Universidade POSITIVO

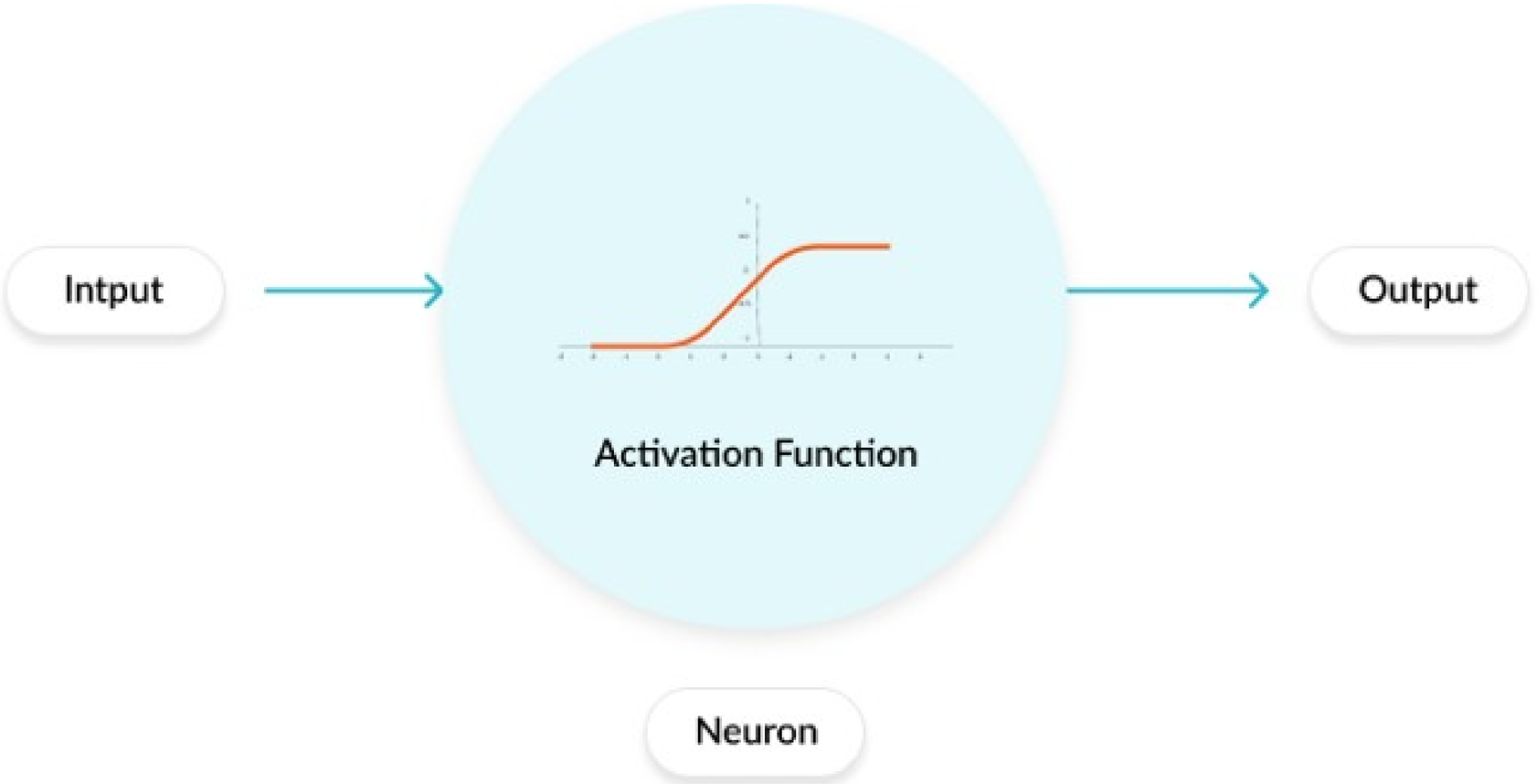
Sistemas Inteligentes
Redes Neurais

Semana 13 – Introdução a Redes Neurais e tipos de Aprendizado

Prof. Malga

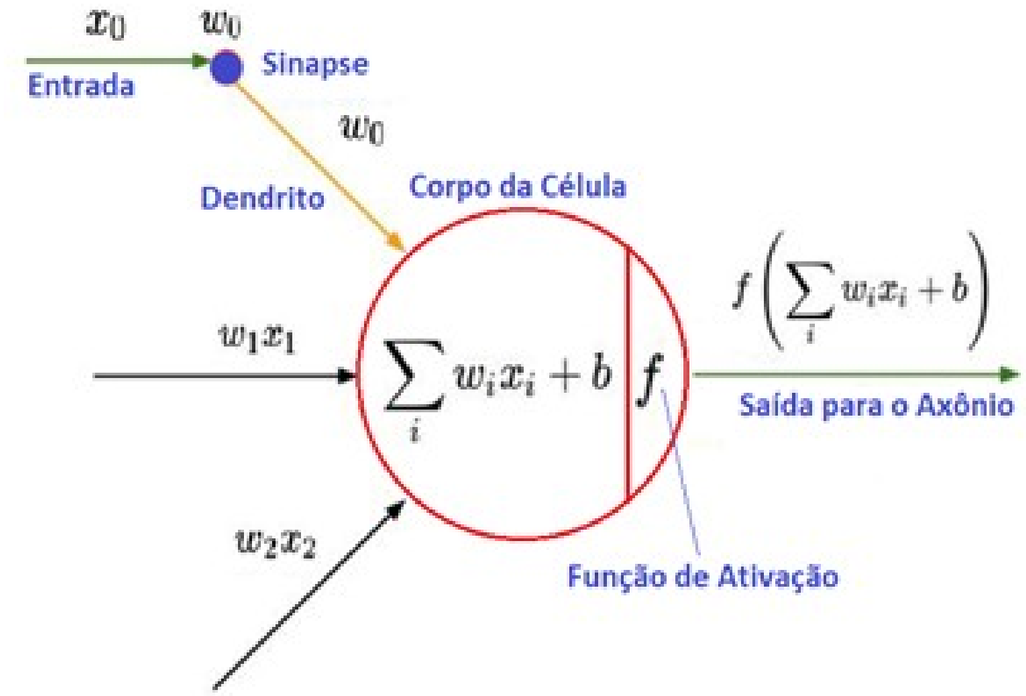
2º semestre/2025

Funções de Ativação



Funções de Ativação

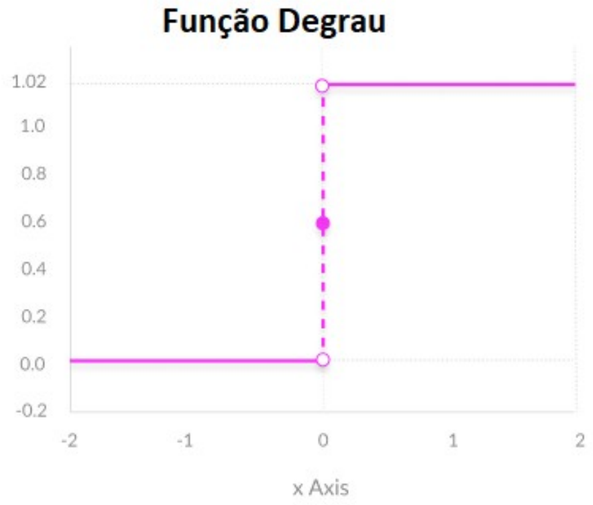
O que é a função de ativação?

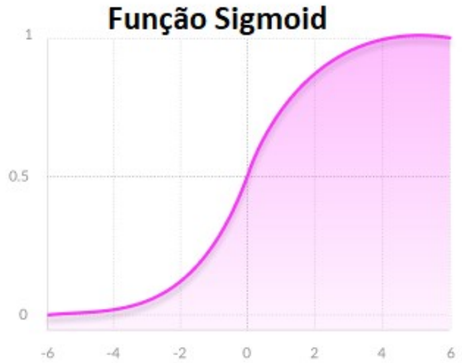


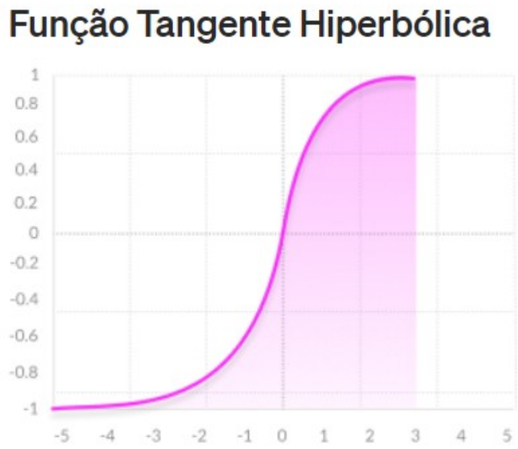
Uma **função de ativação**, nada mais é do que uma equação matemática que determina o output do seu neurônio, podendo ser encontrada no final de cada neurônio de uma rede neural. A principal função da função de ativação é *introduzir não-linearidade na saída* da unidade, o que **permite que a rede neural aprenda padrões mais complexos nos dados**.

