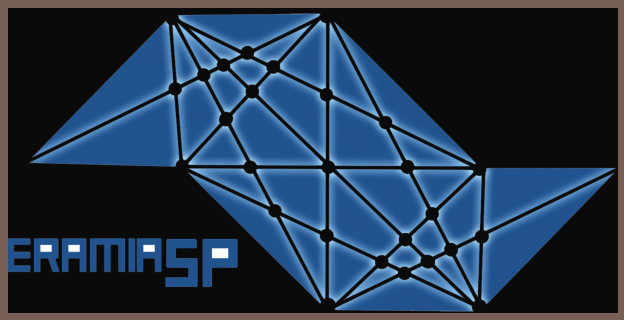


O PERCEPTRON

PROF. RICARDO CERRI

ERAMIA 2020

Universidade Federal de São Carlos
Departamento de Computação – DC/UFSCar

1

Perceptron

- A primeira rede neural descrita algoritmicamente
- Criado por Frank Rosenblatt, um psicólogo, e inspirou engenheiros, físicos e matemáticos a estudarem redes neurais
- O modelo proposto por Rosenblatt em 1958 como publicado em seu artigo, é válido até hoje

2

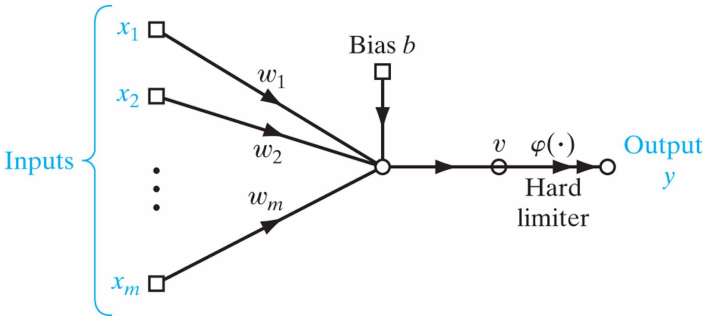
Perceptron

- A forma mais simples de um rede neural utilizada para classificar padrões ditos **linearmente separáveis**
- Consiste de um único neurônio com pesos sinápticos ajustáveis e bias
- Rosenblatt desenvolveu o algoritmo para ajustar os parâmetros livres
- Rosenblatt provou que se os exemplos utilizados no treino pertencerem a classes linearmente separáveis, o algoritmo converge, posicionando um hiperplano entre as duas classes

3

Perceptron

- O Perceptron de Rosenblatt utiliza o modelo de neurônio de McCulloch-Pitts



4

Perceptron

- O Perceptron de Rosenblatt utiliza o modelo de neurônio de McCulloch-Pitts
  - Os pesos sinápticos são denotados por $w_1, w_2, \ldots, w_m$
  - As entradas são denotadas por $x_1, x_2, \ldots, x_m$
  - O bias é denotado por b

$$v = \sum_{i=1}^{m} w_i x_i + b$$

5

Perceptron

- O objetivo do Perceptron é classificar corretamente um conjunto de exemplos denotados por $x_1, x_2, \ldots, x_m$ em uma de umas classes $\mathscr{C}_1$ ou $\mathscr{C}_2$
- O ponto representado por $x_1, x_2, \ldots, x_m$ é classificado como $\mathscr{C}_1$ se a saída y for +1 e como $\mathscr{C}_2$ se a saída y for -1. Há duas regiões separadas por um **hiperplano**:

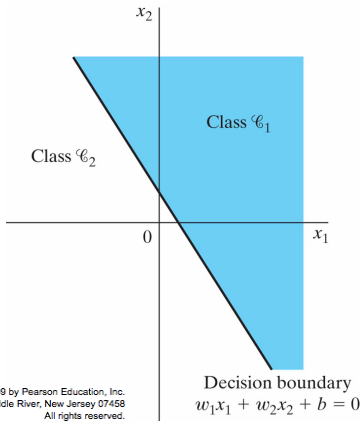
$$\sum_{i=1}^{m} w_i x_i + b = 0$$

6

Perceptron

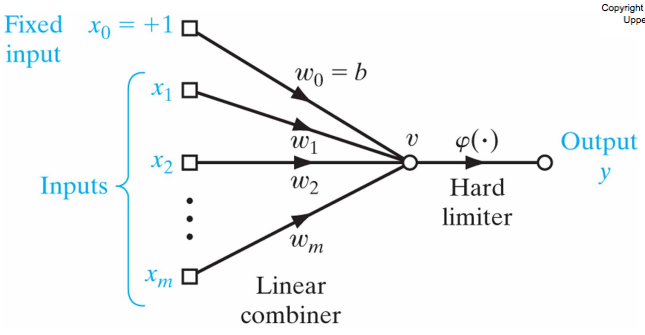
- Ilustração de um hiperplano (linha reta) como fronteira de decisão para um problema de classificação com duas dimensões e duas classes

