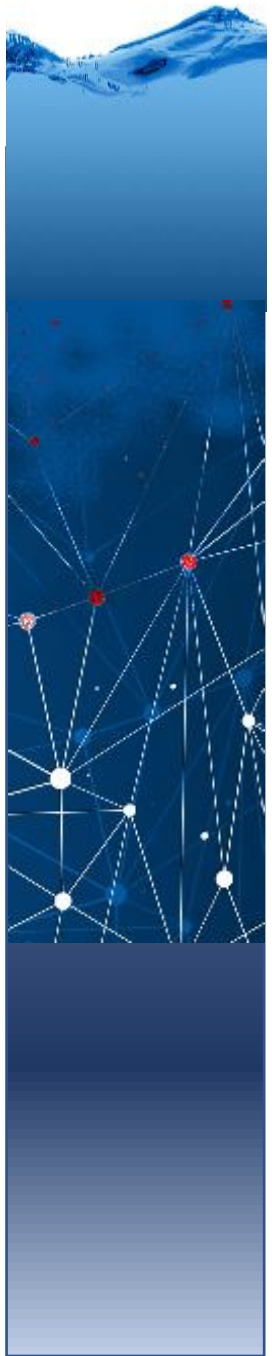


Programa de Pós-Graduação em Engenharia de Recursos Hídricos e Ambiental

PPGERHA

Redes Neurais Artificiais
o perceptron

Jorge Centeno

Introdução

O princípio das redes neurais artificiais é um modelo matemático do fluxo e processamento de informações no sistema nervoso central de um animal (em particular o cérebro).

´”É possível simular o funcionamento do cérebro com algoritmos?”

Elas devem ser capazes de:

- realizar o aprendizado de máquina
- Reconhecer de padrões.

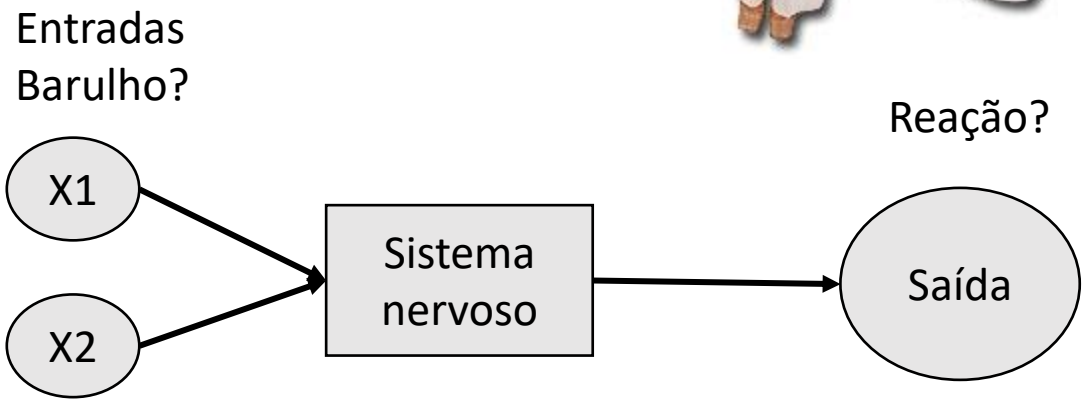
A primeira **rede neural** foi concebida por Warren McCulloch e Walter Pitts em 1943.

O modelo

O sistema nervoso humano detecta estímulos externos e internos e desencadeia as respostas musculares e glandulares.

Ele é formado, basicamente, por células que se interconectam formando os chamados circuitos neurais.

Através desses circuitos, o organismo é capaz de produzir respostas fixas (por exemplo, os reflexos), ou comportamentos variáveis.



O neurônio

Principal componente do sistema nervoso (unidade anatomo-fisiológica).

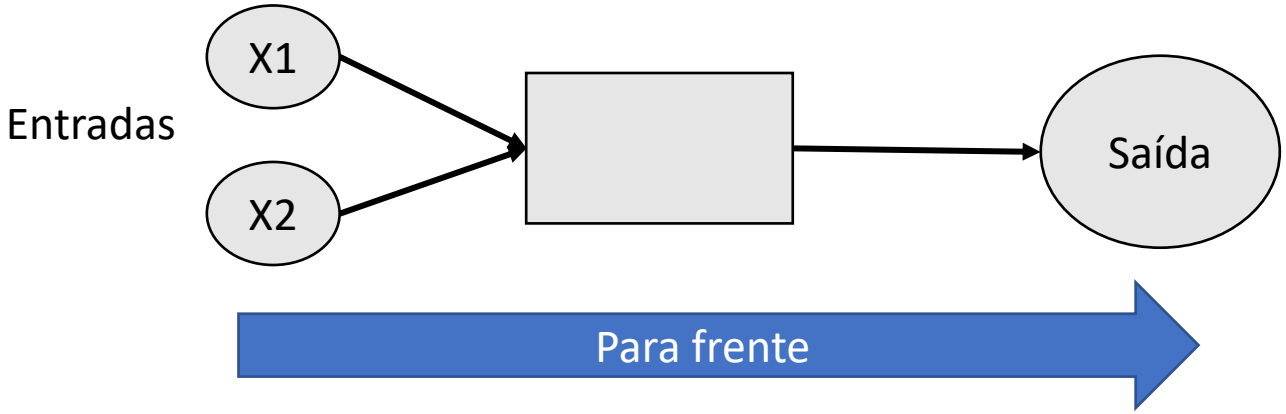
- o **corpo do neurônio** (soma)
- o **axônio** É responsável pela condução do impulso nervoso para o próximo neurônio;
- os **dendritos** ramificações menores responsáveis pela comunicação entre os neurônios através das sinapses. Basicamente, cada neurônio, possui uma região receptiva e outra efetora em relação a condução da sinalização.