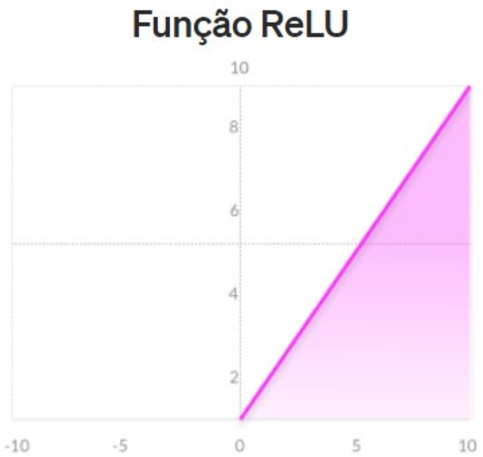
Funções de Ativação

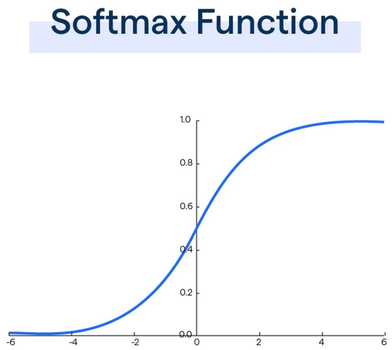
Funções de Ativação

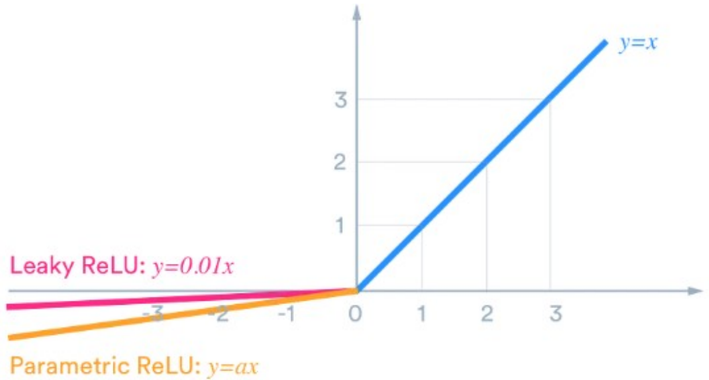








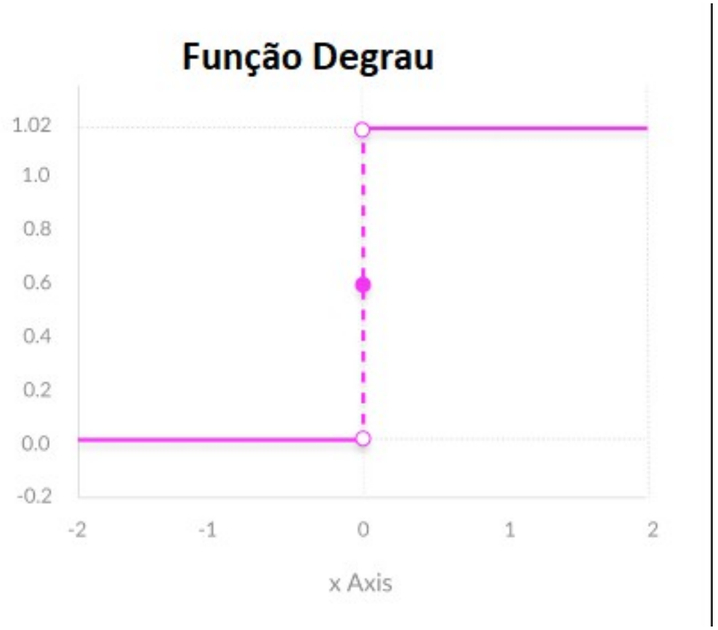




Funções de Ativação

Função Degrau – (A mais adequada para Perceptron simples)

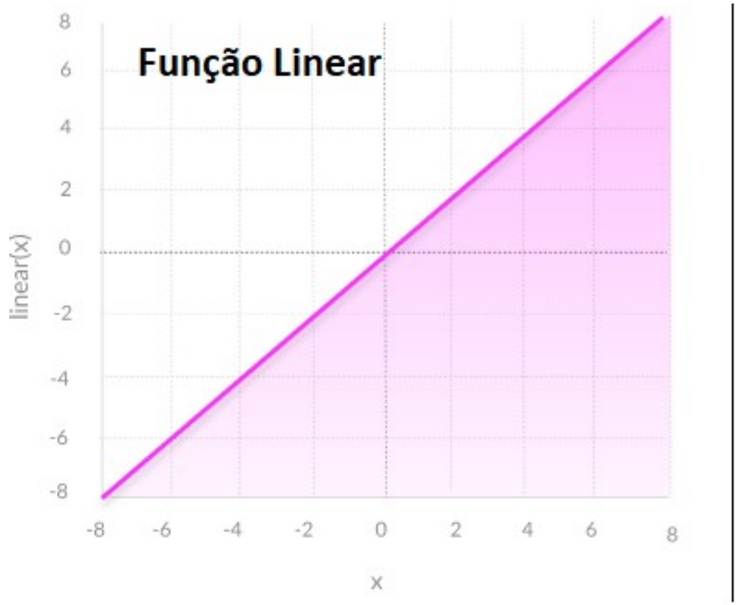
A função de degrau (etapa binária é extremamente simples. Ela pode ser usada ao criar um classificador binário. Quando simplesmente precisamos dizer sim ou não para uma única classe, a função seria a melhor escolha, pois ativaria o neurônio ou deixaria zero.



Funções de Ativação

Função Linear

Essa função recebe a entrada, que é multiplicada por cada peso do neurônio, e emite um output com o valor proporcional ao recebido. É melhor que a binária, pois permite a função emitir sinais com valores além do 0 e 1



Estudou	Conhecimento prévio	Facilidade de aprendizado	Nome	Resultado
0.8	0.5	0.7	Miguel	1
0.3	0.2	0.8	Bruno	0

Funções de Ativação

Sigmoide

A função Sigmoide é contínua, tendo seus valores variando **entre 0 e 1**.

A função de ativação Sigmoide é apropriada para resolução de problemas que envolvem valores probabilísticos como saída.

Graficamente

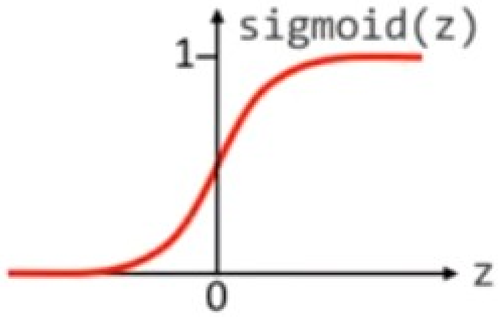
$$f(x) = \frac{1}{1 + e^{-x}}$$

Equação

Funções de Ativação

Softmax

Quando o problema tem mais de uma classe, é aplicado uma Função de Ativação que é uma Generalização da Sigmoide que é a *Softmax*.



$$f(x) = \frac{1}{1 + e^{-x}}$$

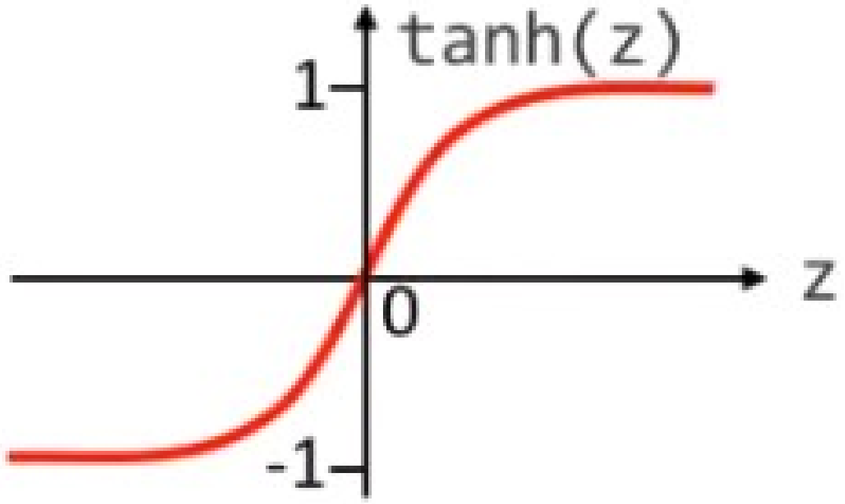
$$Softmax(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}}$$

Funções de Ativação

Funções de Ativação – PARA CAMADAS OCULTAS

Tangente Hiperbólico

Semelhante a sigmoide, porém tem uma variação de -1 até 1 .

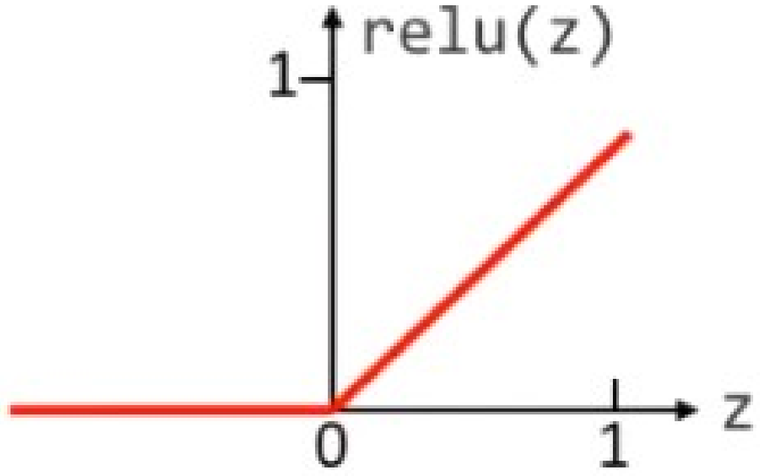


$$f(x) = \frac{\left(e^{x} - e^{-x}\right)}{\left(e^{x} + e^{-x}\right)}$$

Funções de Ativação

Unidade Linear Retificada - ReLU

A vantagem da Relu é que a mesma anula valores negativos tendo como retorno da derivada sempre o valor 1, ou seja, a inclinação é sempre constante.



$$g(z) = \max(0, z)$$

same as

$$g(z) = \begin{cases} 0, & z < 0 \\ z, & z \geq 0 \end{cases}$$

Funções de Ativação

Vantagens da ReLU:

A principal vantagem de usar a função ReLU sobre outras funções de ativação é que ***ela não ativa todos os neurônios ao mesmo tempo***.