- Os pesos sinápticos são ajustados em um processo iterativo utilizando o algoritmo de convergência do Perceptron



7

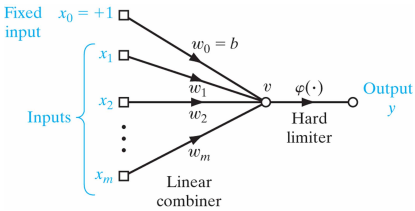
Teorema de Convergência do Perceptron



- O bias $b(n)$ é tratado como um peso associado a uma entrada +1
- Vetor de entrada: $\mathbf{x}(n) = [+1, x_1(n), x_2(n), \ldots, x_m(n)]^T$
- Vetor de pesos: $\mathbf{w}(n) = [b, w_1(n), w_2(n), \ldots, w_m(n)]^T$

8

Teorema de Convergência do Perceptron

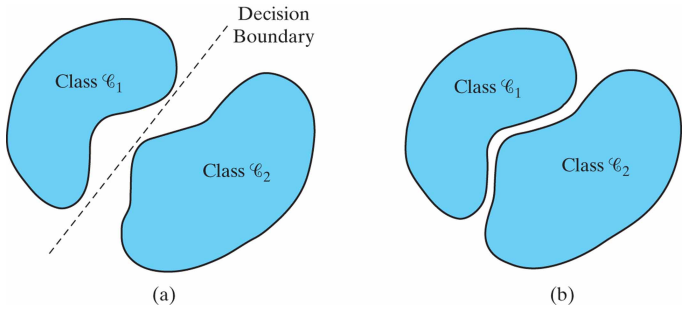


- O bias $b(n)$ é tratado como um peso associado a uma entrada +1
- Vetor de entrada: $\mathbf{x}(n) = [+1, x_1(n), x_2(n), \ldots, x_m(n)]^T$
- Vetor de pesos: $\mathbf{w}(n) = [b, w_1(n), w_2(n), \ldots, w_m(n)]^T$

$$v(n) = \sum_{i=0}^{m} w_i(n)x_i(n) = \mathbf{w}^T(n)\mathbf{x}(n)$$

9

Teorema de Convergência do Perceptron



- $\mathbf{w}^T\mathbf{x} = 0$ define um hiperplano de separação
- $\mathbf{w}^T\mathbf{x} > 0$ para todo vetor $\mathbf{x}$ pertencente à classe $\mathscr{C}_1$
- $\mathbf{w}^T\mathbf{x} \leq 0$ para todo vetor $\mathbf{x}$ pertencente à classe $\mathscr{C}_2$

10

Teorema de Convergência do Perceptron

- Se o *n*-ésimo vetor $\mathbf{x}(n)$ é corretamente classificado pelo vetor $\mathbf{w}(n)$ na *n*-ésima iteração do algoritmo, nenhuma correção é feita no vetor de pesos
  - $\mathbf{w}(n+1) = \mathbf{w}(n)$ se $\mathbf{w}^T\mathbf{x}(n) > 0$ e $\mathbf{x}(n)$ pertence a classe $\mathscr{C}_1$
  - $\mathbf{w}(n+1) = \mathbf{w}(n)$ se $\mathbf{w}^T\mathbf{x}(n) \leq 0$ e $\mathbf{x}(n)$ pertence a classe $\mathscr{C}_2$
- Caso contrário, o vetor de pesos é atualizado
  - $\mathbf{w}(n+1) = \mathbf{w}(n) - \eta(n)\mathbf{x}(n)$ se $\mathbf{w}^T(n)\mathbf{x}(n) > 0$ e $\mathbf{x}(n)$ pertence a classe $\mathscr{C}_2$
  - $\mathbf{w}(n+1) = \mathbf{w}(n) + \eta(n)\mathbf{x}(n)$ se $\mathbf{w}^T(n)\mathbf{x}(n) \leq 0$ e $\mathbf{x}(n)$ pertence a classe $\mathscr{C}_1$
- $\eta(n)$ é a taxa de aprendizado que controla o ajuste dos pesos