O perceptron

é um tipo de rede neural artificial inventada em 1958 por Frank Rosenblatt no Cornell Aeronautical Laboratory.

É o tipo mais simples de rede neural feedforward: um classificador linear.

Feed+forward = o fluxo de informação vai “para frete”. Significa que parte da “entrada”, passa pelo sistema e termina na “saída”



Matematicamente

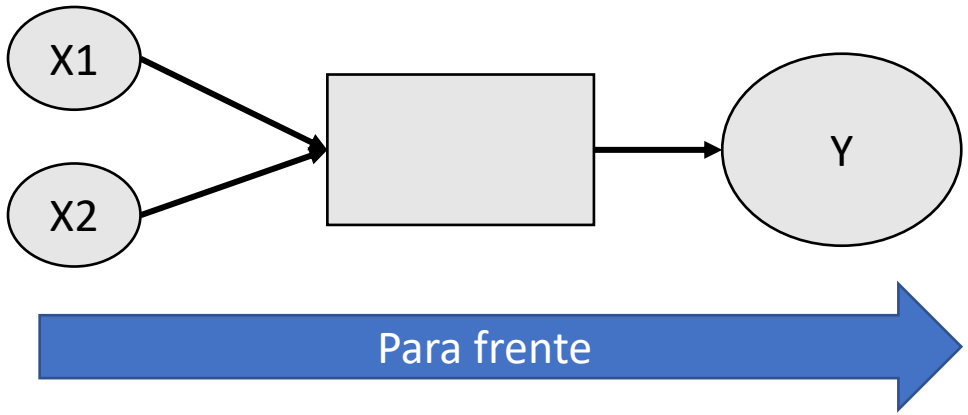
Considerando duas entradas X1 e X2 e uma saída Y.

As duas entradas devem ser combinadas para produzir um único sinal de saída (Y). A forma mais simples é somar as entradas

$$S = X1 + X2$$

Porém, para levar em consideração a “importância” das entradas, a soma pode ser uma soma ponderada:

$S = p_1 * X_1 + p_2 * X_2$



saída

O perceptron somente pode responder 0 ou 1.

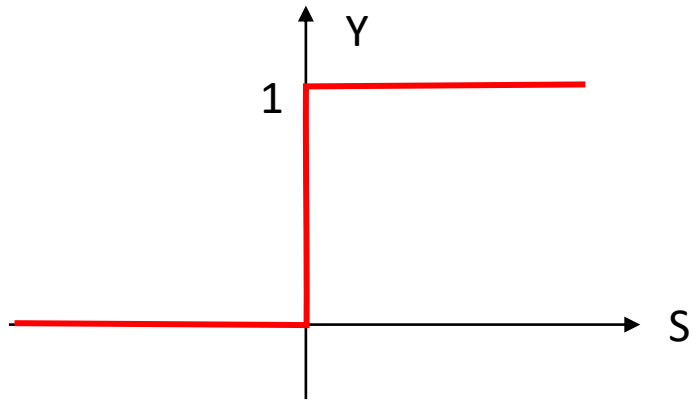
A reação do perceptron é positiva apenas quando a soma é positiva.

Então, a saída deve ser modulada a partir da soma computada

Se $S= p_1*X_1 + p_2*X_2 > 0$ Y=1

Caso contrário Y=0

Ou graficamente:



Função “degrau”

Matematicamente

A Soma $S= p_1*X_1 + p_2*X_2$

Pode ter resultados positivos ou negativos. (sim/não) ou zero.

