,1)

EPOCH

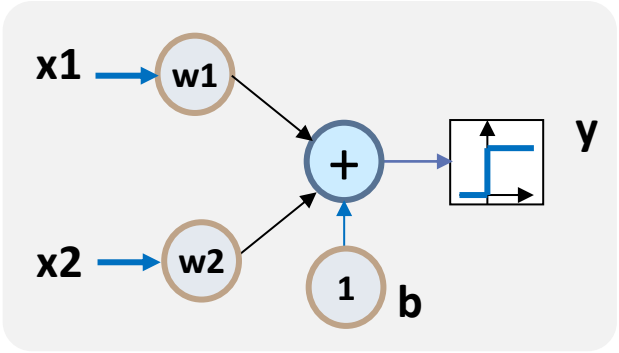
Z (3,N)

$E = T - Y$ (1,N)

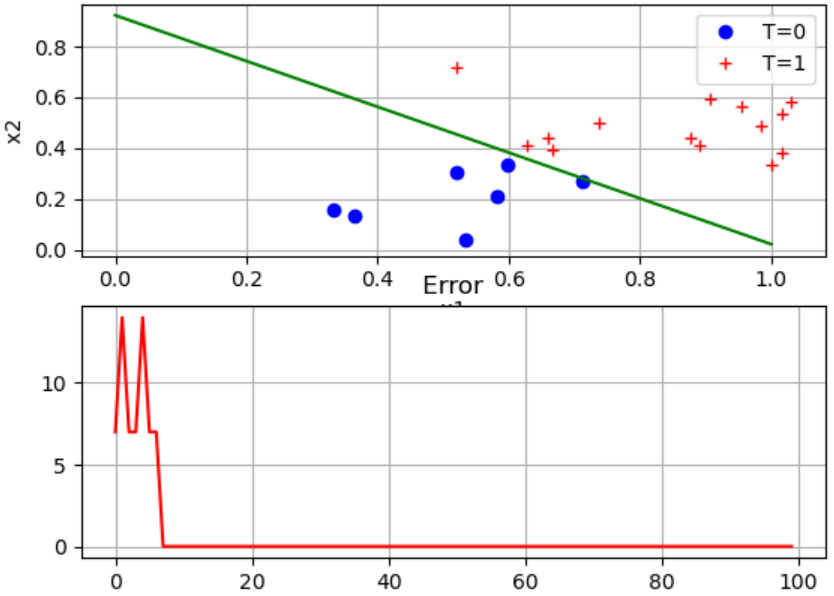
$$\theta^{(new)} = \theta^{(old)} + E\ Z^T$$

(1,3) (1,N) (N,3)

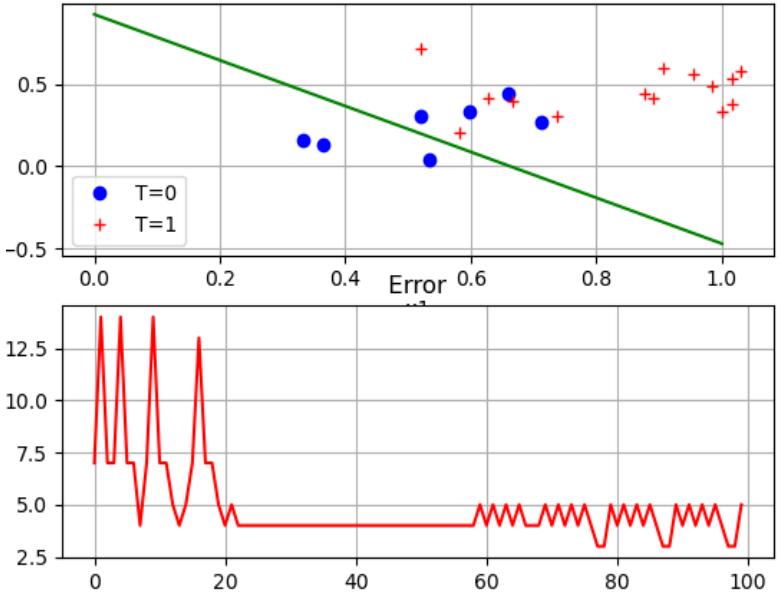
- **1| Perceptron - iterative learning rule**
  - Linearly / nonlinearly separable ?