O que isto significa ?

Se você olhar para a função ReLU e a entrada for negativa, ela será convertida em zero e o neurônio não será ativado. Isso significa que, ao mesmo tempo, apenas alguns neurônios são ativados, tornando a rede esparsa e eficiente e fácil para a computação

- ***Tipos de Aprendizado***

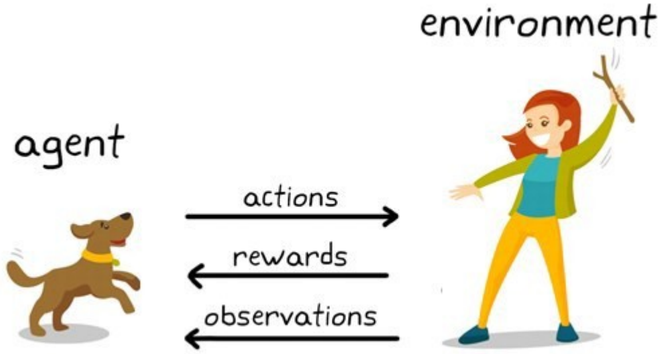
Após entendermos um pouco sobre as funções de ativação, precisamos entender como se dá o aprendizado de uma rede neural.

• *Tipos de Aprendizado*

Aprendizado Supervisionado

Aprendizado Não-Supervisionado

Aprendizado Por Reforço



- *Tipos de aprendizado*

Aprendizado Supervisionado

O aprendizado supervisionado em redes neurais é uma abordagem em que o modelo é treinado usando um conjunto de dados rotulados. Isso significa que o conjunto de dados de treinamento inclui pares de entradas e suas correspondentes saídas desejadas (rótulos). O objetivo do modelo é aprender uma relação entre as entradas e as saídas de maneira a poder fazer especificidades específicas para novas entradas não vistas anteriormente

- *Tipos de aprendizado*

Aprendizado Supervisionado

Tecnicamente chamamos as entradas de:
ATRIBUTOS => Vetor de atributos.

As saídas de:
Classes.

• *Tipos de Aprendizado*

Aprendizado Supervisionado

Por exemplo:

Vamos descrever essas frutas através de 4 atributos:
ATRIBUTOS = [COR, PESO, TEXTURA, PH]

- *Tipos de Aprendizado*

Aprendizado Supervisionado

Por exemplo:

Obs: OS dados referentes aos atributos peso, textura e pH são hipotéticos.

X(k)	Cor	Peso (g)	Textura	pH
X(1)	Vermelha	113	Lisa	6,8
X(2)	Laranja	122	Rugosa	4,7
X(3)	Vermelha	107	Lisa	5,2
X(4)	Vermelha	98	Lisa	3,6
X(5)	Laranja	115	Rugosa	2,9
X(6)	Laranja	120	Rugosa	4,2

Atributos

(entradas)

Y(k)	Fruta
Y(1)	Maçã
Y(2)	Laranja
Y(3)	Maçã
Y(4)	Maçã
Y(5)	Laranja
Y(6)	Laranja

Classificação

(saída)

Notem que apenas os atributos de cor e textura para este exemplo seriam suficiente (problema de dimensionalidade).

Menos atributos significam redes neurais mais simples e eficientes E menor tempo de treinamento.

• *Tipos de Aprendizado*

Aprendizado Supervisionado

Por exemplo:

Obs: OS dados referentes aos atributos peso, textura e pH são hipotéticos.

X(k)	Cor	Peso (g)	Textura	pH
X(1)	Vermelha	113	Lisa	6,8
X(2)	Laranja	122	Rugosa	4,7
X(3)	Vermelha	107	Lisa	5,2
X(4)	Vermelha	98	Lisa	3,6
X(5)	Laranja	115	Rugosa	2,9
X(6)	Laranja	120	Rugosa	4,2

Y(k)	Fruta
Y(1)	Maçã
Y(2)	Laranja
Y(3)	Maçã
Y(4)	Maçã
Y(5)	Laranja
Y(6)	Laranja

Notação matemática:

$\underline{x}(3)$ = [Vermelha, 107, Lisa, 5,2] ⟹ y(3) = Maçã

vetor de atributos ou *feature vector*

Classe

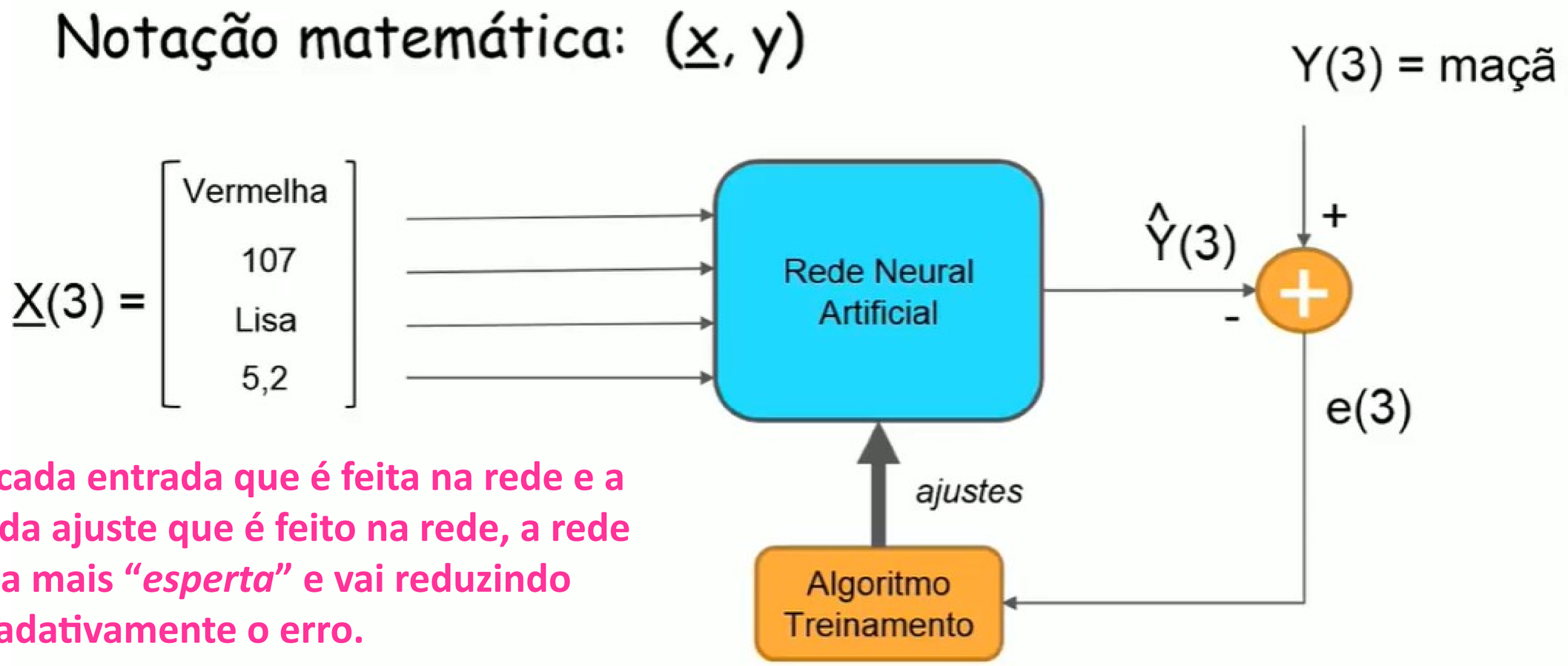
Saída desejada

Alvo (*target*)

• *Tipos de Aprendizado*

Aprendizado Supervisionado

Por exemplo:



A cada entrada que é feita na rede e a cada ajuste que é feito na rede, a rede fica mais "*esperta*" e vai reduzindo gradativamente o erro.

• *Tipos de Aprendizado*

Aprendizado Não-Supervisionado

O aprendizado não supervisionado em redes neurais refere-se a um tipo de treinamento de rede em que o modelo é alimentado com dados não rotulados, e o objetivo principal é descobrir padrões intrínsecos ou estruturas nos dados sem orientação externa sobre o que procurar. Em outras palavras, a rede é deixada para explorar e aprender as características básicas dos dados por conta própria.