É zero quando

$0= p_1*X_1 + p_2*X_2$

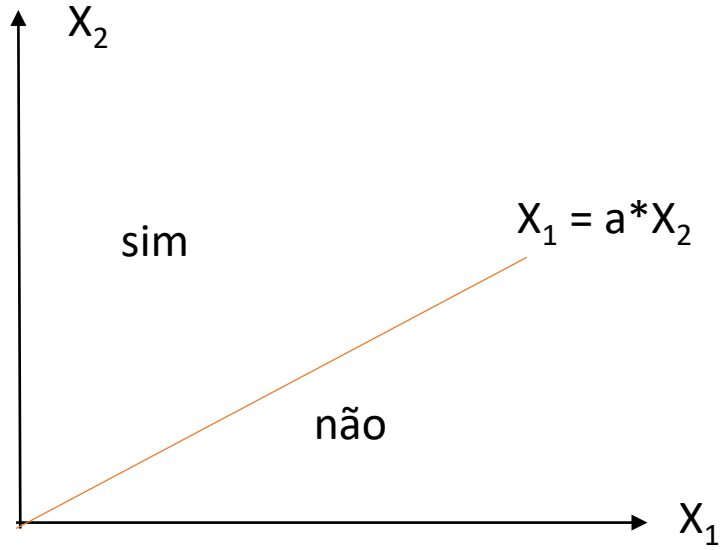
Ou

$p_1*X_1 = -p_2*X_2$

$X_1 = -(p_{2/} p_1)*X_2$

$X_1 = a*X_2$

Uma reta passando pela origem



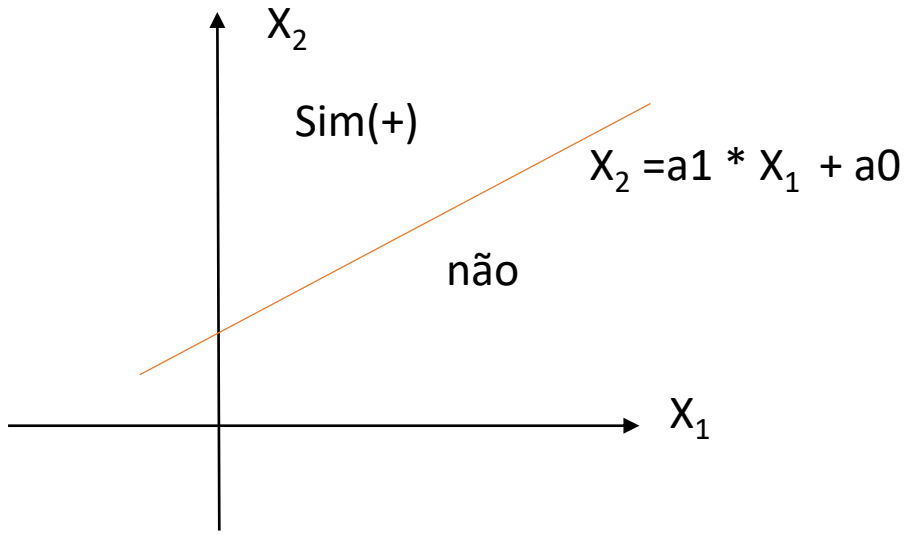
Bias

Para simular situações onde a superfície de decisão não necessariamente passa pela origem, podemos adicionar uma constante (bias)

$S = p_1 * X_1 + p_2 * X_2 + b$

Isto pode ser escrito como:

$$S = \begin{bmatrix} X_1 & X_2 & 1 \end{bmatrix} * \begin{bmatrix} p_1 \\ p_2 \\ b \end{bmatrix}$$



Perceptron

Então, o perceptron basicamente funciona assim:

a) Ler as entradas X_1 X_2

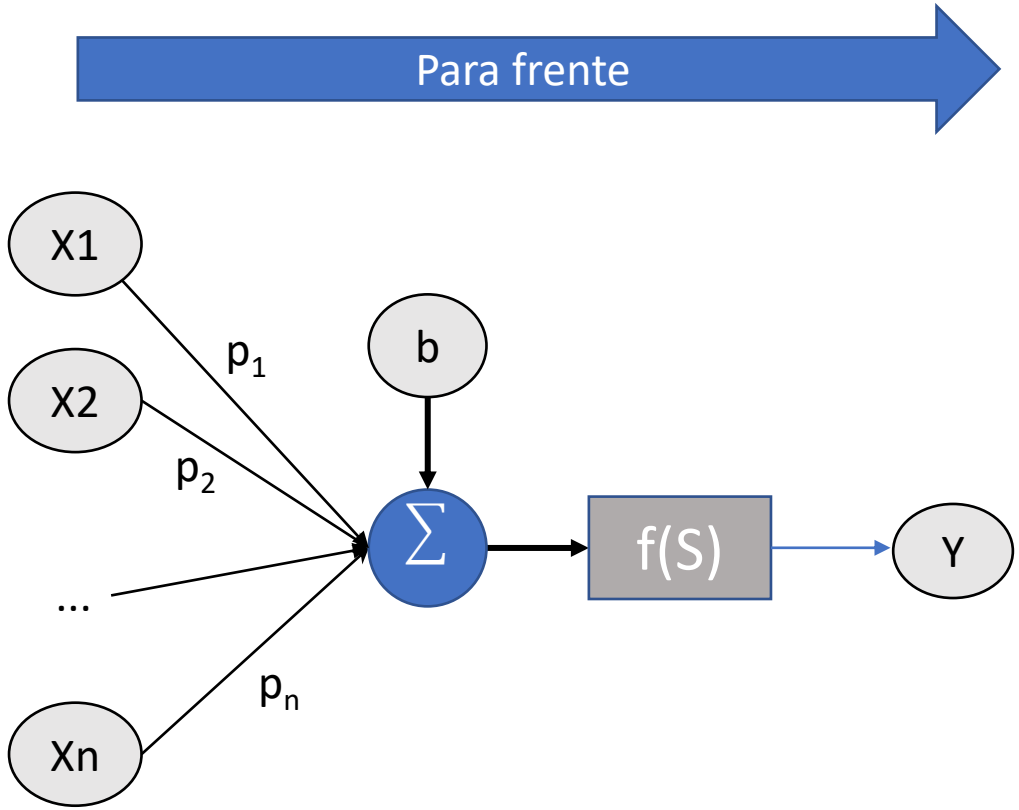
b) Calcular a soma ponderada, incluindo um bias

$$S= p_1*X_1 + p_2*X_2 + b$$

c) Modular a soma para gerar uma saída

$Y= f(S)$

O problema reside em conhecer os pesos...