11

Teorema de Convergência do Perceptron

- A saída do neurônio é computada utilizando a função sinal $sgn(\cdot)$

$$sgn(v) = \begin{cases} +1 \text{ se } v > 0 \\ -1 \text{ se } v < 0 \end{cases}$$

- Expressamos a saída $y(n)$ de maneira compacta:

$$y(n) = sgn[\mathbf{w}^T(n)\mathbf{x}(n)]$$

12

Teorema de Convergência do Perceptron

- No algoritmo de convergência, foi utilizada também a resposta desejada $d(n)$ para cada exemplo:

$$d(n) = \begin{cases} +1 \text{ se } \mathbf{x}(n) \text{ pertence à classe } \mathscr{C}_1 \\ -1 \text{ se } \mathbf{x}(n) \text{ pertence à classe } \mathscr{C}_2 \end{cases}$$

- A adaptação dos pesos ocorre de maneira elegante:

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \eta[d(n) - y(n)]\mathbf{x}(n)$$

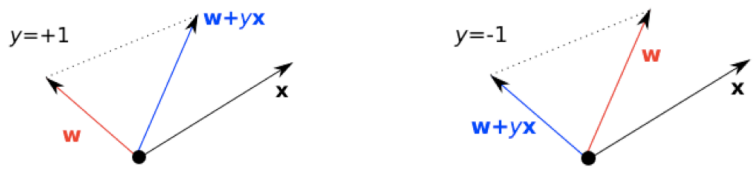
  - η : taxa de aprendizado
  - $d(n) - y(n)$: sinal de erro

13

Teorema de Convergência do Perceptron

- Os pesos são corrigidos de acordo com o valor do produto interno $\mathbf{w}^T(n)\mathbf{x}(n)$

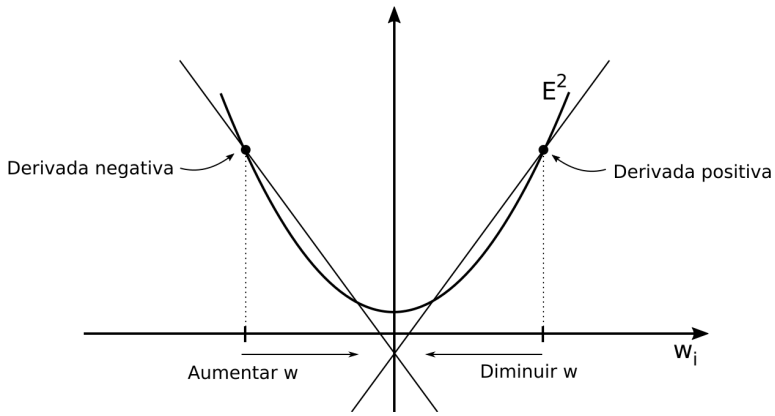
- Se o produto interno, na iteração *n*, tiver um sinal errado, os pesos devem ser ajustados para classificar o exemplo corretamente na iteração *n+1*



14

Como chegamos nisso?

$$E^2 = \left(d(n) - y(n)\right)^2 = \left(d(n) - \mathbf{w}^T(n)\mathbf{x}(n)\right)^2$$

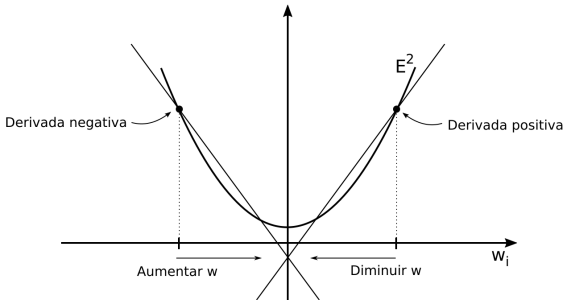


15

Como chegamos nisso?

- Gradiente Descendente: $w_i(n+1) = w_i(n) - \eta \frac{dE^2}{dw_i}$

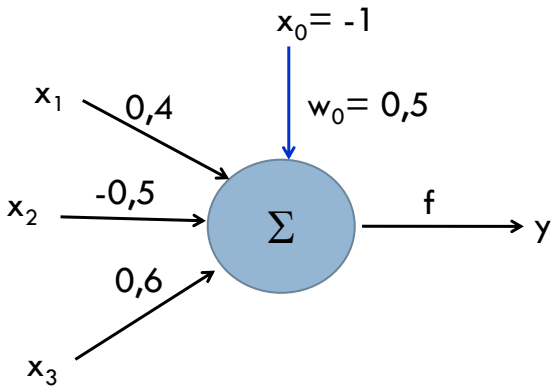
$$\frac{dE^2}{dw_i} = \frac{d\left(d(n) - y(n)\right)^2}{dw_i(n)} = 2 \times \left(d(n) - \mathbf{w}^T(n)\mathbf{x}(n)\right) \times -x_i$$