- *Tipos de Aprendizado*

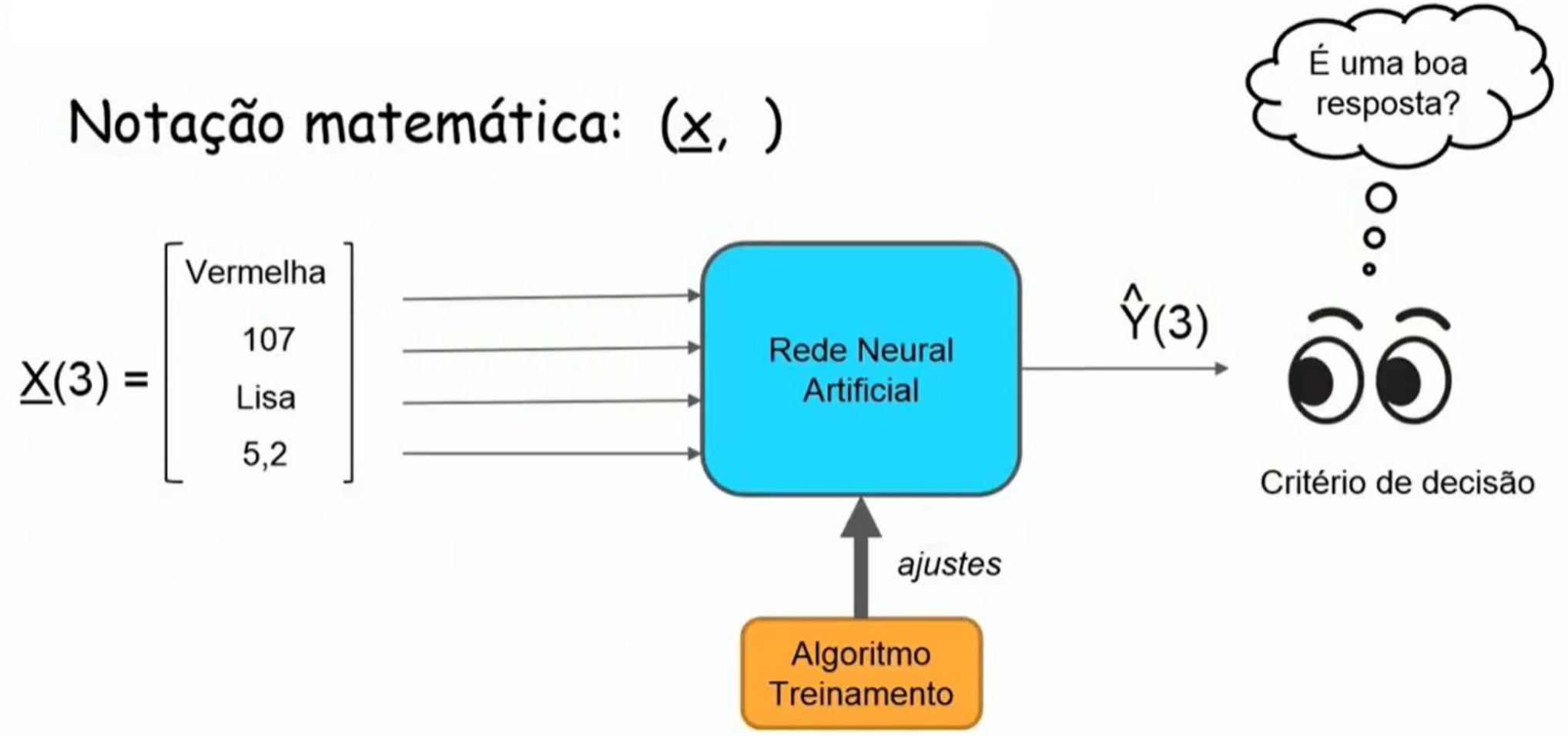
Aprendizado Não-Supervisionado

Agrupamento de Clientes em um Shopping

Suponha que você tenha dados de transações de clientes em uma compra, mas esses dados não têm rótulos indicados na categoria de cada cliente. Você deseja segmentar os clientes em grupos diferentes com base em seus padrões de compra para poder personalizar estratégias de marketing para cada grupo.

- *Tipos de Aprendizado*

Aprendizado Não-Supervisionado



• *Tipos de Aprendizado*

Aprendizado Por Reforço

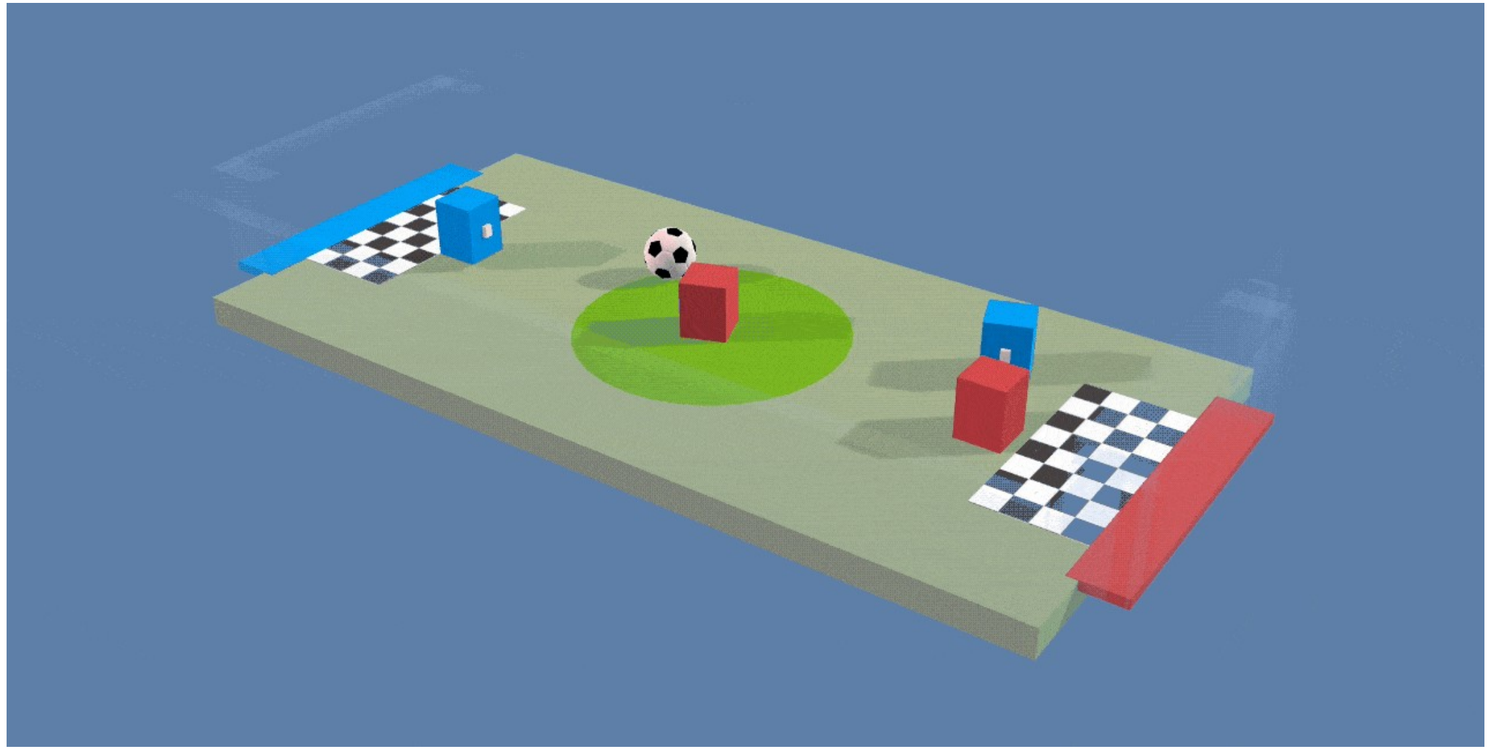
O aprendizado por reforço (RL) é um paradigma de aprendizado de máquina em que um agente interage com um ambiente dinâmico para aprender a realizar ações que maximizam uma noção de recompensa cumulativa. Esse tipo de aprendizado é inspirado na psicologia comportamental, onde um agente é incentivado ou desencorajado com base nas consequências de suas ações.

Em contextos de redes neurais, o aprendizado por reforço pode ser aplicado usando arquiteturas específicas, como Redes Neurais Profundas (Deep Neural Networks), para representar políticas do agente ou para estimar funções de valor.

• *Tipos de Aprendizado*

Aprendizado Por Reforço

É uma técnica que tem como objetivo treinar um **Agente** (programa, código, algoritmo, etc.) a interagir em um **Ambiente** por meio de **Ações** para atingir um **Objetivo**. Por meio de **Recompensas ou punições** dadas a esse agente, ele irá **aprender quais ações deve executar para aumentar a recompensa e atingir o objetivo.**



- *Qual das três Abordagens é a Melhor?*

A resposta a essa pergunta depende do que você está analisando. Cada problema possui suas peculiaridades.

Hierarquia do Aprendizado

Conforme visto em estudos anteriores, o aprendizado de máquinas **obedece uma hierarquia segundo os tipos de tarefas de aprendizado**.

Alguns algoritmos utilizam tarefas que induzem ao modelo preditivo, ou seja, seguem o paradigma do aprendizado supervisionado.

As tarefas supervisionadas se distinguem pelo tipo de rótulos dos dados, discreta no caso de classificação e contínuos no caso de regressão Linear.