Backpropagation

Retro-propagação

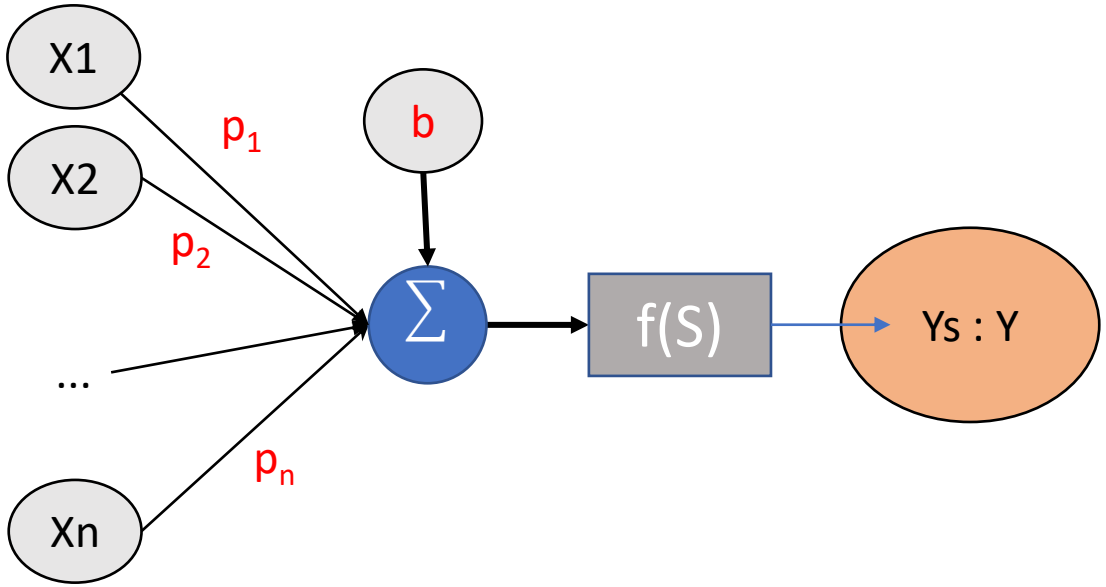
Como a princípio os pesos são desconhecidos, o perceptron parte de uma situação com pesos aleatórios.

O que ocorre quando a saída produzida “Ys” é diferente da saída esperada Y, ou seja, o perceptron erra?

O erro deve ser atribuído aos pesos, que devem estra errados.

Então, os pesos devem ser corrigidos (atualizados) proporcionalmente ao erro cometido e ao tamanho da variável envolvida

Correção= f(Erro, X)



Erro grande, correções grandes
Erro pequeno, pequenas correções

Valor da variável grande: maior correção
Valor da variável pequeno: menor correção

Backpropagation

Então

a) se parte da saída

b) Calcula-se o erro

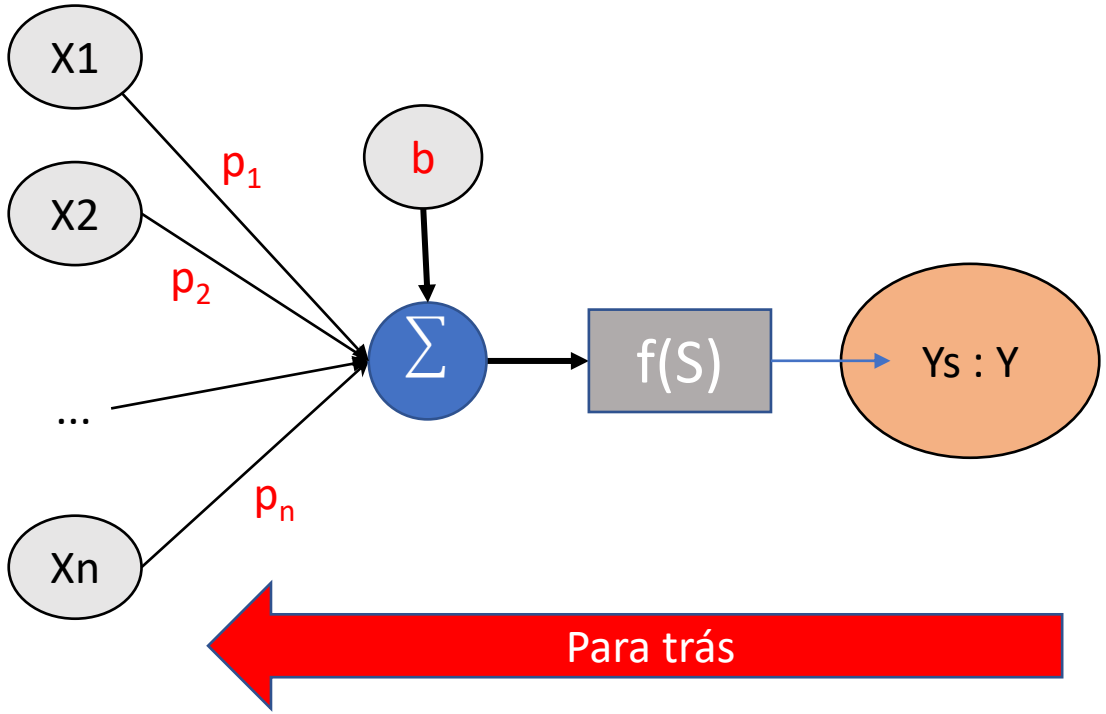
c) Ajusta-se os pesos de forma que a saída seja correta