16

Exemplo

Dado	X1	X2	x3	Classe
E1	0	0	1	-1
E2	1	0	0	1



17

Exemplo

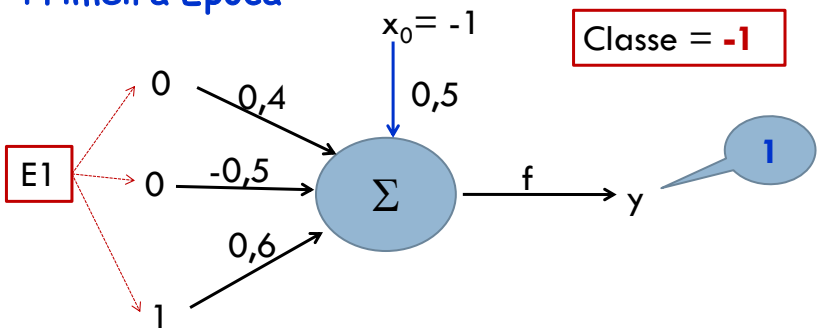
Dado	x1	x2	x3	Classe
E1	0	0	1	-1
E2	1	0	0	1

Aprendizado supervisionado

$$f(X) = \begin{cases} 1 & se \quad X \geq 0 \\ -1 & se \quad X < 0 \end{cases}$$