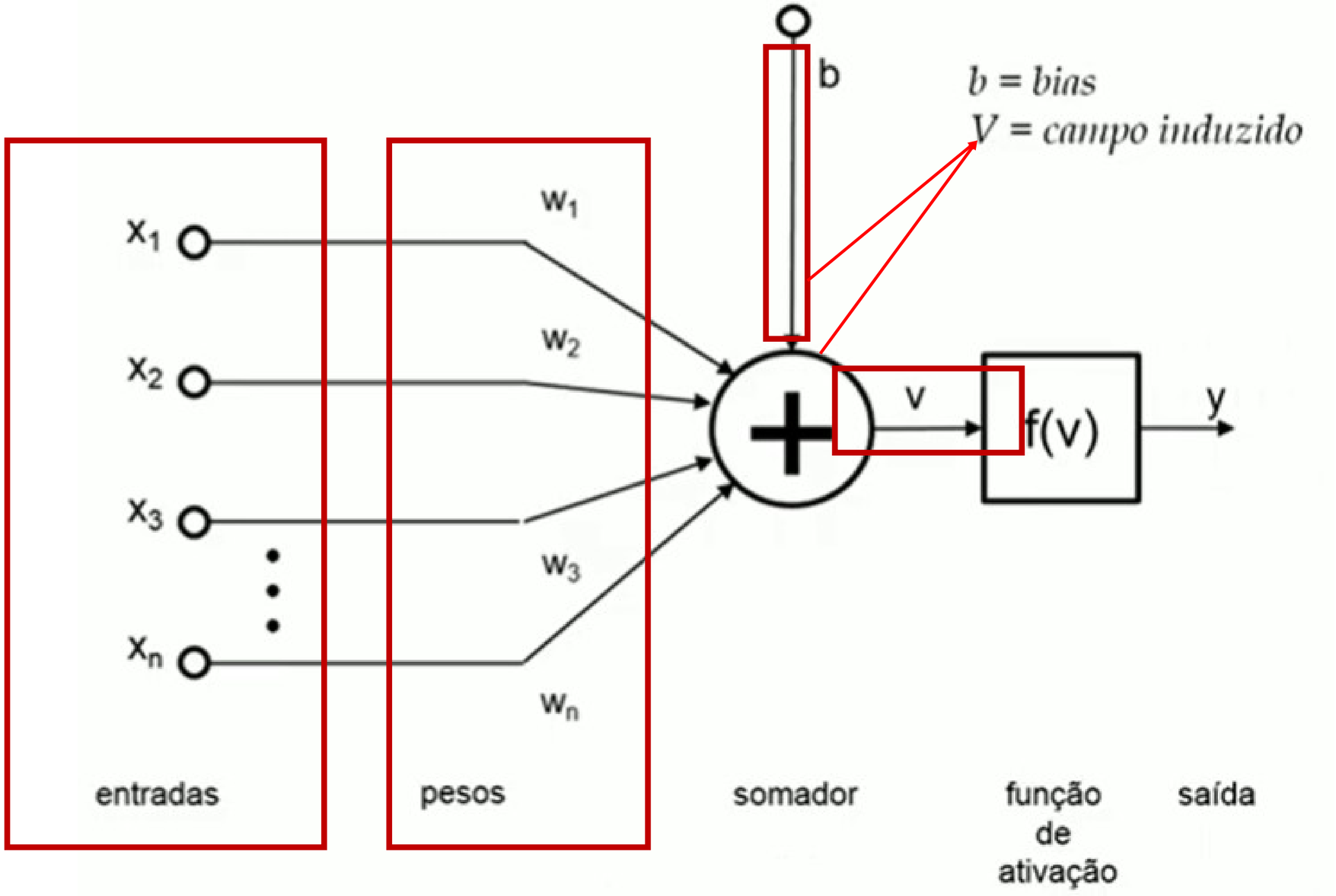
Hierarquia do Aprendizado

Já alguns algoritmos de aprendizado, utilizam tarefas de descrição, não utilizando-se de atributos de saída, sendo assim, seguem o paradigma de aprendizado não supervisionado.

Uma tarefa descritiva de agrupamento de dados por exemplo, tem por meta encontrar grupos de objetos semelhantes enquanto que as regras de associação um grupo de atributo a outro grupo de atributos.

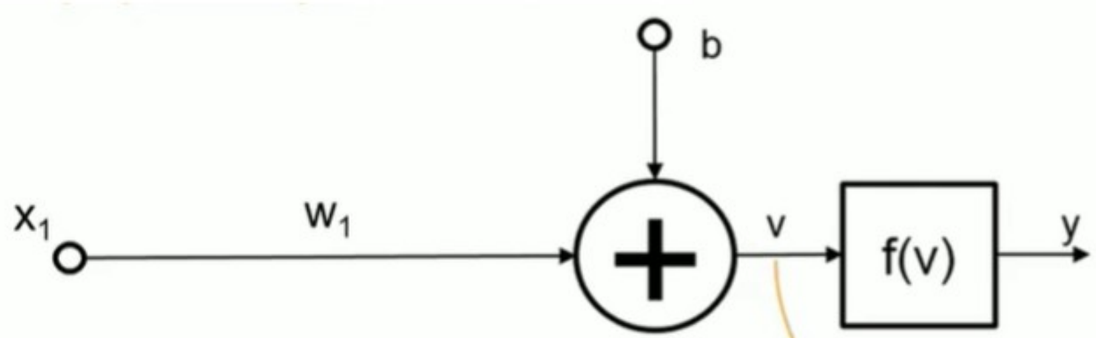
Hierarquia do Aprendizado

O Papel do Bias e o Modelo Matemático de um Neurônio



Papel do Bias

O "bias" **é um elemento que serve para aumentar o grau de liberdade dos ajustes dos pesos**.



Para que serve o coeficiente linear?
Responsável por posicionar na reta na origem dos eixos do gráfico, acima ou abaixo.

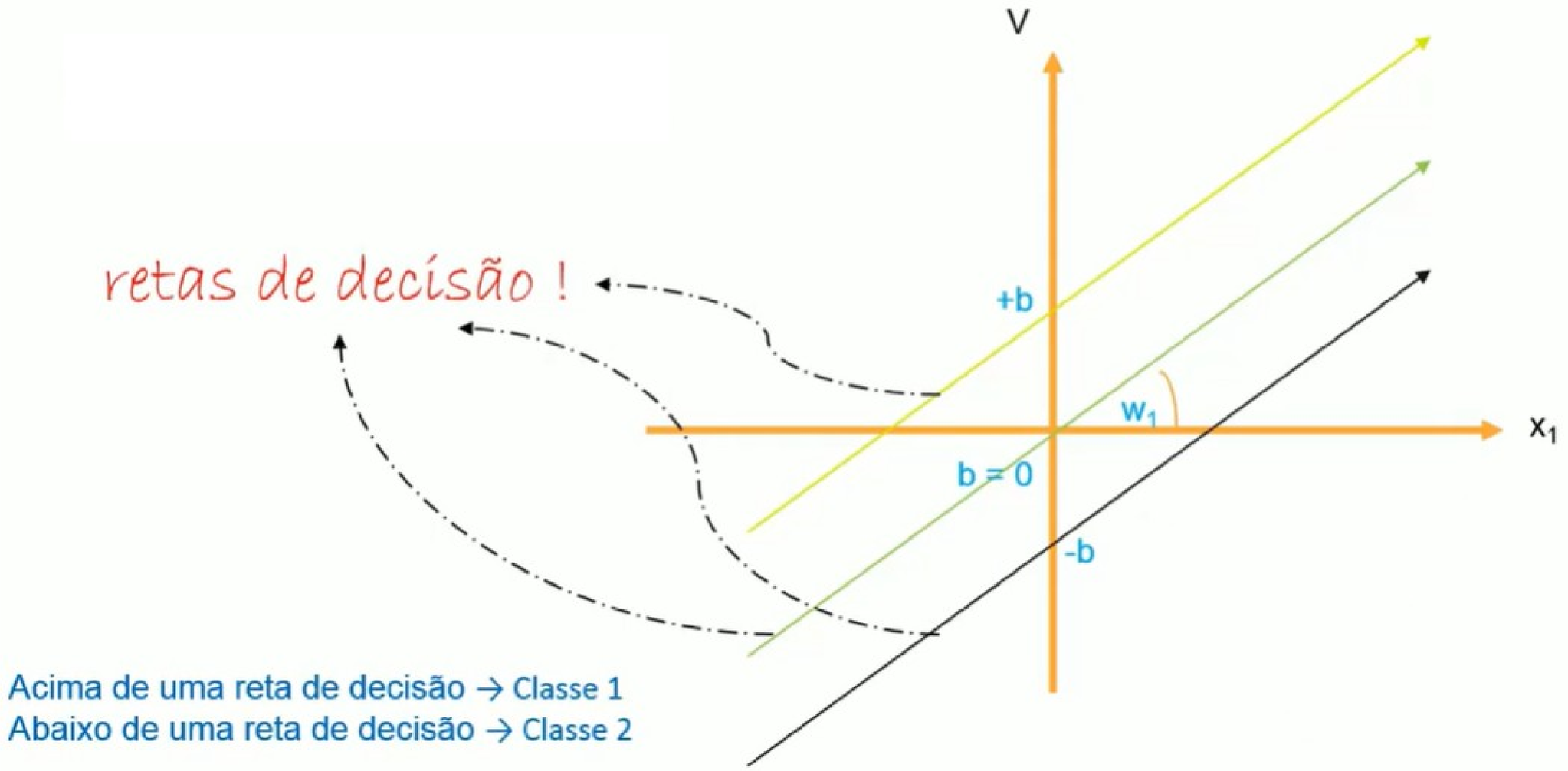
$$v = w_1 x_1 + b$$ (equação da reta)