Ou seja, de trás para frente

BACK

E como o erro é distribuído entre os elementos de entrada:

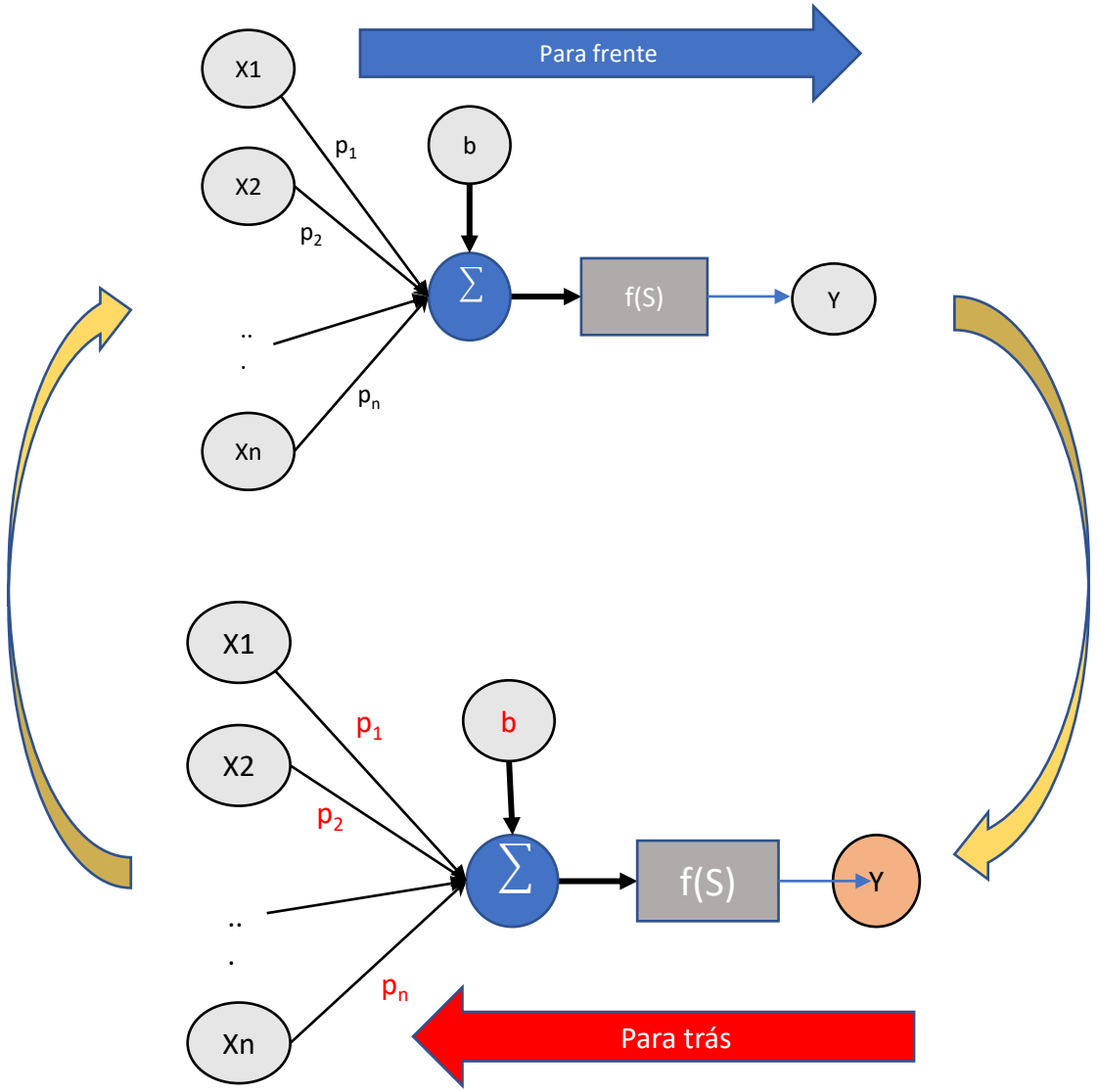
“Propaga-se o erro”



Aprendizado

O processo deve ser repetido para todas as amostras de treinamento até se atingir um ponto onde o erro cometido com todas as amostras seja mínimo.

Este processo iterativo permite ajustar os pesos de maneira que se encontre a solução a um problema determinado.



A atualização dos pesos atende a seguinte lei:

$$p_i(t+1) = p_i(t+1) + \Delta_i(t)$$

p=peso

t=iteração

i= amostra

Ou seja, uma pequena correção “delta” é somada ao peso.

