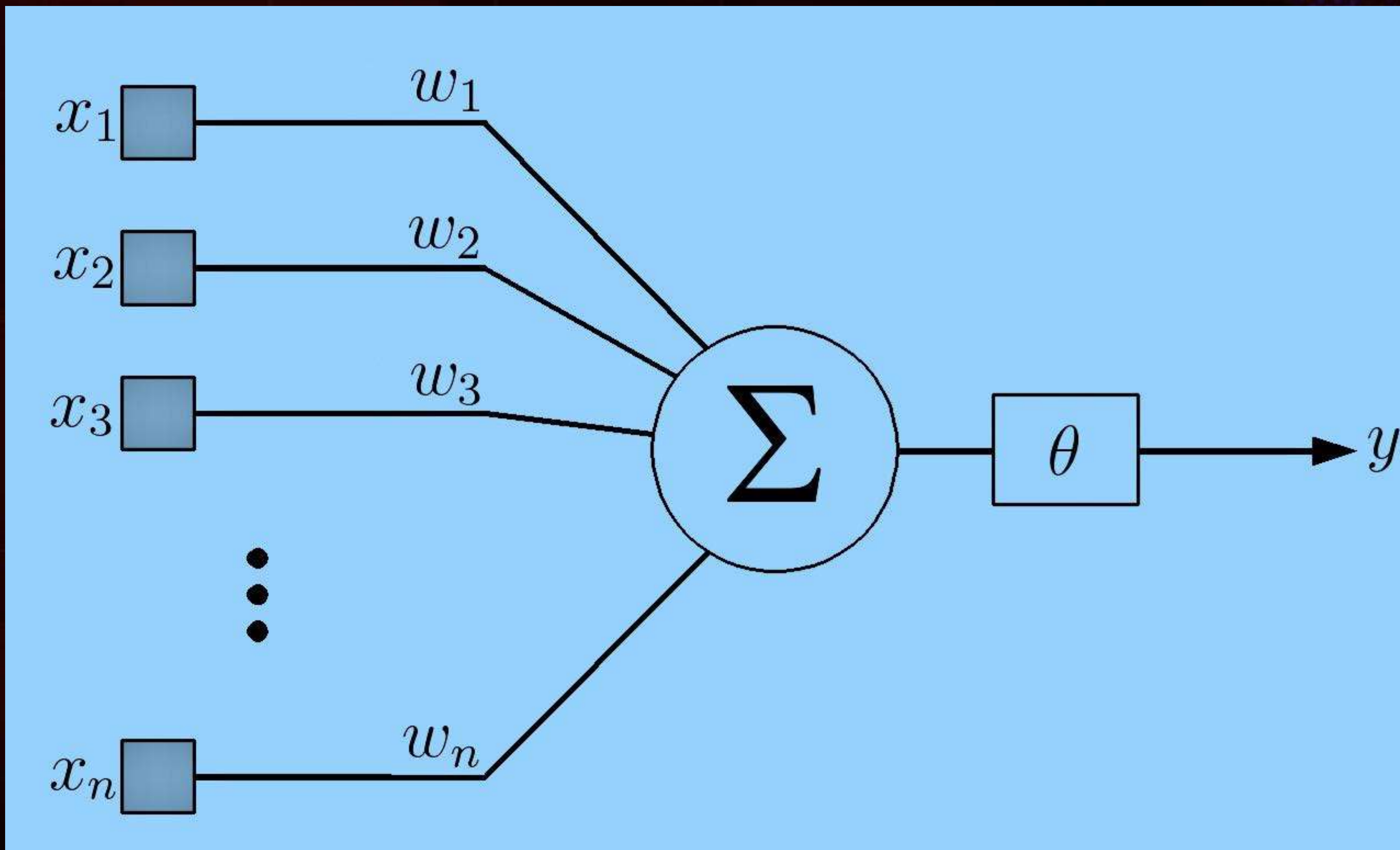
REDES NEURAIS

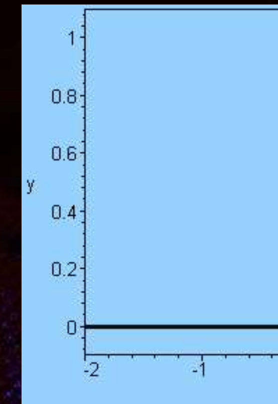
Primeiros modelos: Perceptron e Adaline

TÓPICOS

1. Revisitar o neurônio MCP
2. O Perceptron
3. O Adaline

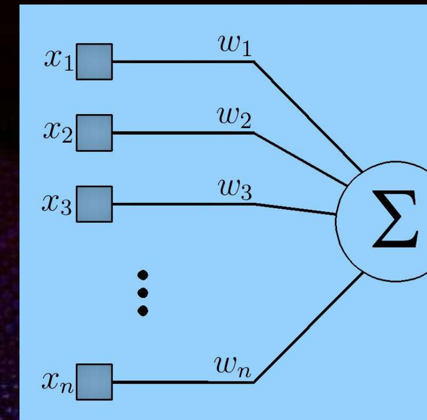
O NEURÔNIO MCP



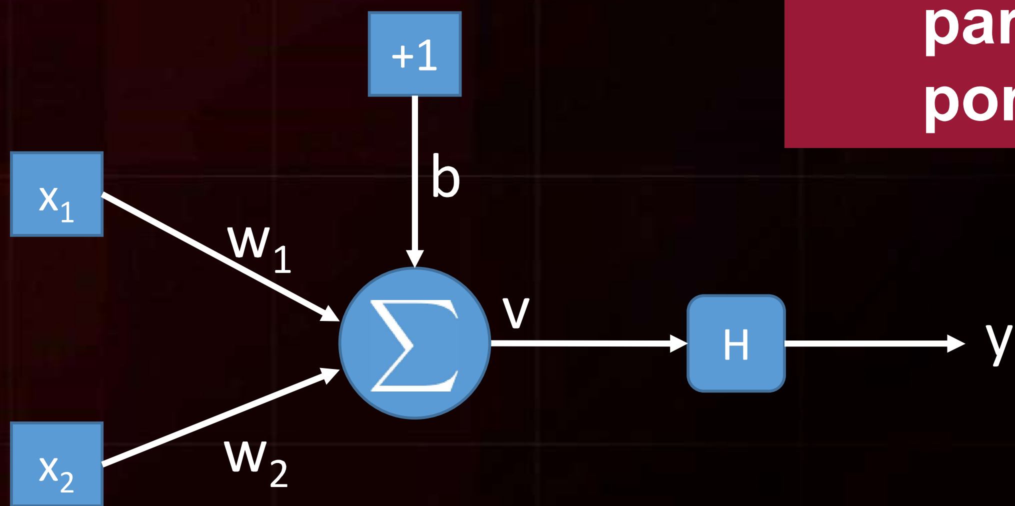


O QUE FAZ O MCP?

1. Representa uma abstração do Neurônio Biológico
2. Pode ser configurado para implementar portas lógicas, i.e. AND, OR