Coeficiente linear

Coeficiente angular

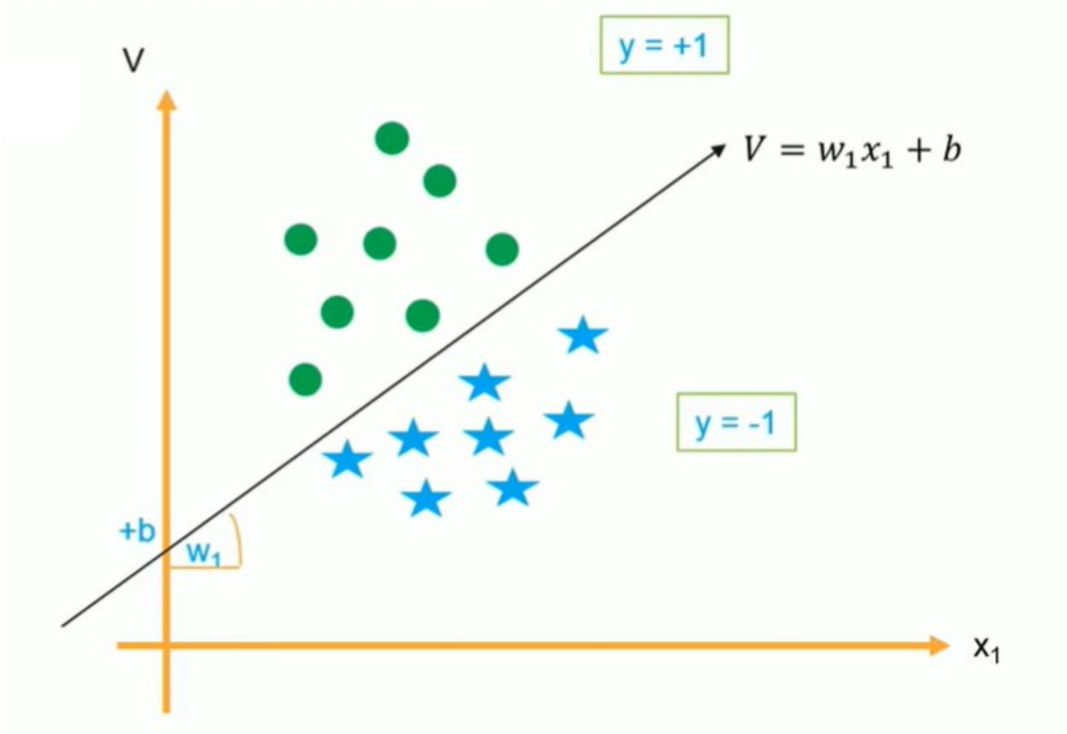
Para que serve o coeficiente angular?
Serve para girar a reta inteira no sentido horário ou anti-horário

Papel do Bias



Papel do Bias

Vamos imaginar que queremos classificar um objeto *unidimensional*, ou seja, **possui uma única classe**.



Aqui vamos usar pontos verdes e estrelinhas azuis como objetos meramente ilustrativos.

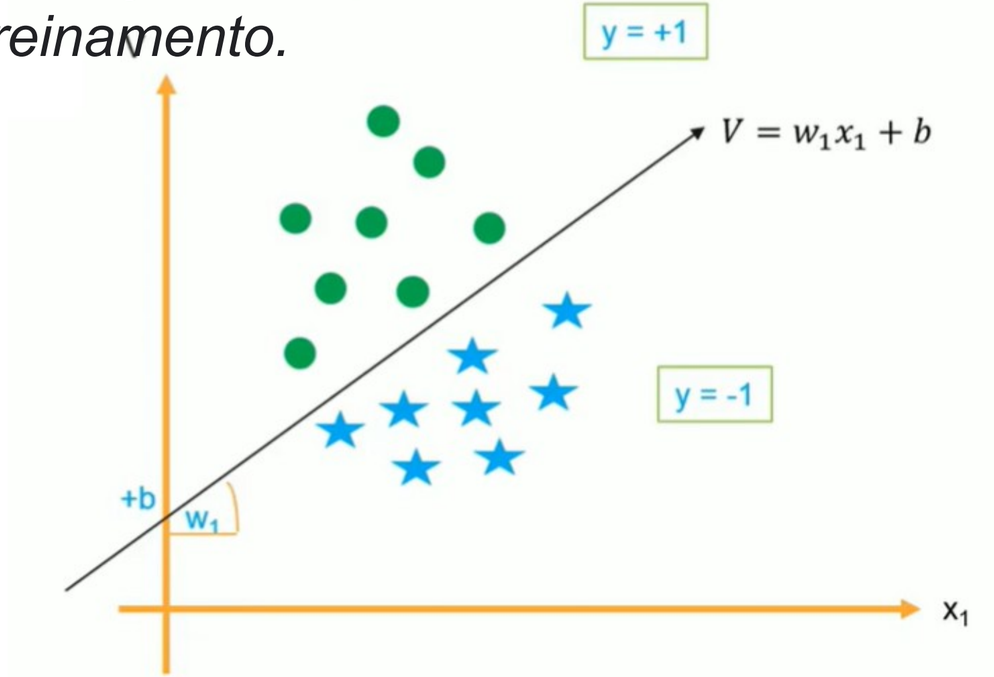
E para classifica-los podemos utilizar qualquer grandeza (temperatura, peso, comprimento). E vamos chamar essa grandeza de x_1.

Papel do Bias

Após medir o **atributo x_1** de todos os objetos a serem classificados, ***entregamos*** os resultados para nosso ***neurônio Perceptron com uma única entrada.***

A cada dado que fornecemos comparamos a resposta dada pelo neurônio com a resposta **verdadeira**.
Se o algoritmo ***acertou*** *nada acontecerá em* ***termos de treinamento****.*

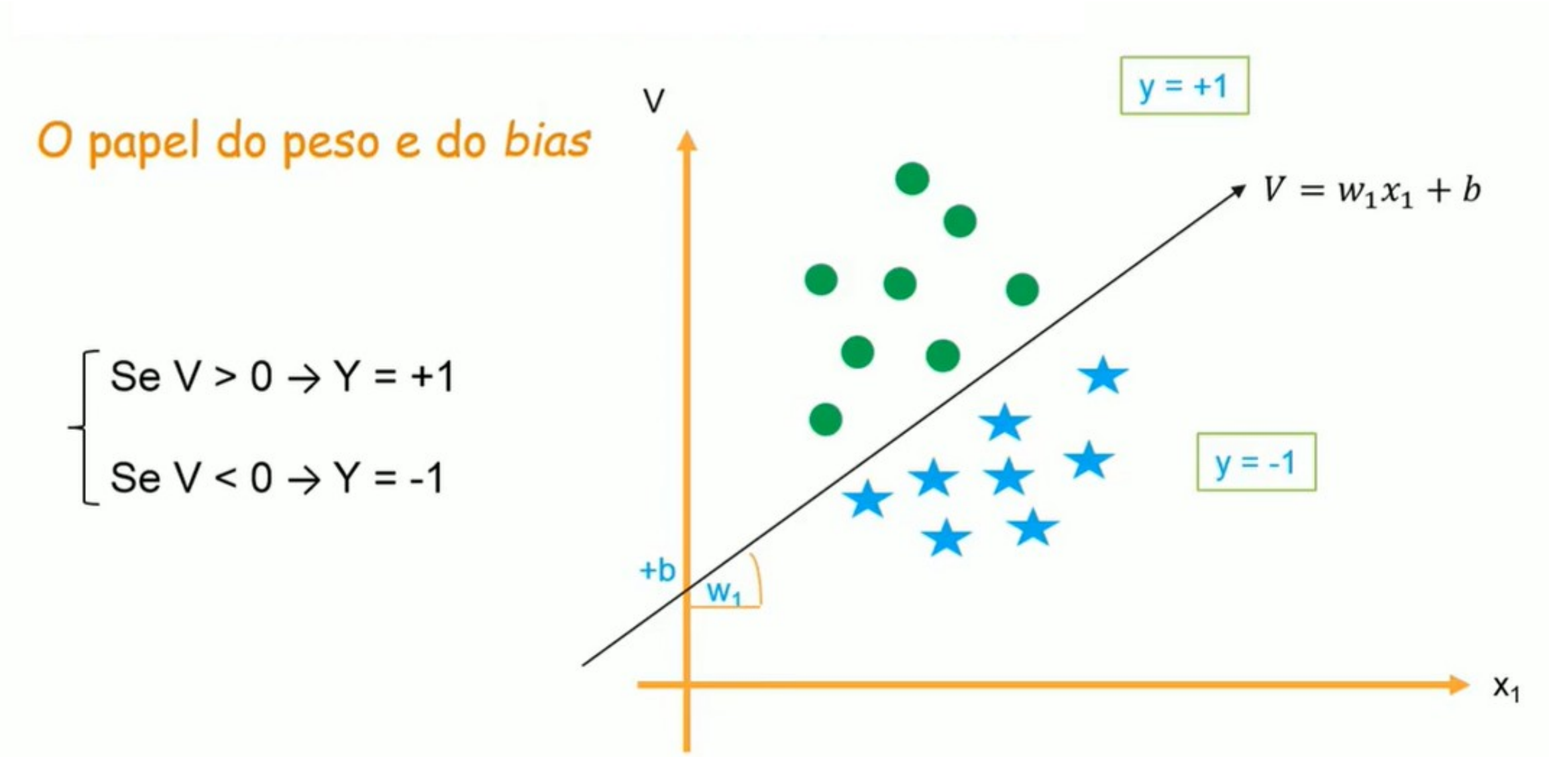
Se o algoritmo **errou** *o algoritmo de treinamento* ***ira ajustar o peso do bias e das sinapses*** *de modo a diminuir o erro na próxima etapa de treinamento.*