Se ocorreu erro, a correção é proporcional ao erro cometido

Erro= $[\, Y(t) - Ys(t)]$

Y = valor verdadeiro

Ys=valor produzido pelo perceptron

A correção é:

$$\Delta_i(t) = \eta * [Y(t) - Ys(t)] * X_i(t)$$

Aqui:

X representa a entrada associada ao peso em questão.

η é a taxa de aprendizado, uma fração entre zero e um (geralmente 0.5). Controla que as correções não sejam muito grandes para evitar oscilações grandes.

Então:

$$p_i(t+1) = p_i(t) + \eta [Y(t) - Ys(t)] * X_i(t)$$

Regra

Lembre: O perceptron somente produz saídas binárias

Correção dos pesos:

- não ocorre variação no peso se a saída estiver correta.
- caso contrario, cada peso é:
  - incrementado quando a saída é menor que o valor esperado

$$Y - Ys > 0$$

  - diminuído quando a saída é maior que o valor esperado

$$Y - Ys < 0$$

Exemplo clássico

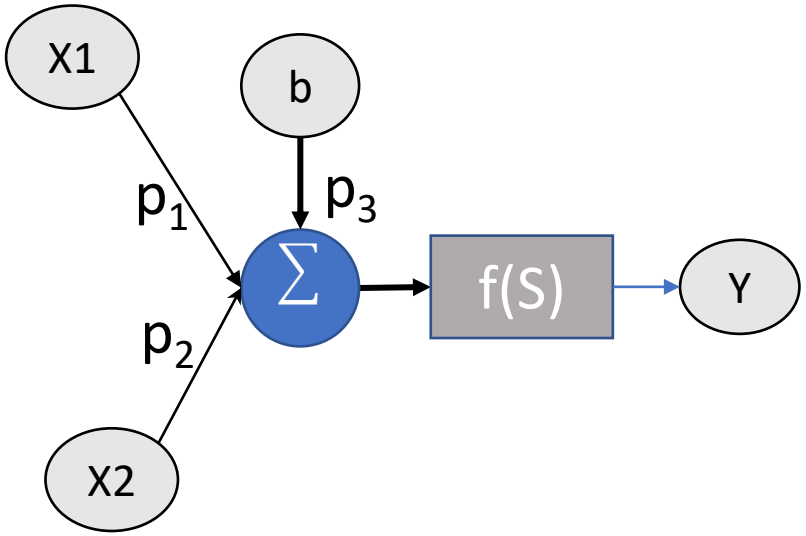
Dadas duas entradas (X1,X2) e a respectiva verdadeira saída, determinar os pesos do perceptron:

X2	X1	b	Y
0	0	1	0
0	1	1	0
1	0	1	1
1	1	1	1

Para começar... Iniciar com todos os pesos nulos

P=0

P3=peso do bias



Uma iteração

```
iteração 0
i    [X1 X2 b] [pesos  ] Soma          true|computed  erro
0 ) [0, 0, 1] [0, 0, 0] 0             [ 0 | 0 ]       0
1 ) [1, 0, 1] [0, 0, 0] 0             [ 0 | 0 ]       0
2 ) [0, 1, 1] [0, 0, 0] 0             [ 0 | 0 ]       0
3 ) [1, 1, 1] [0, 0, 0] 0     [ 1 | 0 ]       1
Ocorreu erro na quarta amostra: Corrigir os pesos:
a) Calcular a correção para cada elemento (X1, X2 e b)
        Delta= 0.5 * 1 [1 1 1]=[.5 .5 .5]
b) Corrigir cada peso
    P1=p1+0.5=0.5
    P2=p2+0.5=0.5
    P3=p3+0.5=0.5
Novos pesos p=[0.5, 0.5, 0.5]
```

De novo...

iteração 1

i [X1 X2 b] [pesos] Soma true|computed erro

0 [0, 0, 1] [0.5, 0.5, 0.5] 0.5 [0 | 1] erro= -1

Ocorreu erro na primeira amostra: Corrigir os pesos:

a) Calcular a correção para cada elemento (X1, X2 e b)

Delta= 0.5 * (-1)* [0 0 1]=[0 0 -.5]

b) Corrigir cada peso

P1=p1+0.0=0.5+0.0

P2=p2+0.0=0.5+0.0

P3=p3-0.5=0.5-0.5

Novos pesos p=[0.5, 0.5, 0.0]

De novo...

iteração 1

i [X1 X2 b] [pesos] Soma true|computed erro

1 [1, 0, 1] [0.5, 0.5, 0.0] 0.5 [0 | 1] erro= -1

Ocorreu erro na segunda amostra:

a) Calcular a correção para cada elemento

Delta= 0.5 * (-1)* [1 0 1]=[-0.5 0 -.5]

b) Corrigir cada peso

P1=p1+0.0=0.5-0.5

P2=p2+0.0=0.5

P3=p3-0.5=0.0-0.5

Novos pesos p= [0.0, 0.5, -0.5]

De novo...