3. Como configurá-lo (treiná-lo)?



MCP: PORTA AND



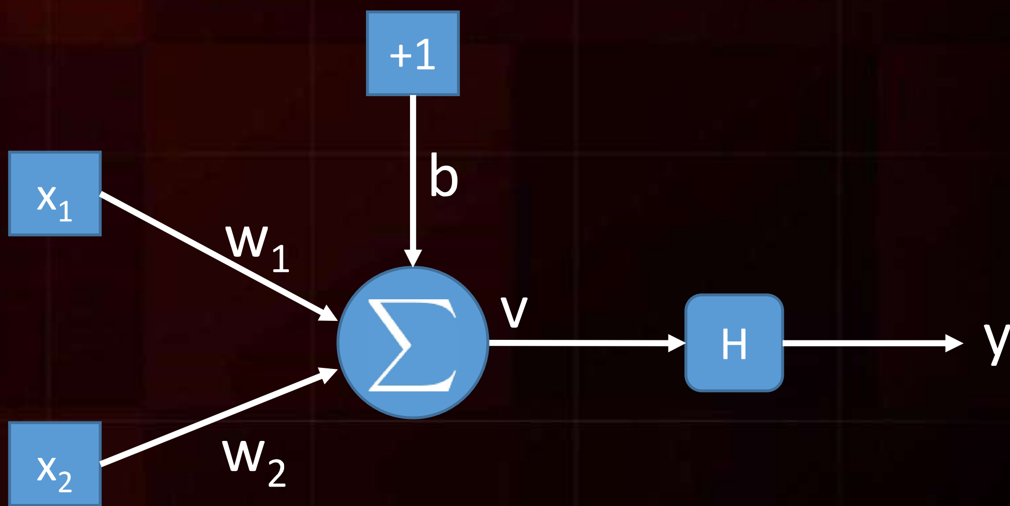
- O que precisamos fazer para configurar a porta AND?

AND

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

- Ajustar os valores de w_1, w_2 e b

MCP: PORTA AND



$$v = \sum_j w_j x_j$$

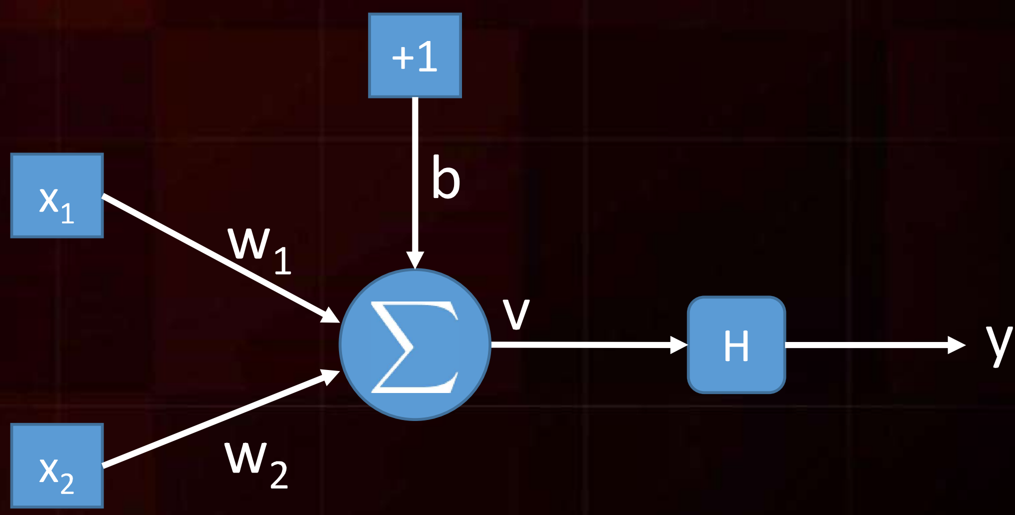
Caso 01: se x_1 ou x_2 forem iguais a 0 (ou ambos), y deve ser 0

