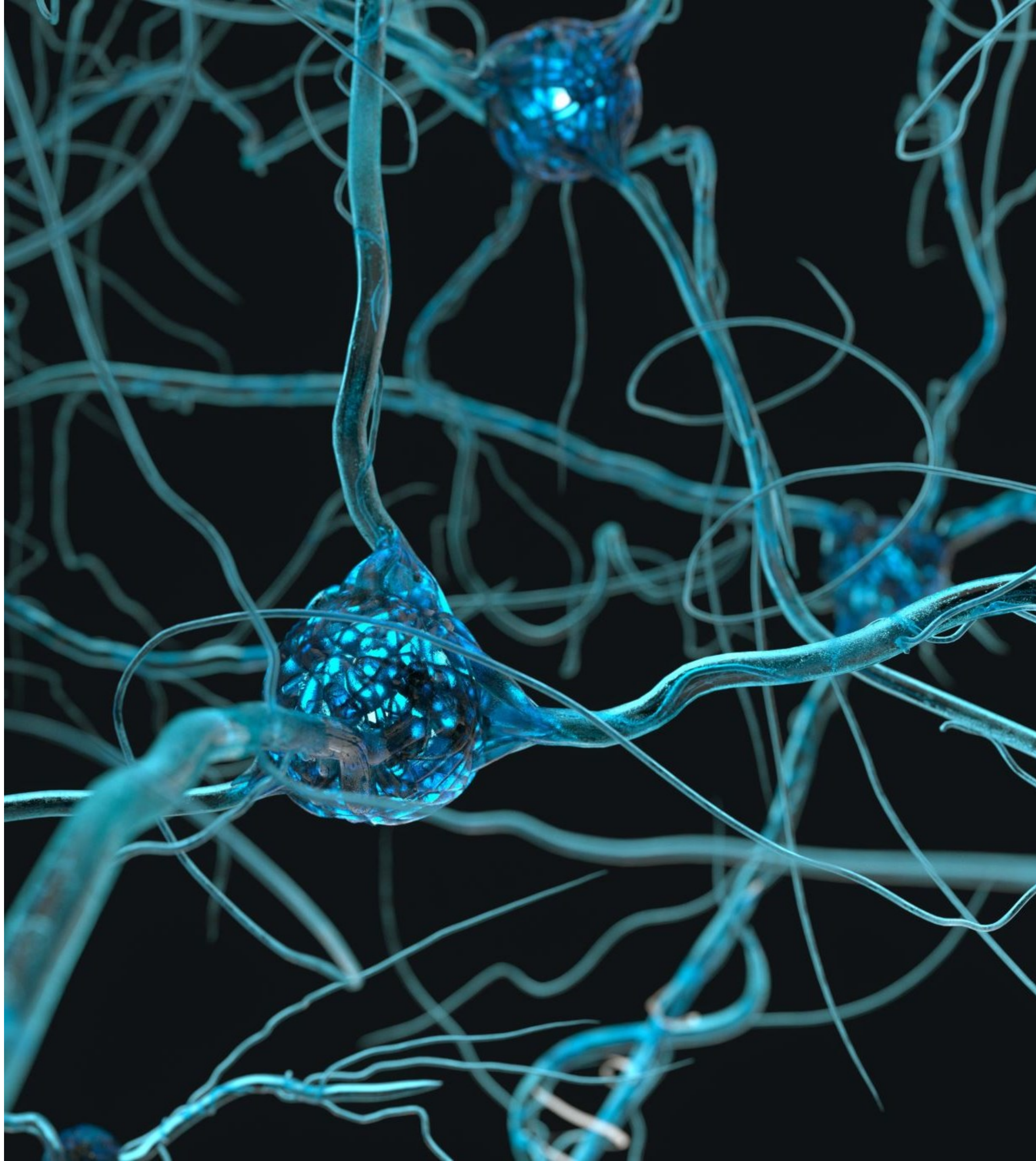
Rede Neural Perceptron

Silvia Moraes

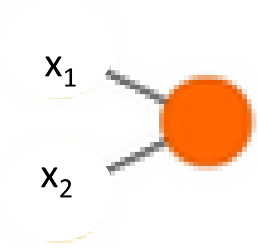
Roteiro

- Arquiteturas de Rede
- Redes FeedForward
- Revisitando a estrutura de um neurônio artificial
- Algoritmo de Aprendizado de uma rede perceptron
- Passo a passo de execução
- Limitação da rede

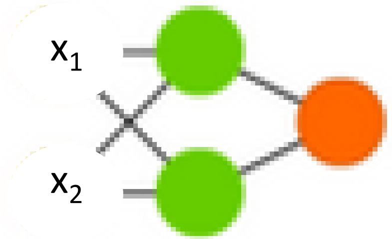


A arquitetura de uma rede neural define o tipo de tarefa que ela pode executar.