Dataset 1

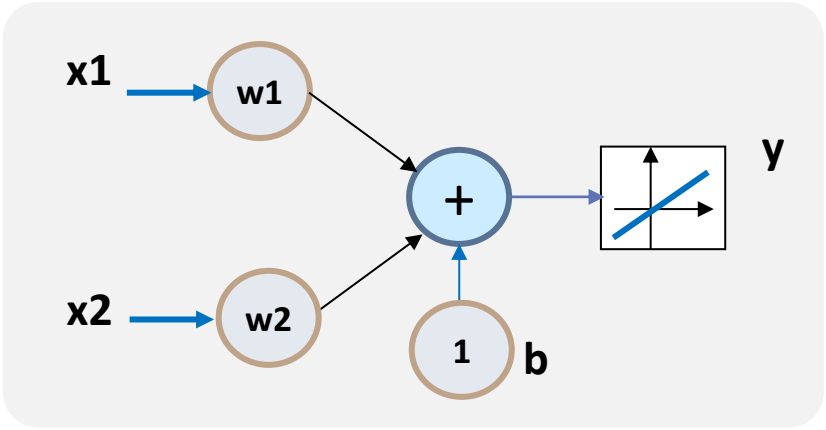


Dataset 2



2| Adaline – offline pseudoinverse

- Activation function: **purelin**



$$\theta = \begin{bmatrix} w & b \end{bmatrix} \quad (1,3) \qquad z = \begin{bmatrix} x_i \\ 1 \end{bmatrix} \quad (3,1)$$

$$y = \theta z$$

Z (3,N)

$$e = T - Y \quad (1,N)$$

$$\theta = T \left[Z^T Z \right]^{-1} Z^T$$

(1,3) (1,N) (N,3)(3,N) (N,3)

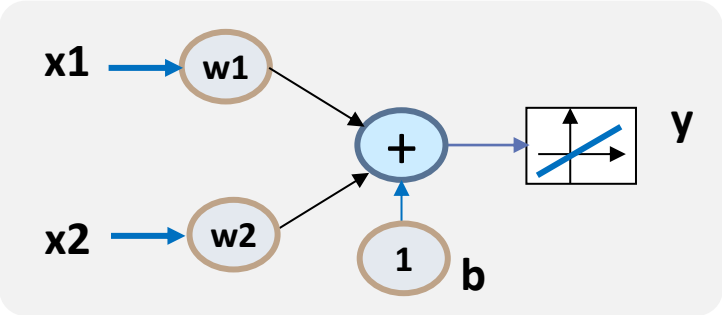
$$\theta = T\ Z^{+}$$

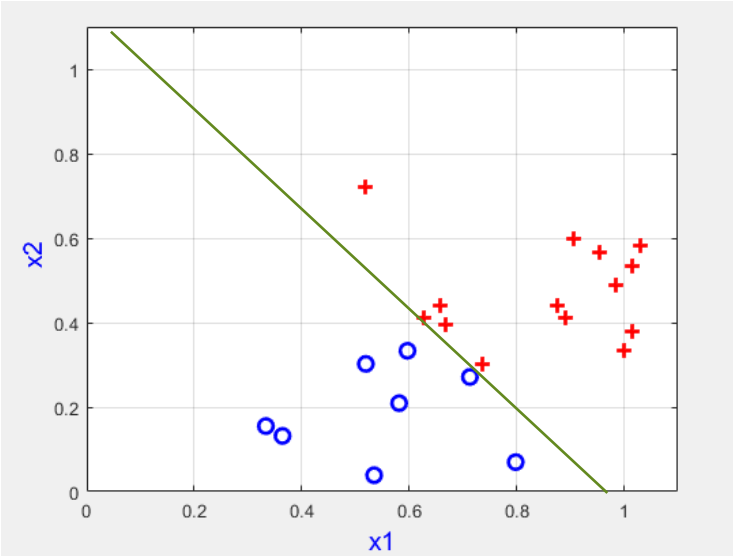
(1,N) (N,3)

```
ONES = np.ones([1,N])                          #(1,22)
Z    = np.concatenate( (X,ONES), axis=0)       #(3,22)
Pinv = np.linalg.pinv(Z)                       #(22,3)
PAR  = np.dot(T, Pinv)                         #(1,22)(22,3)
```