$$bx_0 + w_1x_1 + w_2x_2 \leq 0$$
$$b \leq 0$$
$$b + w1 \leq 0$$

- **b = -0,5**
- **w_1 = 0,3**
- **w_2 = 0,3**

x
0
0
1
1

MCP: PORTA AND



$$v = \sum_j w_j x_j$$

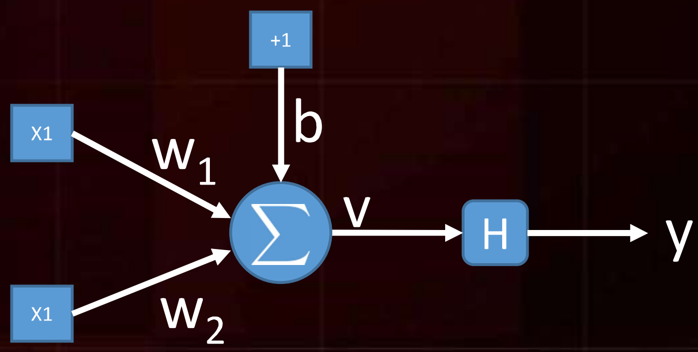
Caso 02: se x_1 e x_2 forem iguais a 1, y deve ser 1

$$bx_0 + w_1x_1 + w_2x_2 > 0$$
$$b + w_1 + w_2 > 0$$

- **b = -0,5**
- **w_1 = 0,3**
- **w_2 = 0,3**

x
0
0
1
1

MCP: PORTA AND



Como podemos interpretar o neurônio MCP neste caso?

b = -0,5 e $w_1 = w_2 = 0,3$

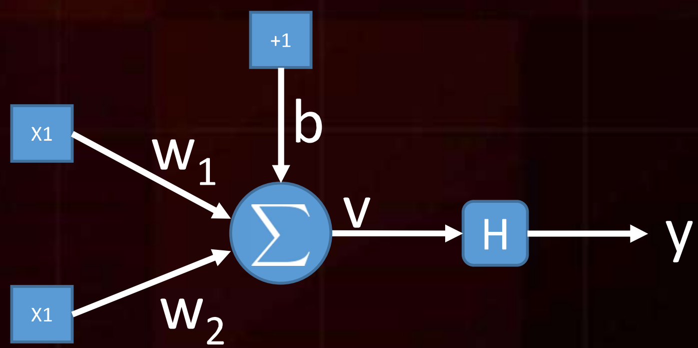
x
0
0
1
1

$$-0,5 + 0,3x_1 + 0,3x_2 = 0$$

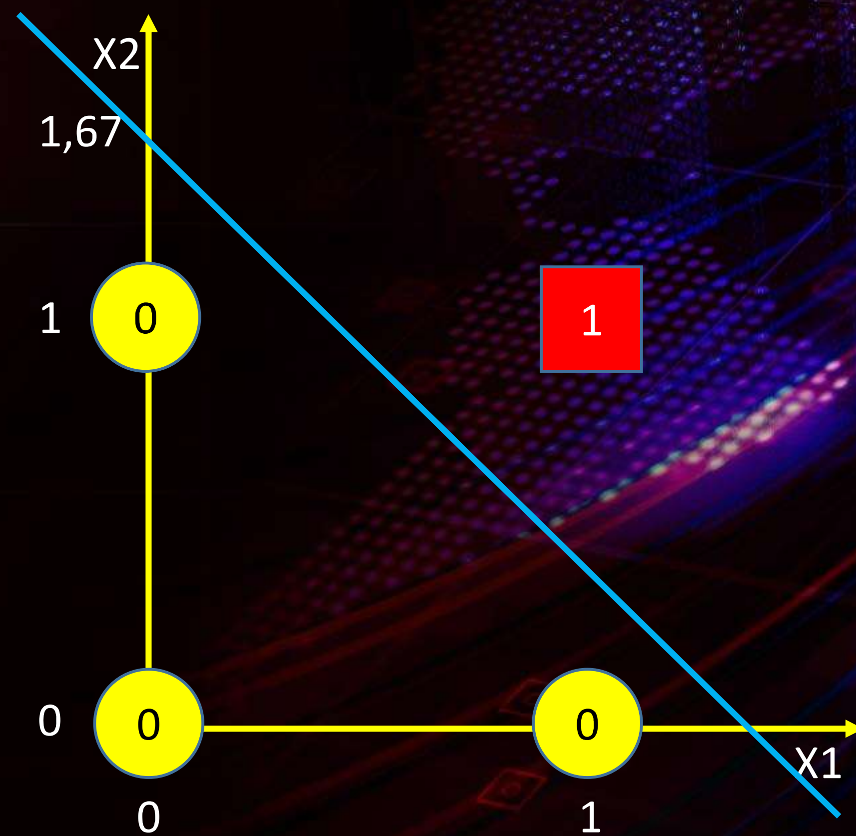
$$x_2 = \frac{-0,3}{0,3}x_1 + \frac{0,5}{0,3}$$

$$x_2 = -1x_1 + 1,67$$

MCP: PORTA AND



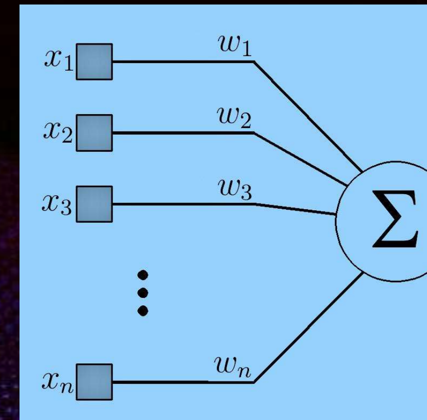
$$x_2 = -1x_1 + 1,67$$



- **O que os MCP representam?**
- **O que o bias representa?**
- **Qual é o problema do MCP?**