18

Primeira Época

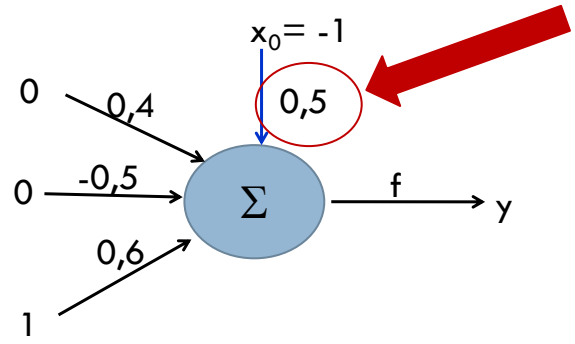


X = (-1 * 0,5) +(0 * 0,4) + (0 * -0,5) + (1 * 0,6) = 0,1

f(0,1) = 1

e = -1 – 1 = -2

19

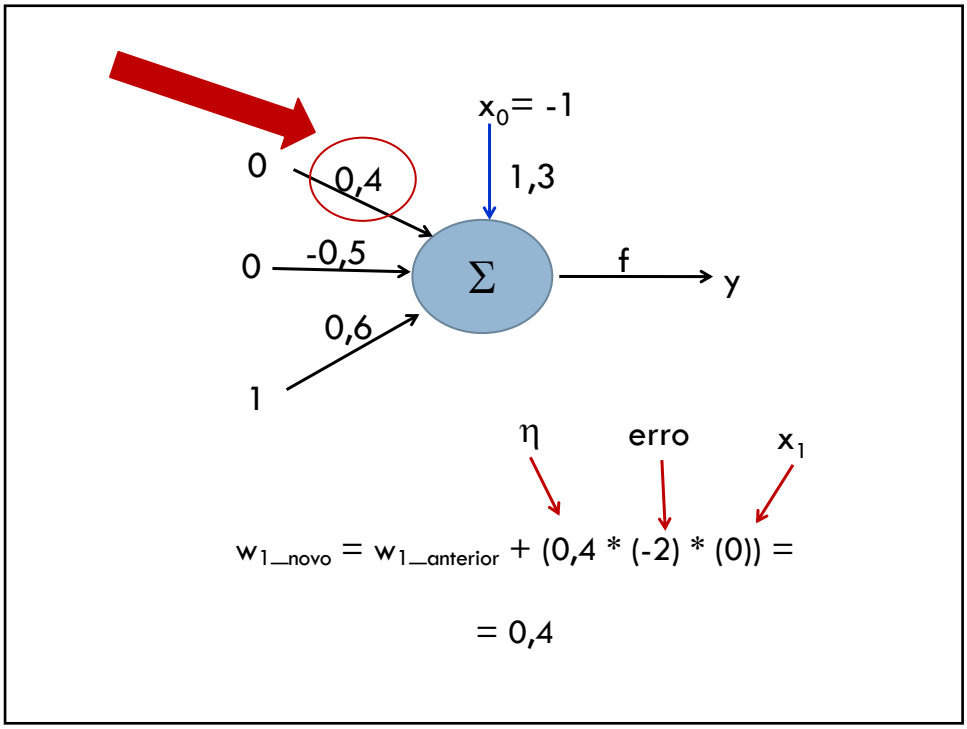


η erro x_0

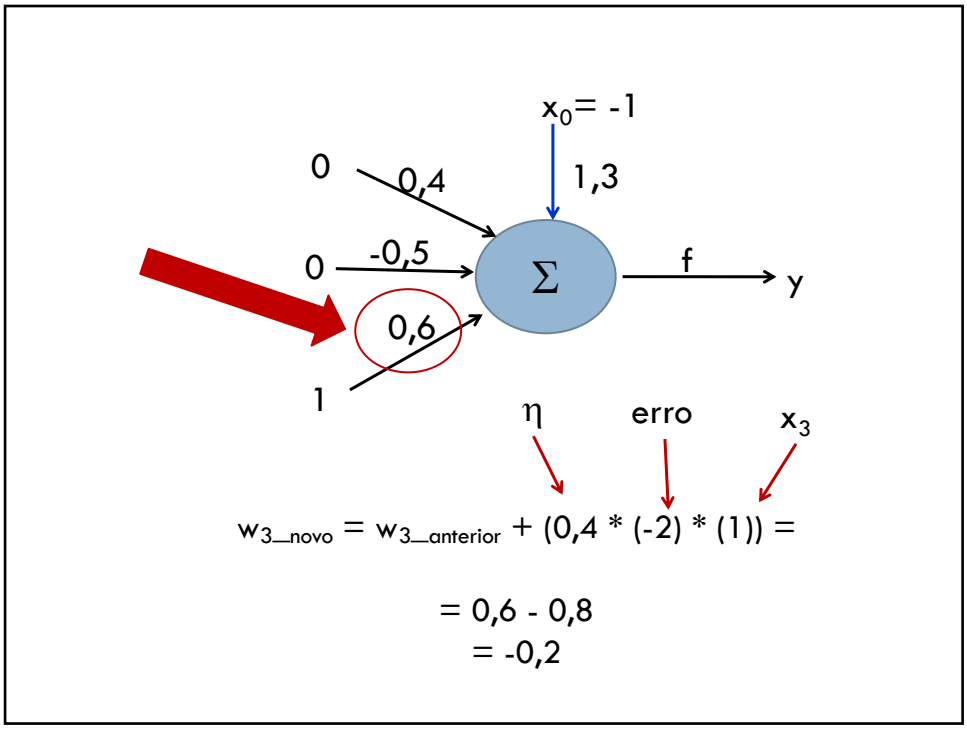
$w_{0\_novo} = w_{0\_anterior} + (0,4 * (-2) * (-1)) =$

$= 0,5 + 0,8 = 1,3$

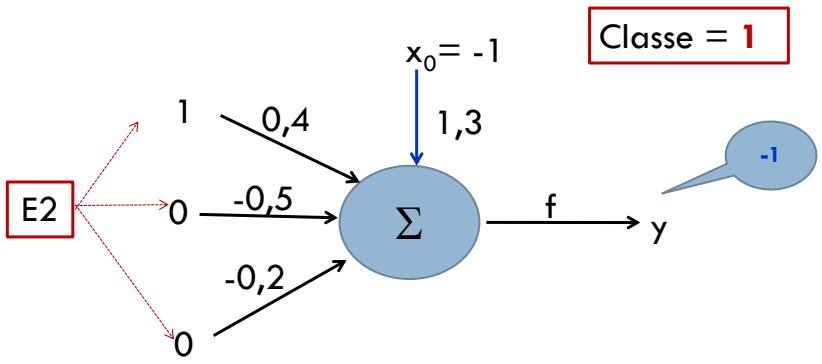
20


x_0= -1
0
0,4
1,3
0
-0,5
Σ
f
y
0,6
1
η
erro
x_1
$w_{1\_novo} = w_{1\_anterior} + (0,4 * (-2) * (0)) =$
= 0,4

21


x_0= -1
0
0,4
1,3
0
-0,5
Σ
f
y
0,6
1
η
erro
x_3
$w_{3\_novo} = w_{3\_anterior} + (0,4 * (-2) * (1)) =$
= 0,6 - 0,8
= -0,2