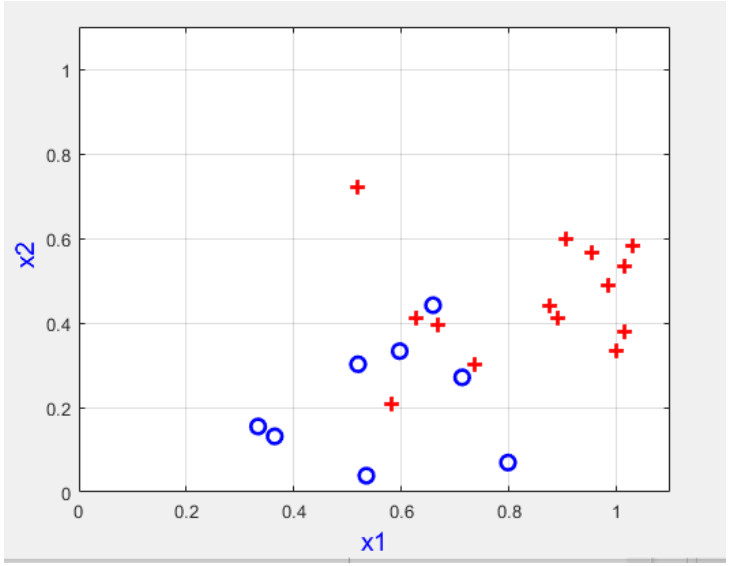
2| Adaline - pseudoinverse

- Activation function: **purelin**



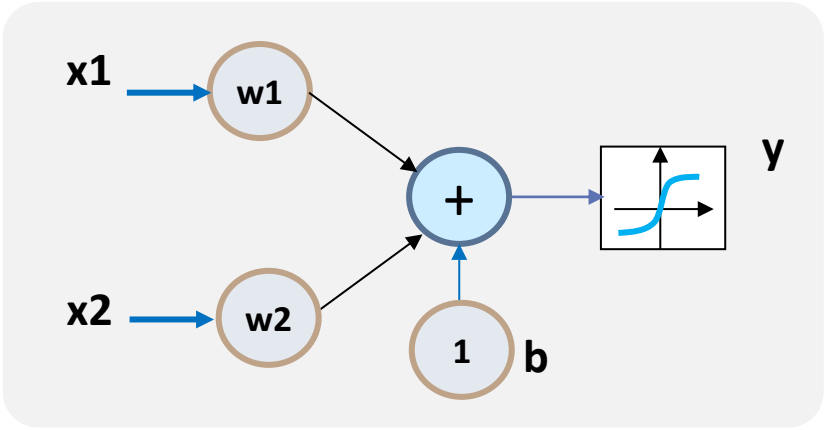


???



3| Nonlinear – recursive least squares

- Activation function: **sigmoidal**



$$\theta = \begin{bmatrix} w & b \end{bmatrix} \quad z = \begin{bmatrix} X \\ 1 \end{bmatrix}$$

(1,3) (3,1)

$$y = \log sig(\theta z)$$

$$W^{(new)} = W^{(old)} + \alpha \left(E \otimes F' \right) X^T$$

(1,2) (1,N) ⊗ (1,N) (N,2)

$$b^{(new)} = b^{(old)} + \left(E \otimes F' \right) \mathrm{I}$$

(1,1) (1,N) ⊗ (1,N) (N,1)

Z (3,N)