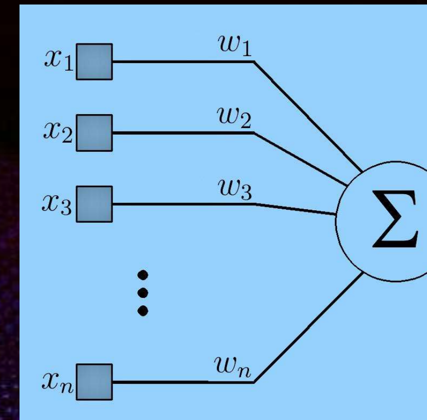
x
0
0
1
1

PERCEPTRON



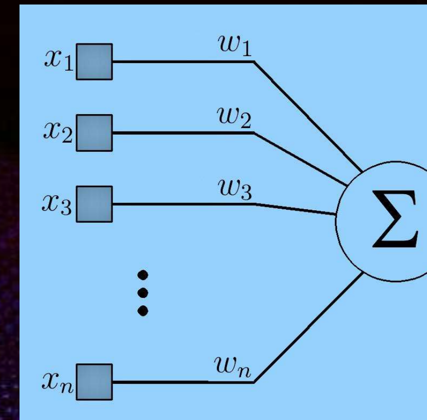
- Proposto por Rosenblatt em 1958
- Associa um algoritmo de aprendizagem ao neurônio MCP: ajuste automático dos pesos via correção de erros
- A rede possui apenas uma camada de neurônios ajustáveis
- Usado para classificação de padrões
- Converge com erro zero se as classes forem linearmente separáveis

PERCEPTRON



- Neurônios assumem saídas binárias: função de ativação degrau
- Algoritmo de aprendizado supervisionado via correção de erros
- Um único neurônio permite resolver problemas de classificação binários
- Com múltiplos neurônios, problemas com várias classes podem ser resolvidos

PERCEPTRON

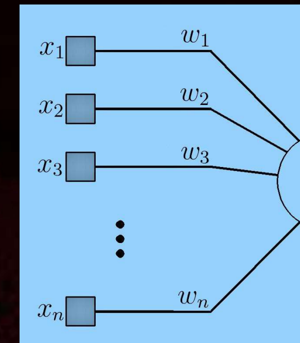


- Regra de atualização dos Pesos:
  - Se o padrão é corretamente classificado, o peso não é alterado
  - Se o padrão for erroneamente classificado, o peso é atualizado por:

$$w(n+1) = w(n) + \eta \left[d(n) - y(n)\right] x(n)$$

$$\Delta w = \eta e(n) x(n)$$

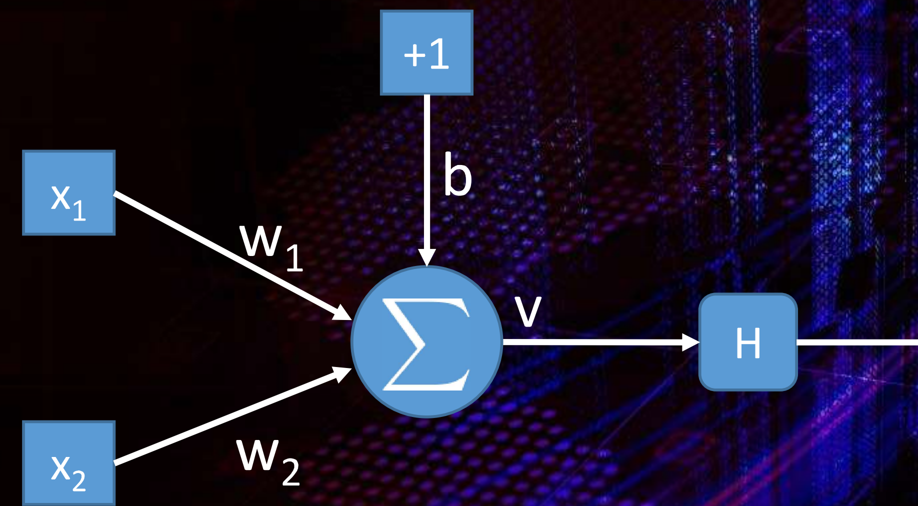
O PERCEPTRON: ALGORITMO



1. Iniciar os pesos sinápticos com valores aleatórios e pequenos ou iguais a zero
2. Aplicar um padrão com seu respectivo valor desejado de saída (d_j) e verificar a saída da rede (y_j)
3. Calcula o erro na saída $e_j = d_j - y_j$;
4. Se $e_j = 0$, volta ao passo 2;
5. Senão: atualiza os pesos: $\Delta w = \eta e(n) x(n)$
6. volta ao passo 2.