22

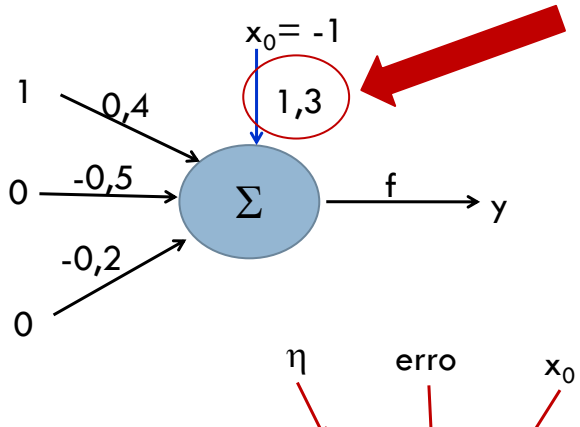


X = (-1 * 1,3) +(1 * 0,4) + (0 * -0,5) + (0 * -0,2) = -0,9

f(-0,9) = **-1**

e = **1** – **(-1)** = 2

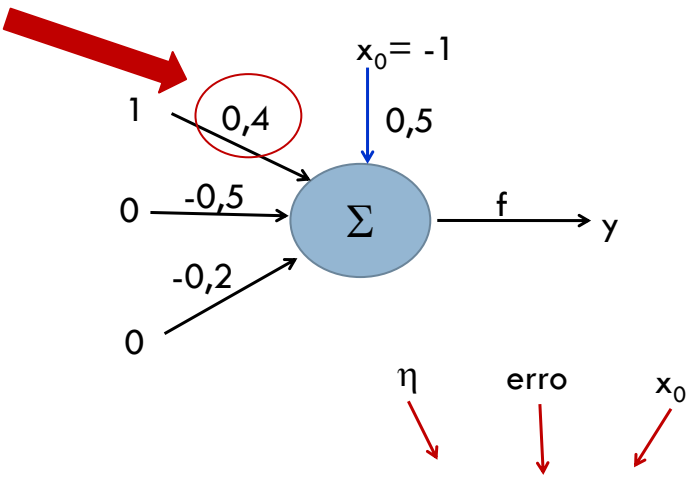
23



$w_{0\_novo} = w_{0\_anterior} + (0,4 * (2) * (-1)) =$

$= 1,3 - 0,8 = 0,5$

24

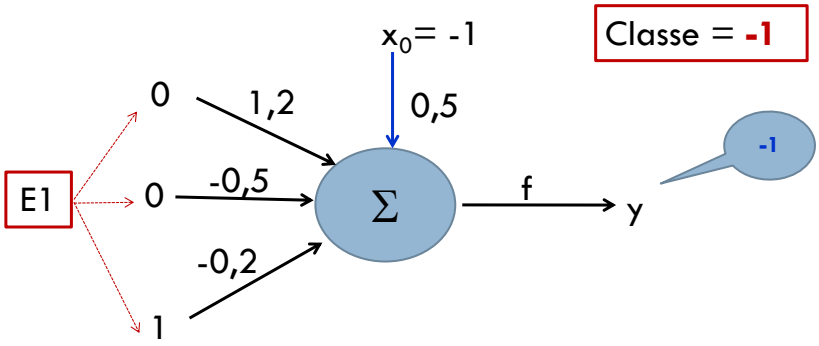


$w_{1\_novo} = w_{1\_anterior} + (0,4 * (2) * (1)) =$

$= 0,4 + 0,8 = 1,2$

25

Segunda Época



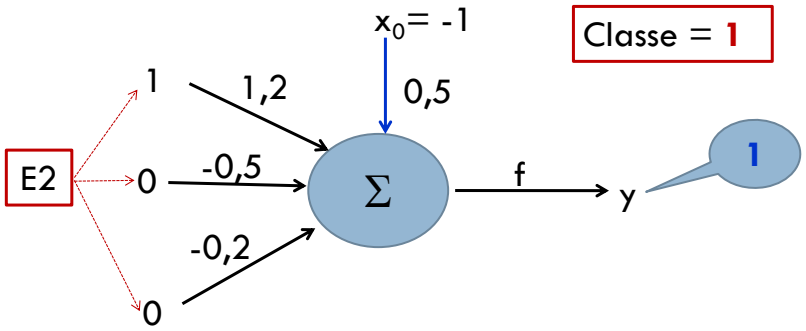
X = (-1 * 0,5) +(0 * 1,2) + (0 * -0,5) + (1 * -0,2) = -0,7