Papel do Bias

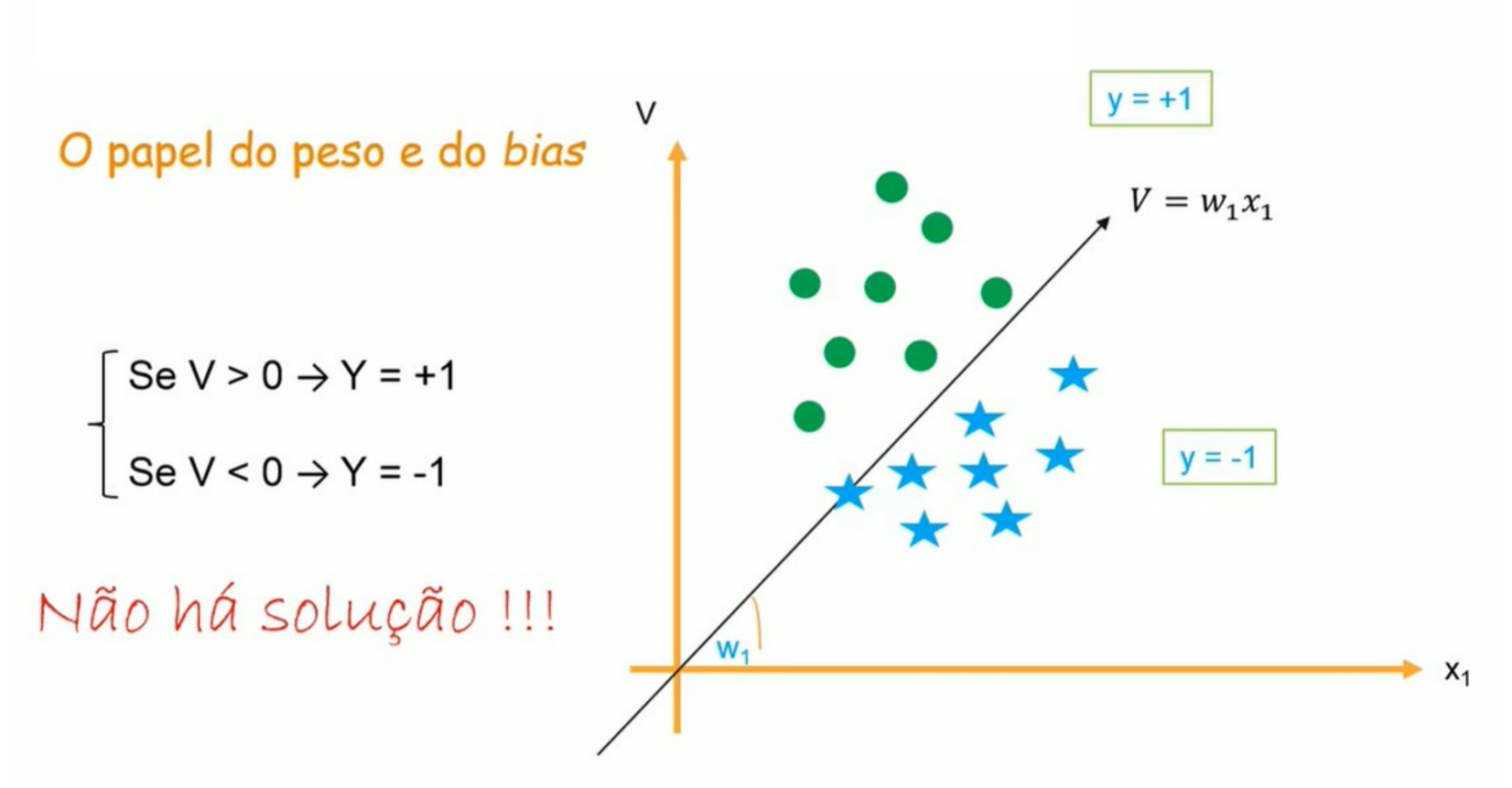
O que isso significa?

Significa que o algoritmo de treinamento irá girar a reta no sentido horário ou anti-horário ***E*** subir ou descer mais a reta de modo que ao final do treinamento uma reta seja encontrada de modo a separar todos os objetos corretamente.



Papel do Bias

Se neste mesmo exemplo caso o bias fosse nulo!



Tecnicamente então podemos dizer que o Bias aumenta o grau de liberdade do nosso sistema.

Papel do Bias

O papel dos pesos e do *bias*

E se a rede tiver duas entradas? x_1 e x_2?

Então: $v = w_1x_1 + w_2x_2 + b$ (equação de um plano)

E se a rede tiver mais do que duas entradas? x_1 , x_2, ... , x_n?

Então: $v = w_1x_1 + w_2x_2 + \cdots + w_nx_n + b$ (equação de um hiperplano)

Papel do Bias

Exercício

Seja a entrada de um neurônio dado pelo vetor $\underline{x}$ = [1,4 -2,5 0,7] e seja o vetor de pesos das sinapses $\underline{w}$ = [0,4 0,6 0,3]. Determine se a saída é positiva ou negativa nos casos:

a) Com *bias* nulo

b) Com *bias unitário*

Papel do Bias

Exercício

Lembrando que: $\underline{x}$ = [1,4 -2,5 0,7]
$\underline{w}$ = [0,4 0,6 0,3].

a) Com bias nulo (b = 0)

$$x_1w_1 + x_2w_2 + x_3w_3 + 0 = 1{,}4 * 0{,}4 - 2{,}5 * 0{,}6 + 0{,}7 * 0{,}3 = -0{,}73$$

colocando em forma vetorial a mesma conta, ficaria assim:

$$\underbrace{[1{,}4 \quad -2{,}5 \quad 0{,}7]}_{\underline{x}} \underbrace{\begin{bmatrix} 0{,}4 \\ 0{,}6 \\ 0{,}3 \end{bmatrix}}_{\underline{w}^T} = -0{,}73 \Rightarrow v = -0{,}73 \therefore f(v) = -1$$

Nosso campo induzido (V) retornou um valor negativo, significa que nosso neurônio não disparou.

Papel do Bias

Exercício

Lembrando que: $\underline{x}$ = [1,4 -2,5 0,7]
$\underline{w}$ = [0,4 0,6 0,3].

b) Com bias unitário (b = 1)

$$x_1w_1 + x_2w_2 + x_3w_3 + 1 = 1{,}4 * 0{,}4 - 2{,}5 * 0{,}6 + 0{,}7 * 0{,}3 + 1 = 0{,}27$$

colocando em forma vetorial a mesma conta, ficaria assim:

$$\underbrace{[1{,}4 \quad -2{,}5 \quad 0{,}7]}_{\underline{x}} \underbrace{\begin{bmatrix} 0{,}4 \\ 0{,}6 \\ 0{,}3 \end{bmatrix}}_{\underline{w}^T} + 1 = 0{,}27 \Rightarrow v = 0{,}27 \therefore f(v) = +1$$

Nosso campo induzido (V) retornou um valor positivo, e isso significa que nosso neurônio disparou.

Papel do Bias

Truque matemático!

Transformar o bias em uma entrada comum da rede.

$x_0 = +1$ (sempre !!!)

$$[1{,}4 \quad -2{,}5 \quad 0{,}7] \begin{bmatrix} 0{,}4 \\ 0{,}6 \\ 0{,}3 \end{bmatrix} + 1 \equiv [+1 \quad 1{,}4 \quad -2{,}5 \quad 0{,}7] \begin{bmatrix} 1 \\ 0{,}4 \\ 0{,}6 \\ 0{,}3 \end{bmatrix} = 0{,}27$$

Porque usar esse truque?

Para melhorar a eficiência dos pacotes Numpy do Python (pacote que trabalha com vetores de matrizes.

Papel do Bias

Ao invés de trabalharmos com loop´s para implementar o somatório do neurônio:

```
# Calcula o vetor campo induzido.
for i in range(n):
    V = V + W[i]*Xb[i]
V = V + bias
```

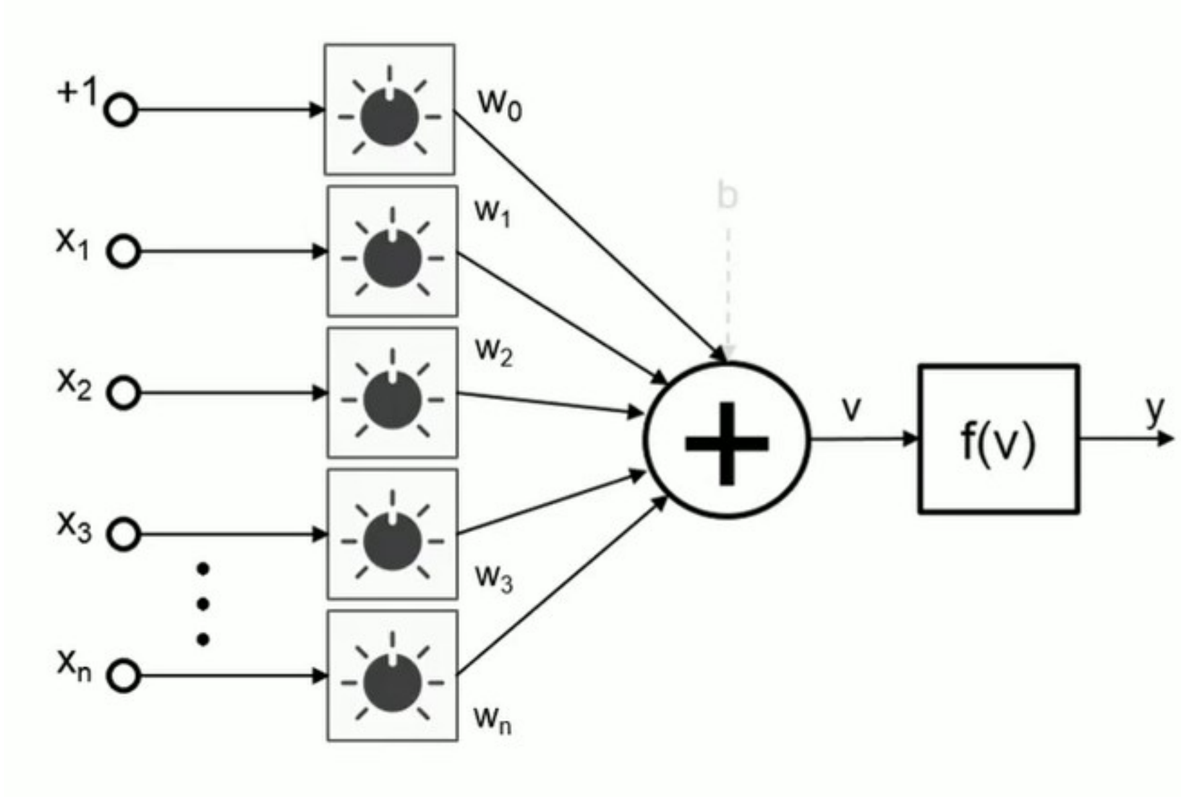
Fizemos uma multiplicação de vetores que é bem mais eficiente

```
# Calcula o vetor campo induzido.
V = numpy.dot(W, Xb)
```

Papel do Bias

Como essa alteração afeta nosso modelo?