EXEMPLO: PORTA AND

AND	x_0	x_1	x_2	d
Entrada 1:	1	0	0	0
Entrada 2:	1	0	1	0
Entrada 3:	1	1	0	0
Entrada 4:	1	1	1	1



Peso inicial: $w_0= 0$, $w_1= 0$, $w_2=0$

Taxa de aprendizado: eta = 0.5

ÉPOCA 01

AND	x_0
Entrada 1:	1
Entrada 2:	1
Entrada 3:	1
Entrada 4:	1

Entrada 1: $y = f(w_0x_0 + w_1x_1 + w_2x_2)$

$= f(0\times1+0\times0+0\times0) = f(0) = 0$ $\quad$ [y = d]

Entrada 2: $y = f(0\times1+0\times1+0\times0) = f(0) = 0$ $\quad$ [y = d]

Entrada 3: $y = f(0\times1+0\times0+0\times1) = f(0) = 0$ $\quad$ [y = d]

Entrada 4: $y = f(0\times1+0\times1+0\times1) = f(0) = 0$ $\quad$ [y != d]

$w_0 = w_0+(d-y)x_0 = 0 + 0{,}5 \times(1-0) \times1=0{,}5$

$w_1 = w_1+(d-y)x_1 = 0 + 0{,}5 \times(1-0) \times1=0{,}5$

$w_2 = w_2+(d-y)x_2 = 0 + 0{,}5 \times(1-0) \times1=0{,}5$

ÉPOCA 02

AND	x_0
Entrada 1:	1
Entrada 2:	1
Entrada 3:	1
Entrada 4:	1

Entrada 1: $y = f(0,5 \times 1 + 0,5 \times 0 + 0,5 \times 0) = f(0.5) = 1$

$w0 = w0 + (d-y)x0 = 0,5 + 0,5 \times (0-1) \times 1 = 0$

$w1 = w1 + (d-y)x1 = 0,5 + 0,5 \times (0-1) \times 0 = 0,5$

$w2 = w2 + (d-y)x2 = 0,5 + 0,5 \times (0-1) \times 0 = 0,5$

Entrada 2: $y = f(0 \times 1 + 0.5 \times 0 + 0.5 \times 1) = f(0.5) = 1$

$w0 = w0 + (d-y)x0 = 0 + 0,5 \times (0-1) \times 1 = -0,5$

$w1 = w1 + (d-y)x1 = 0,5 + 0,5 \times (0-1) \times 0 = 0,5$

$w2 = w2 + (d-y)x2 = 0,5 + 0,5 \times (0-1) \times 1 = 0$

APÓS ALGUMAS ÉPOCAS

AND	x_0
Entrada 1:	1
Entrada 2:	1
Entrada 3:	1
Entrada 4:	1

Entrada 1: $y = f(-1{,}0 \times 1 + 1 \times 0 + 0{,}5 \times 0) = f(-1{,}0) = 0$

Entrada 2: $y = f(-1{,}0 \times 1 + 1 \times 0 + 0{,}5 \times 1) = f(-0{,}5) = 0$

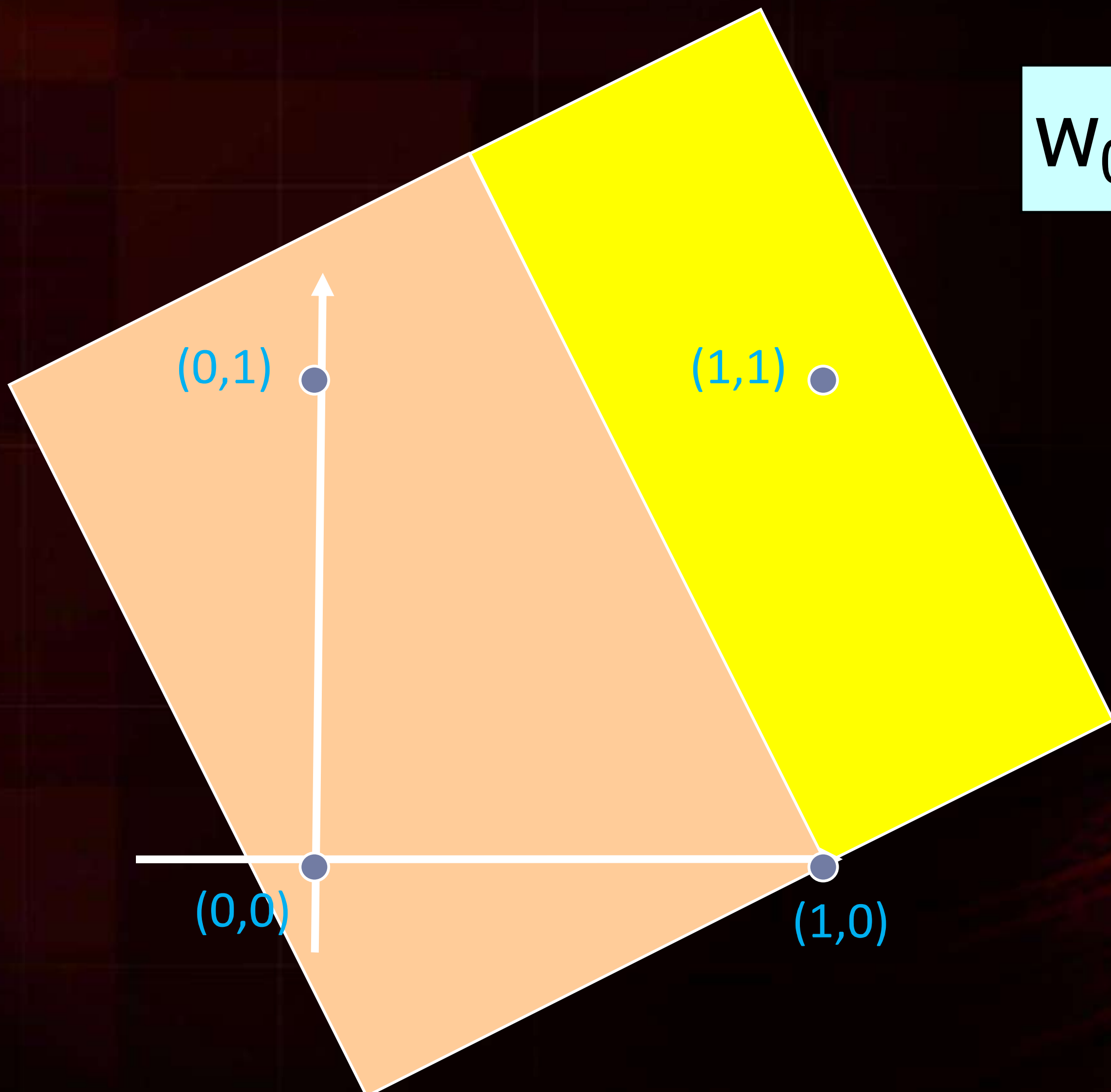
Entrada 3: $y = f(-1{,}0 \times 1 + 1 \times 1 + 0{,}5 \times 0) = f(0{,}0) = 0$