f(-0,7) = **-1**

e = **-1** – **(-1)** = 0

26

Segunda Época



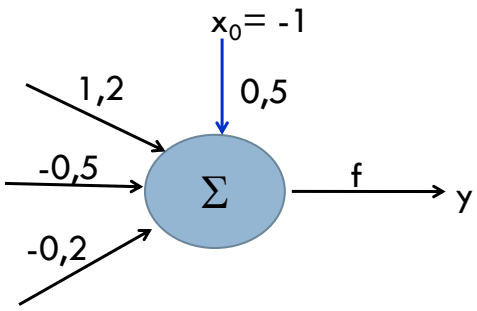
X = (-1 * 0,5) +(1 * 1,2) + (0 * -0,5) + (0 * -0,2) = 0,7

f(0,7) = 1

e = 1 – 1 = 0

27

Rede final

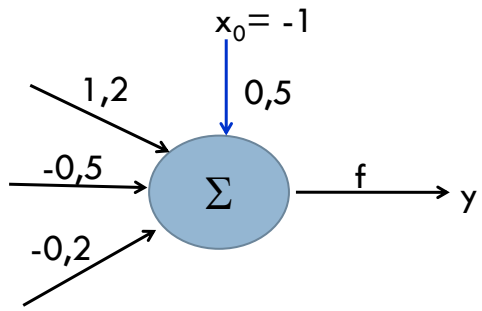


Não ocorreram erros durante a última época.

28

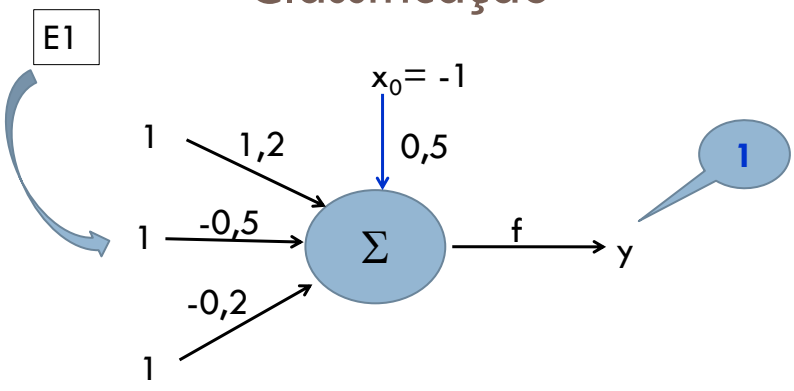
Classificação

Dado	X1	X2	x3	Classe
E1	1	1	1	?
E2	1	1	0	?
E3	0	1	1	?