Resumindo a segunda iteração (t=1)

i	[X1 X2 b]	[pesos]	Soma	true|computed	erro
0	[0, 0, 1]	[0.5, 0.5, 0.5]	0.5	[0 | 1]	erro= -1
	novo	[0.5, 0.5, 0.0]			
1	[1, 0, 1]	[0.5, 0.5, 0.0]	0.5	[0 | 1]	erro= -1
	novo	[0.0, 0.5, -0.5]			
2	[0, 1, 1]	[0.0, 0.5, -0.5]	0.0	[0 | 0]	erro= 0
3	[1, 1, 1]	[0.0, 0.5, -0.5]	0.0	[1 | 0]	erro= 1
	novo	[0.5, 1.0, 0.0]			

Como houve erro corrigido, então devemos executar uma nova iteração até que não ocorra erro ou atingir o erro mínimo aceitável ou o máximo de iterações...

Em py

```
# perceptron simples com duas entradas X1 e X2 e uma saida Y
import numpy as np
import matplotlib.pyplot as plt
# dados de entrada
X1 =[0, 1, 0, 1]
X2 =[0, 0 ,1, 1]
Y =[0, 0, 0, 1]
b=1 # bias
taxa=0.5                #taxa de aprendizado
p=[0, 0 ,0]             #inicializa os pesos
soma_erro=1             # soma dos erros cometidos, deve ser zero ao final
t=0                     # contador de iterações
```

```
while soma_erro>0:
        soma_erro=0;
        for i in range(4): #varrer todos os elementos, 4 neste caso
                v=[X1[i], X2[i], b] #cria vetor
                S=np.dot(v,p) #produto vetor * pesos
                y1=0 #se a soma for abaixo de zero saida =0
                if S>0: # se nao, saida=1
                        y1=1
                erro=Y[i]-y1 #diferenca entre real e calculado
                soma_erro=soma_erro+np.abs(erro) #acumule os erros
                print(i,')', v,p,'S=', S, '(Y|y1)=[',Y[i],'|',y1, '] erro=',erro)
                if erro!=0:    #se houve erro, corrigir pesos
                        for j in range(3):
                                delta=taxa*erro*v[j]
                                p[j]=p[j]+delta
                                print (' novo',p )
        print('soma do erro=', soma_erro)
        t=t+1
```

interpretação

A solução pode ser entendida como a divisão do espaço das entradas com uma reta

$0 = p_1 * X_1 + p_2 * X_2 + b$

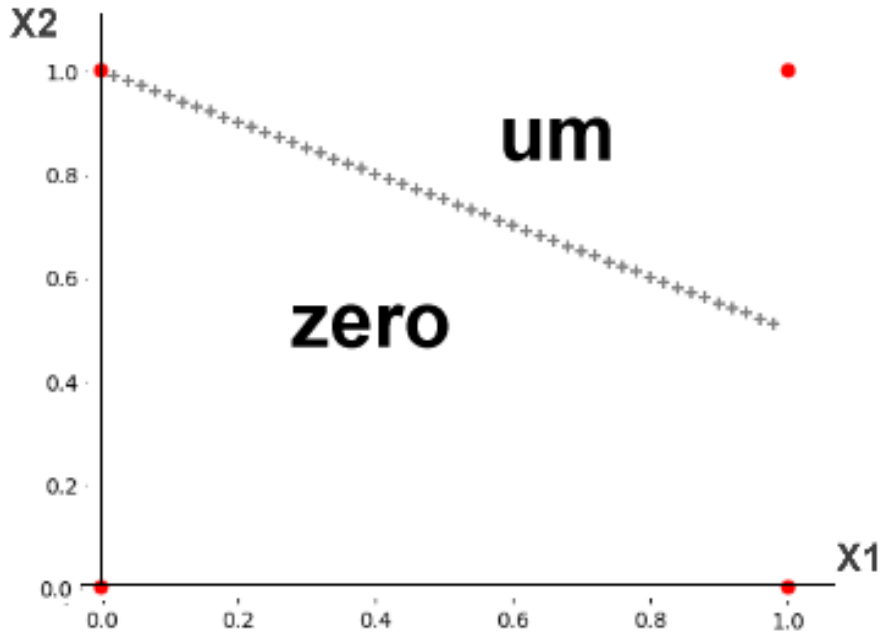
$p_2 * X_2 = -b - p_1 * X_1$

$X_2 = -(b/p_2) - (p_1/p_2) * X_1$

$X_2 = a0 + a1 * X_1$

$a0 = -(b/p_2)$

$a1 = -(p_1/p_2)$



limitações

O perceptron, como descrito acima, por ser muito simples, somente pode resolver problemas linearmente separáveis.

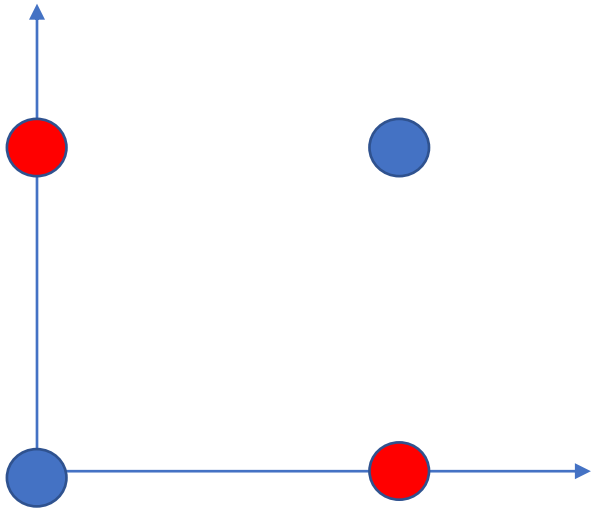
Porém, ele serve como exemplo para mostrar o processo de aprendizado backpropagation e o papel dos pesos aplicados às variáveis de entrada.