Entrada 4: $y\ f(-1{,}0 \times 1 + 1 \times 1 + 0{,}5 \times 1) = f(0{,}5) = 1$

$w_0 = -1,\ w_1 = 1,\ w_2 = 0.5$

INTERPRETAÇÃO GEOMÉTRICA

w_0 = -1, w_1= 1, w_2 =



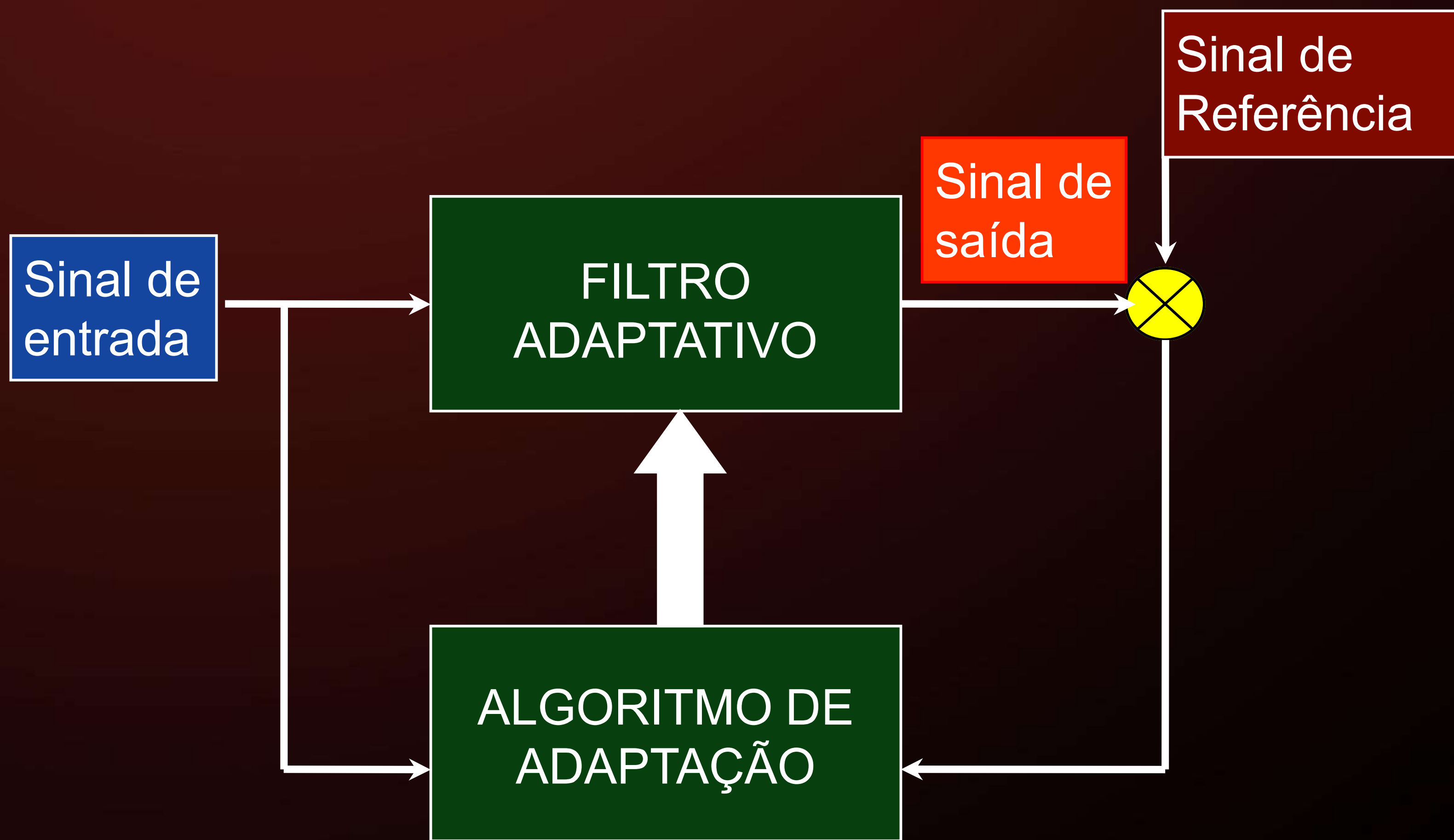
ADALINE: FILTRAGEM ADAPTATIVA

- Filtros adaptativos são dispositivos autoajustáveis que modificam seus parâmetros de acordo com critérios preestabelecidos.
    - Em ambiente estacionário, acompanham a solução ótima do problema.
    - Em ambiente não estacionário, acompanham as modificações do sinal envolvido.
- Base para o desenvolvimento do Adaline

ADALINE

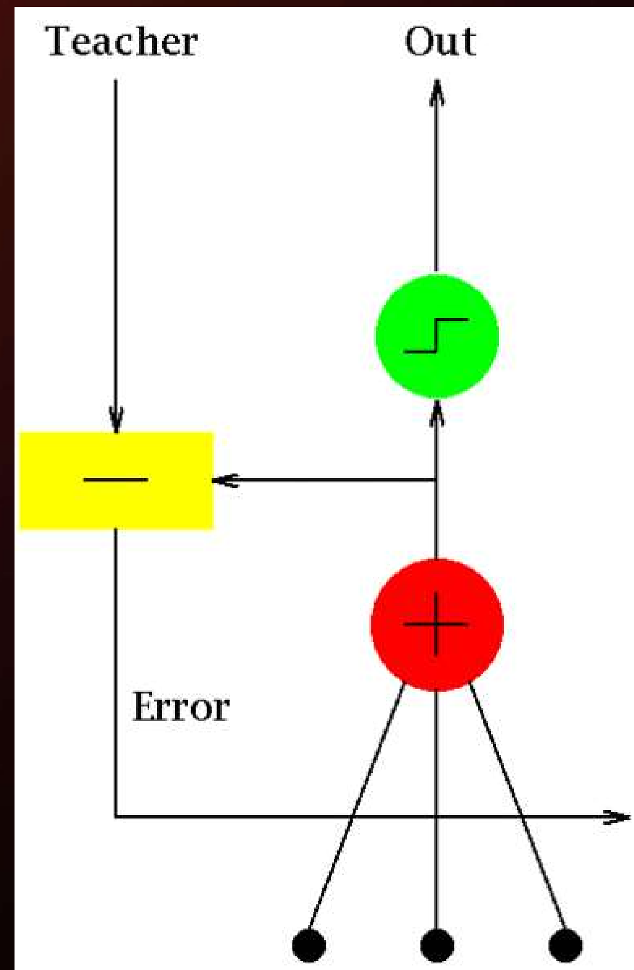
- ADAptive LINear Element ou ADAptive LInear NEuron
- Concebido por Widrow e Hoff (1960)
- Método linear adaptável para classificações de padrões, filtros, etc.,