Matematicamente em nada.



$$v = \sum_{i=1}^{n} x_n w_n + b \qquad (3)$$

$$v = \underline{x}\,\underline{w}^T \qquad (4)$$

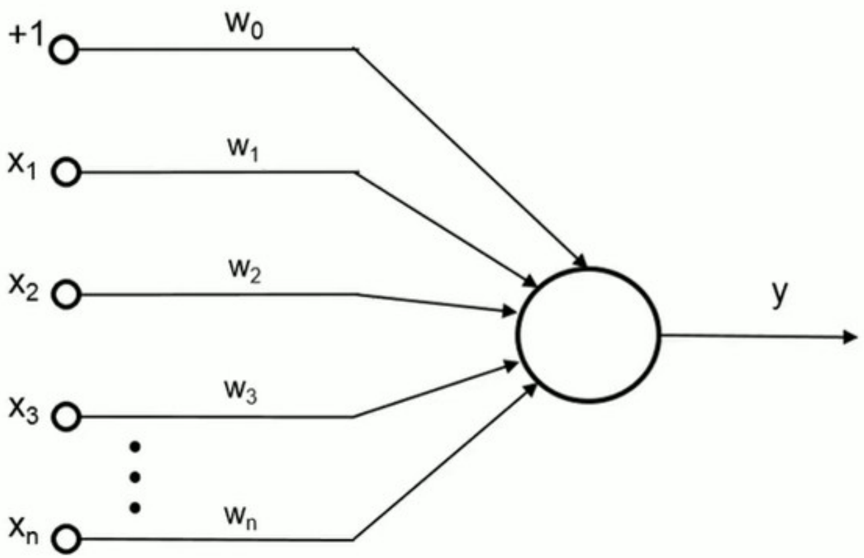
onde

$$\underline{x}_b = [+1 \quad x_1 \quad x_2 \ldots x_n]$$

$$\underline{w} = [w_0 \quad w_1 \quad w_2 \ldots w_n]$$

Papel do Bias

Nosso modelo Perceptron terá a seguinte forma:



$$v = \underline{x}\,\underline{w}^T \qquad (5)$$

$$y = f(v) \qquad (6)$$

onde

$$\underline{x}_b = [+1 \quad x_1 \quad x_2 \ldots x_n]$$

$$\underline{w} = [w_0 \quad w_1 \quad w_2 \ldots w_n]$$

Utilizaremos as questões 5 e 6 para implementação

Coração do algoritmo de aprendizado

Maneira como os pesos são ajustados durante fase de treinamento:

$$w(k+1) = w(k) + \eta[\hat{y}(k) - y(k)]x_b(k)$$

W(k) – vetor de pesos atual;
W (K + 1) – novo valor do vetor de pesos
η = Taxa de aprendizado (letra grega Éta)
$\hat{y}$ = Taxa de erro que multiplica (a saída da rede pela a saída desejada)
X_b (k) = A entrada que foi aplicada.

A taxa de aprendizado, nós que definimos e deve estra entre 0 e 1.

- **Se adotarmos valores muito pequenos, teremos um aprendizado muito lento.**
- **Se adotarmos valores muito grande , teremos um aprendizado mais rápido, mas corremos o risco do erro nunca ser zero.**

Treinamento do Perceptron de Camada Simples

1 – Inicialização

a. Faça w = [0]

2 – Ativação e treinamento

a. para k =1 a **q** faça //q numero de dados que você dispõe para treinar a rede

I – apresente o vetor de entradas $\mathbf{x}_b(k)$ ao perceptron

II – Calcule a saída do neurônio

III – Calcule o erro

IV – Faça W(k+1)

3 – Repita o passo 2 até e(k) = 0 para todo x(k). //x(k) todos os dados da base

Onde:

Eta = taxa de aprendizado (0 < eta < 1)

Yr(k) = +1 se x(k) pertence a Classe 1 e Yr(k) = -1 se x pertence a Classe 2

Perceptron de Camada Simples

Atributos = [peso, ph]

Perceptron de Camada Simples

Obs: OS dados referentes aos atributos peso e pH são hipotéticos.

q = 6

X(k)	Peso (g)	pH
X(1)	113	6,8
X(2)	122	4,7
X(3)	107	5,2
X(4)	98	3,6
X(5)	115	2,9
X(6)	120	4,2

Atributos

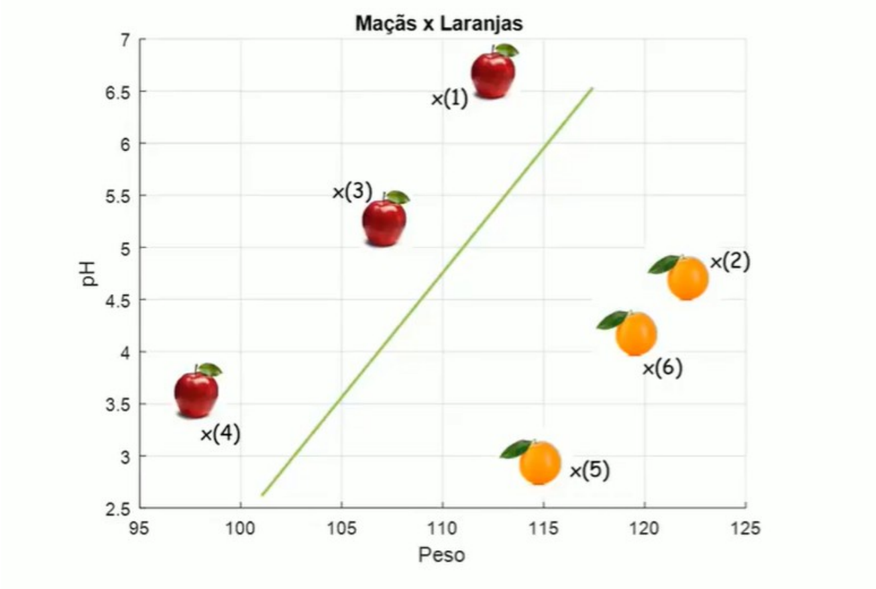
(entradas)

Y(k)	Fruta
Y(1)	Maçã
Y(2)	Laranja
Y(3)	Maçã
Y(4)	Maçã
Y(5)	Laranja
Y(6)	Laranja

Classificação

(saída)

Perceptron de Camada Simples



Preparação dos dados

$$\underline{X} = \begin{bmatrix} 113 & 122 & 107 & 98 & 115 & 120 \\ 6.8 & 4.7 & 5.2 & 3.6 & 2.9 & 4.2 \end{bmatrix}$$

peso

pH

$$\underline{y} = [-1 \quad +1 \quad -1 \quad -1 \quad +1 \quad +1]$$

Maçã = -1

Laranja = +1

Perceptron de Camada Simples

No Python teremos algo assim:

$$\underline{X} = \begin{bmatrix} 113 & 122 & 107 & 98 & 115 & 120 \\ 6.8 & 4.7 & 5.2 & 3.6 & 2.9 & 4.2 \end{bmatrix}$$

Matriz correspondendo ao peso (linha 1) e ph (linha 2)

$$\underline{X_b} = [+1 \quad 113 \quad 6.8]$$

Entrada do nosso Neurônio Perceptron.

$$y = -1$$

Valor de saída (nossa classe)

Perceptron de Camada Simples

È gero um loop que irá percorrer da seguinte forma

$$\underline{X} = \begin{bmatrix} 113 & 122 & 107 & 98 & 115 & 120 \\ 6.8 & 4.7 & 5.2 & 3.6 & 2.9 & 4.2 \end{bmatrix}$$

$$\underline{X_b} = [+1 \quad 120 \quad 4.2]$$

$$y = +1$$

Ao apresentarmos o último vetor do conjunto de dados para o perceptron, dizemos que completamos um ciclo chamado de "época".

Implementação de Uma Rede Perceptron de 1 camada

Exercício

Você tem dados de 24 estudantes com três atributos:

- `horas`: horas de estudo/semana
- `freq`: frequência às aulas em %
- `projs`: nº de atividades/projetos entregues no semestre

Objetivo: treinar e **comparar** três variantes do Perceptron para classificar **aprovado (1)** vs **reprovado (-1)**:

Considere os seguintes dados

horas = [2,5,1,7,3,4,6,8,2,9, 10,11,12, 1, 3, 5, 6, 7, 4, 8, 9, 2, 10, 6]

freq = [60,85,55,90,70,75,80,95,50,98, 92, 88, 96,58,68,74,77,86,72,89,93,57, 91, 83]

projs = [0, 2,0, 3,1, 1, 2, 4,0, 4,   5, 4, 5,0, 1, 2, 3, 3, 1, 4, 4, 0, 5, 3]

Y = [-1, 1,-1, 1,-1, 1, 1, 1,-1, 1,   1, 1, 1,-1,-1, 1, 1, 1, 1, 1, 1, -1, 1, 1]