α *learning rate*

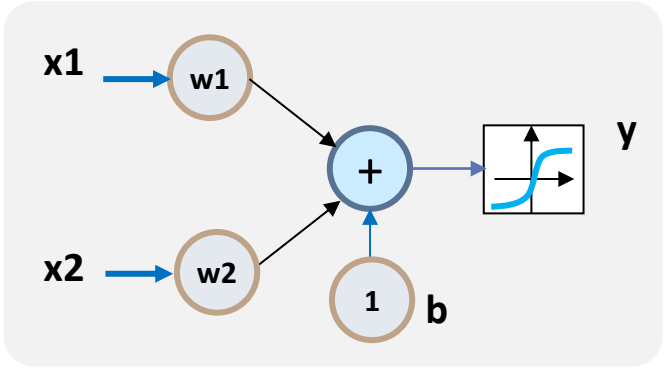
$$e = T - Y$$

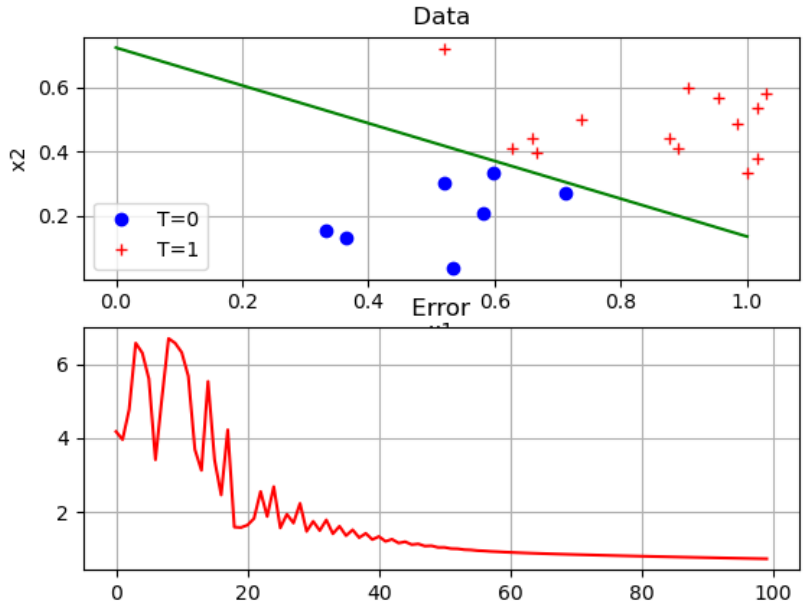
(1,N)

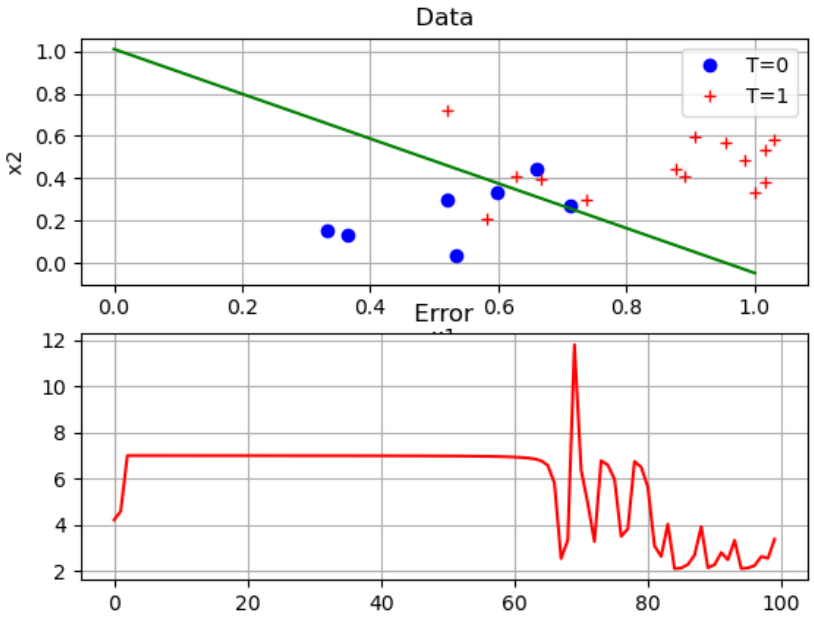
$$\theta^{(new)} = \theta^{(old)} + \alpha \left(E \otimes F' \right) Z^T$$

(1,3) (1,N) ⊗ (1,N) (N,3)

- **3| Nonlinear – RLS**
  - Learning rate, α ?
  - Epochs ?