- Possui saída contínua (linear)
- Usa o algoritmo LMS para sua operação (regra delta)

ADALINE


Sinal de entrada
FILTRO ADAPTATIVO
Sinal de saída
Sinal de Referência
ALGORITMO DE ADAPTAÇÃO

ADALINE

- O Modelo Adaline tem seus pesos adaptados em função do erro de sua saída linear (antes da aplicação da função de ativação)



Fonte: https://en.wikipedia.org/wiki/ADALINE

ADALINE

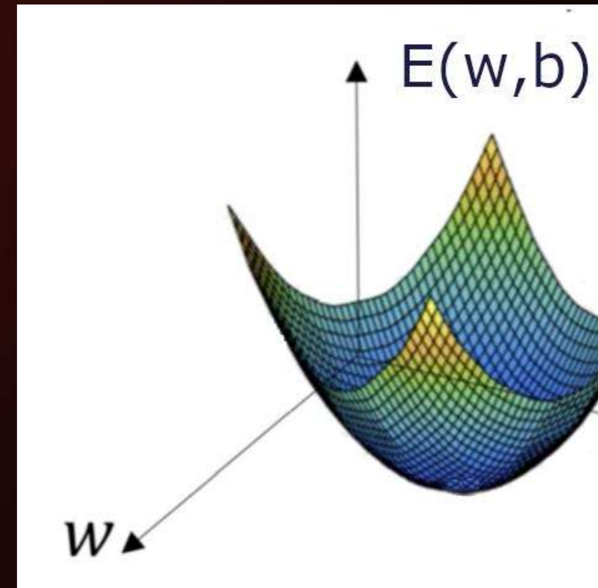
- O algoritmo de aprendizagem tem como objetivo minimizar o erro das saídas em relação aos valores desejados (conjunto de treinamento)

- A função de custo a ser minimizada é a soma dos erros quadráticos:

$$E(n) = \frac{1}{2}\sum(d_k(n) - y_k(n))^2$$

ADALINE

- Processo de minimização do erro quadrático pelo método do Gradiente Descendente