Altere o problema

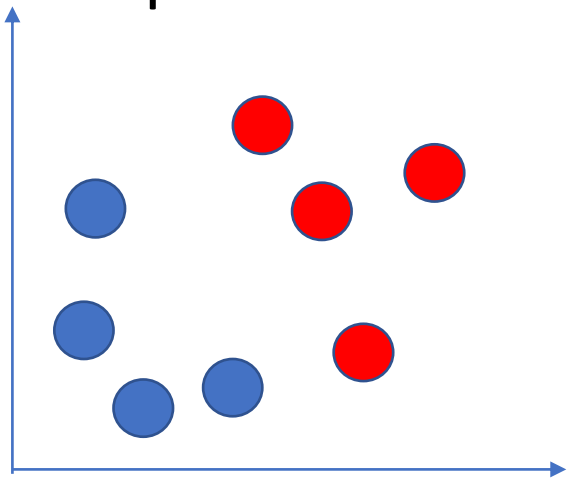
Para verificar o programa, altere a classificação dos pontos e confira o resultado.

Lembre-se que tem que ser um problema linearmente separável!

Não vale isto:



mas pode isto?



Para casa

Ao lado você encontra parte das notas de uma disciplina. Foram feitas duas provas e a nota final (média) é a soma ponderada das notas das provas (N1 e N2).

Você recebeu por email o conjunto total de dados

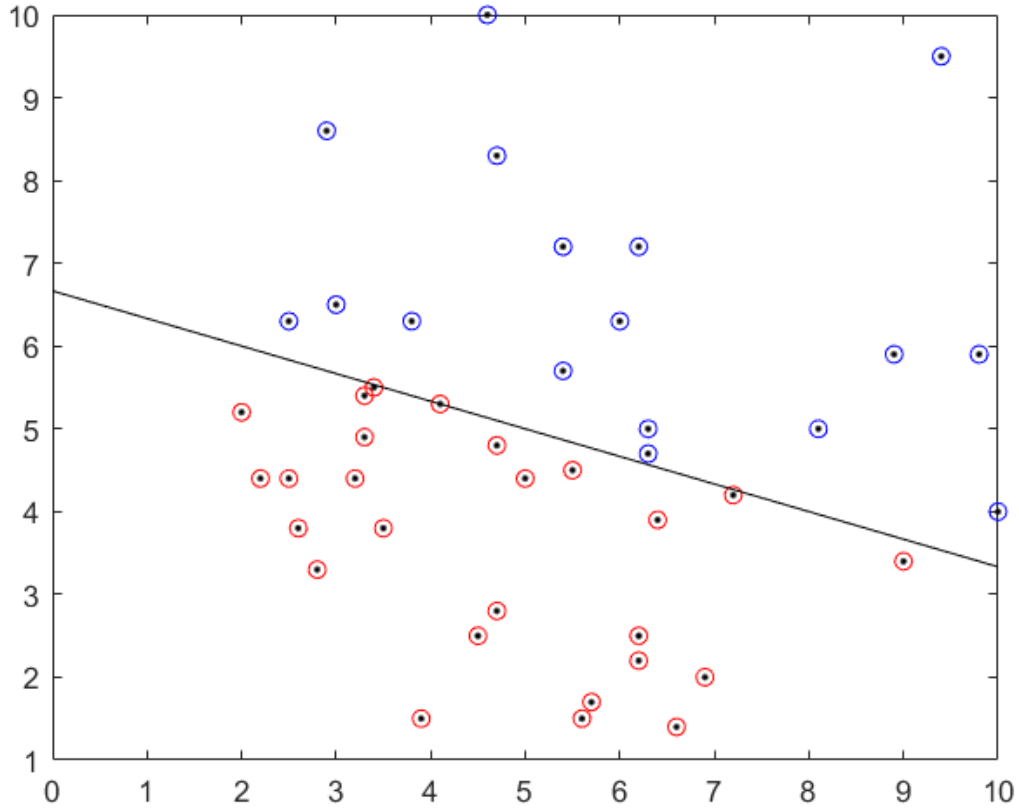
O aluno é aprovado se a média é maior a 5, mas ...

Qual é o peso de cada prova?

Descubra o peso de cada prova usando o perceptron.

Passou=1 Reprovou=0	N1	N2
0	5.70	1.70
1	3.80	6.30
1	4.60	10.0
0	6.60	1.40
1	5.40	7.20
1	9.80	5.90
0	4.70	4.80
1	8.10	5.0
0	4.10	5.30
0	5.0	4.40
0	6.40	3.90
0	2.80	3.30
0	2.50	4.40
1	2.90	8.60
0	2.0	5.20
1	6.20	7.20
1	2.50	6.30
0	4.50	2.50
1	3.0	6.50
1	5.40	5.70
0	5.50	4.50
1	6.0	6.30
	...	

Solução gráfica



Uma rede?