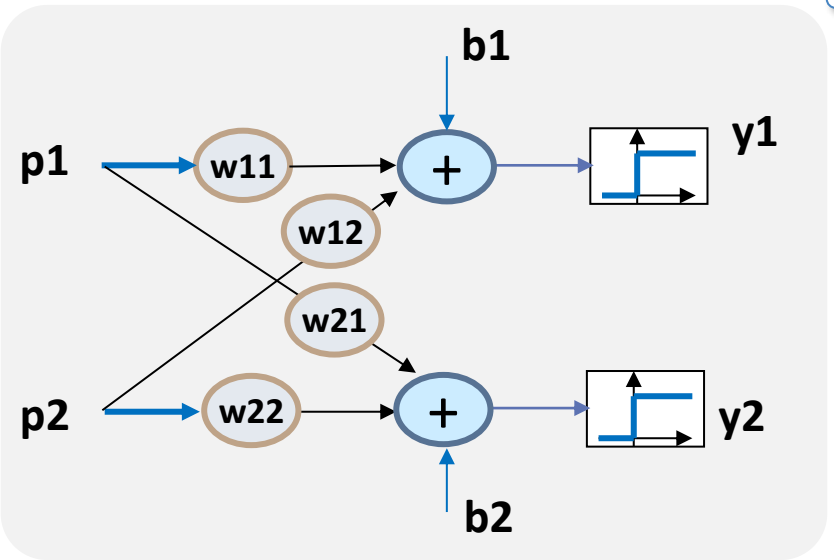
Contents

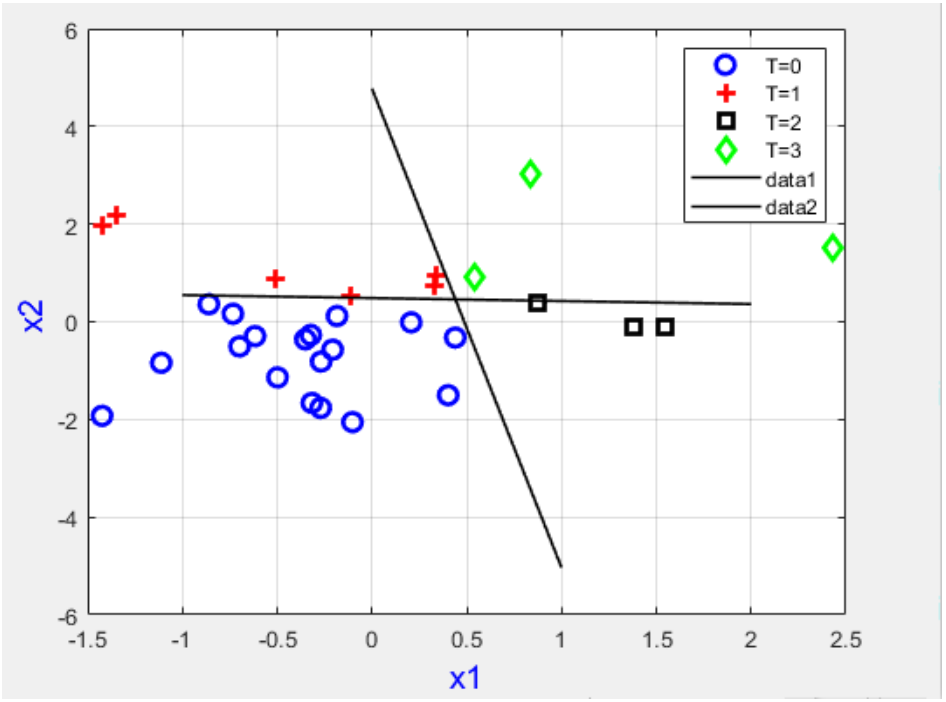
4| Conclusions

- Structure
  - 1| Implement perceptron + perceptron learning rule
    - Epochs ?
  - 2| Implement Adaline + pseudo-inverse
    - Right, Left
  - 3| Implement nonlinear neuron + recursive least squares
    - Epochs ?
    - α - Learning rate value ?

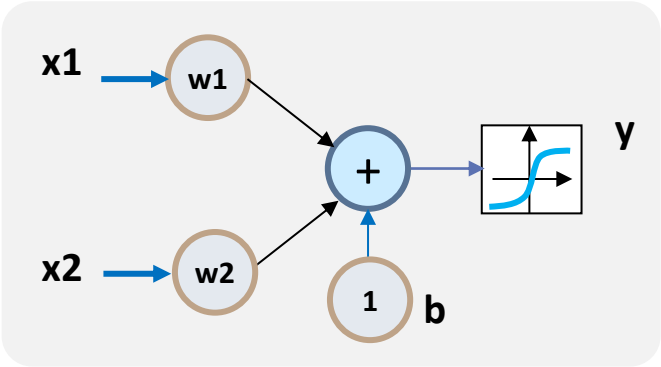
- Dataset
  - Linearly separable
  - Not linearly separable

- **Other ideas - 1| Multiclass**
  - 4 classes
  - Layer of two perceptrons
  - P4_dataset3.csv
  - [x1 x2 y1 y2]





- **Other ideas – 2| nonlinear function**



- Non-linear function

```
t1= 0:0.1:10
x1= sin(t1);
x2= sin(t).cos(t)^22;
T = sin(x1 + 2.4*x2);
```