29

Classificação

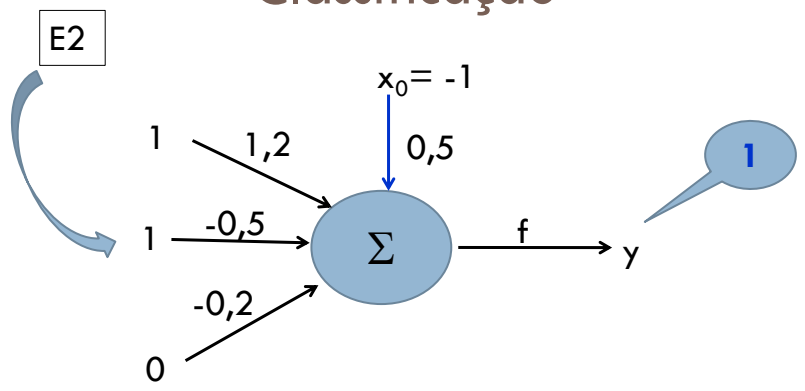


X = (-1 * 0,5) +(1 * 1,2) + (1 * -0,5) + (1 * -0,2) = 0

f(0) = **1**

30

Classificação

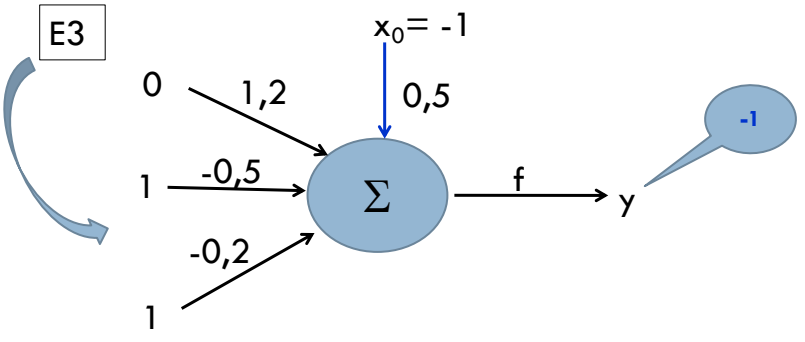


X = (-1 * 0,5) +(1 * 1,2) + (1 * -0,5) + (0 * -0,2) = 0,2

f(0,2) = **1**

31

Classificação



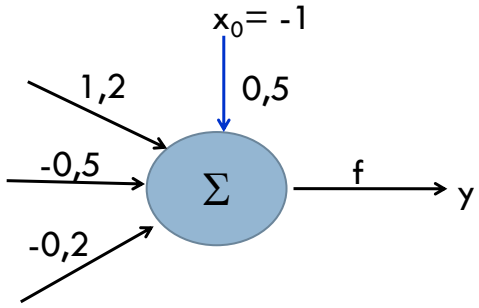
X = (-1 * 0,5) +(0 * 1,2) + (1 * -0,5) + (1 * -0,2) = -1,2

f(-1,2) = **-1**

32

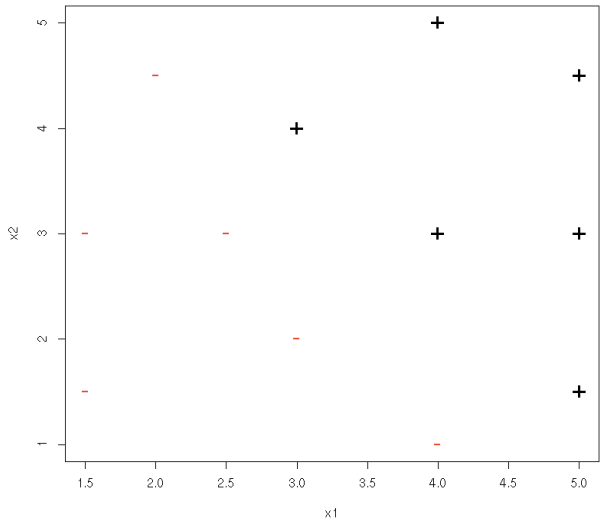
Classificação

Dado	X1	X2	x3	Classe
E1	1	1	1	**1**
E2	1	1	0	**1**
E3	0	1	1	**-1**



33

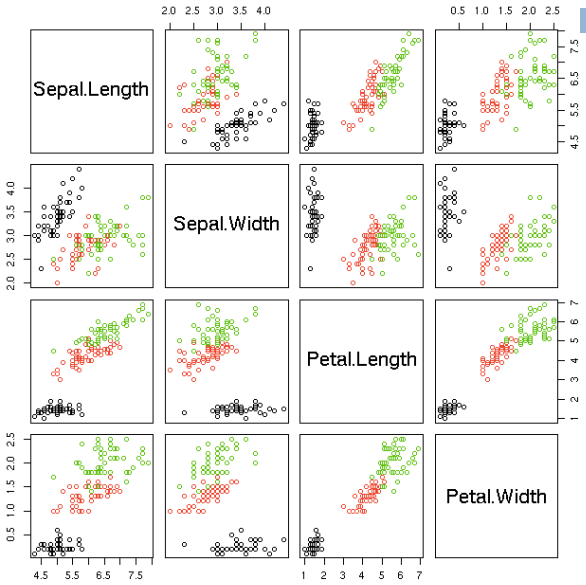
Prática: implementação do Perceptron



34

Prática: implementação do Perceptron

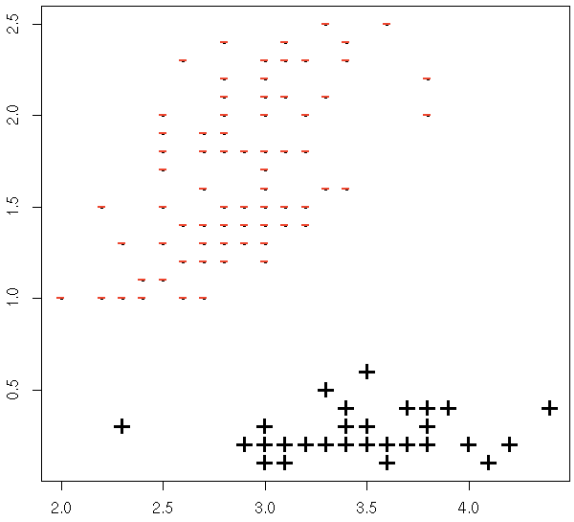
- Conjunto de dados Iris



35

Prática: implementação do Perceptron

- Conjunto de dados Iris
- Sepal Width x Petal Width



36