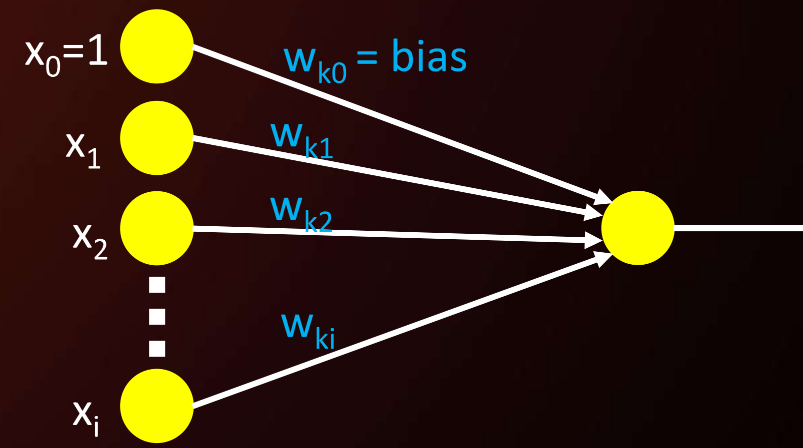


$$\Delta w_{ki} = -\eta \frac{\partial E}{\partial w_{ki}}$$

- Cada peso sináptico i do neurônio k é atualizado proporcionalmente ao negativo da derivada parcial do erro em relação ao peso

Adaptado de: https://builtin.com/data-science/gradient-descent

$$\Delta w_{ki} = -\eta \frac{\partial E}{\partial w_{ki}} = -\eta \frac{\partial E}{\partial y_k} \frac{\partial y_k}{\partial w_{ki}}$$



$$E(n) = \frac{1}{2} \sum (d_k(n) - y_k(n))^2$$

$$\frac{\partial E}{\partial y_k} = (d_k - y_k)(-1)$$

$$y_k = \sum_i x_i w_{ki}$$

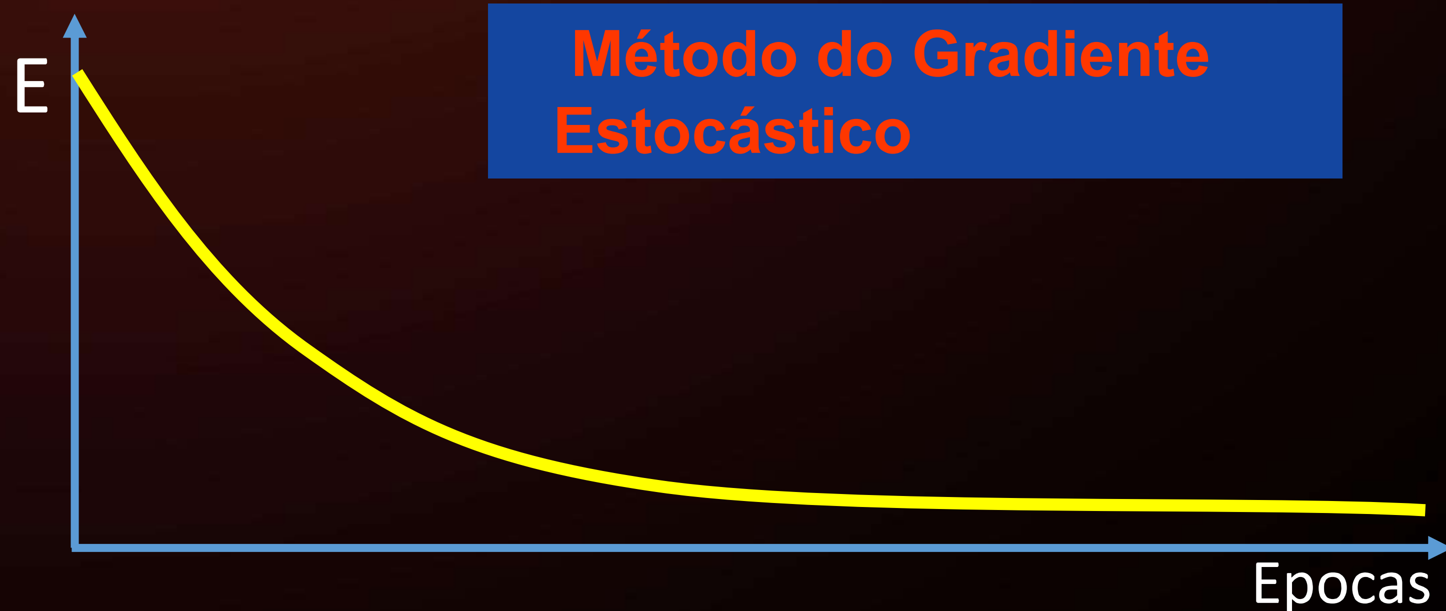
$$\frac{\partial y_k}{\partial w_{ki}} = x_i$$

$$\Delta w_{ki} = \eta (d_k - y_k) x_i$$

ADALINE

- É importante observar que a regra Delta (LMS) produz uma estimativa do vetor de pesos que resultaria da utilização do da descida mais íngreme (gradiente descendente)

Método do Gradiente Estocástico



ADALINE VS. PERCEPTRON

- Regra de atualização

$$\Delta w_{ki} = \eta(d_k - y_k)x_i$$

- Perceptron → Neurônio não-linear
- Adaline → Neurônio linear
- Ambos podem ser usados para resolver problemas linearmente separáveis

O QUE VIMOS?

- Revisitamos o neurônio MCP
- Conhecemos os primeiros modelos de redes neurais:
  - Perceptron e
  - Adaline

PRÓXIMA VIDEOAULA

- Rede Multilayer Perceptron (MLP)