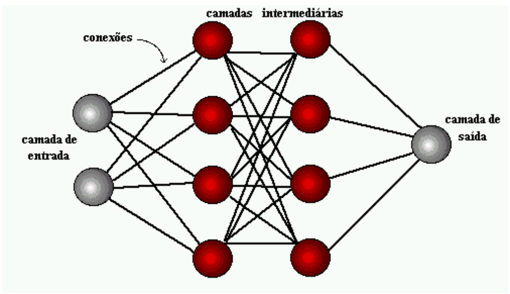
Redes Feed Forward
Rede alimentadas para Frente



Perceptron

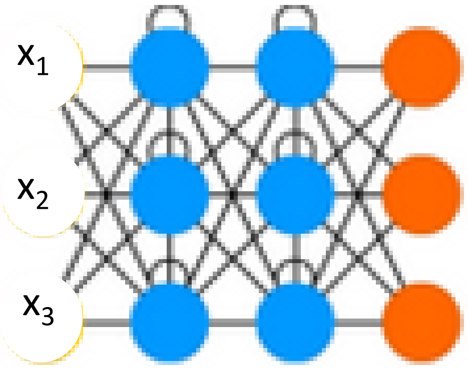


MultiLayer Perceptron

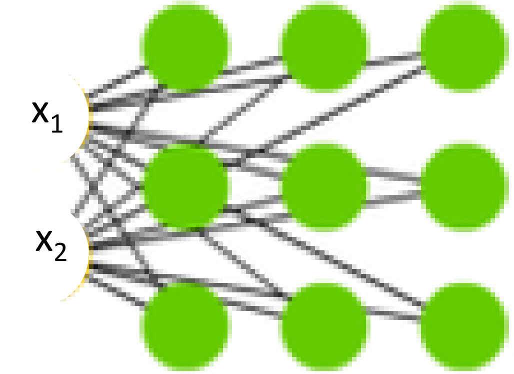


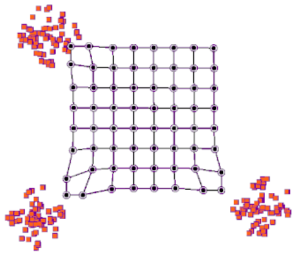
https://sites.icmc.usp.br/andre/research/neural/

Redes Recorrentes
Rede com retroalimentacao



Mapa auto-organizável
Redes de Kohonen

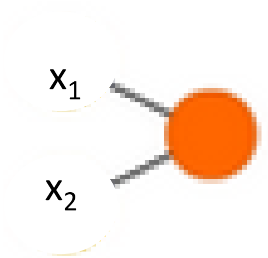




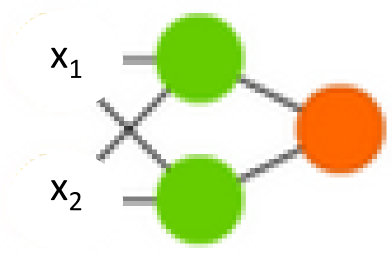
https://pt.wikipedia.org/wiki/Mapas_de_Kohonen

Redes Feed Forward

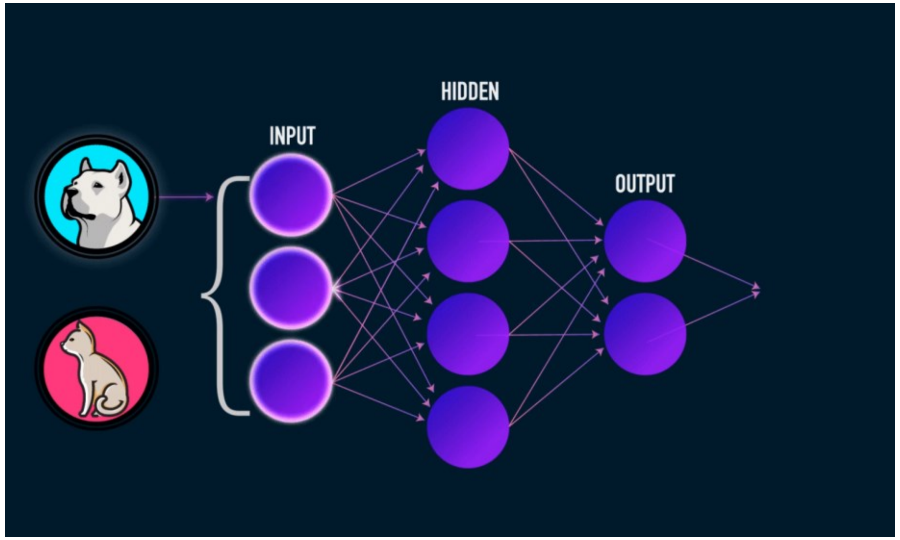
Tarefas Supervisionadas: **classificação o e regressão**



Perceptron

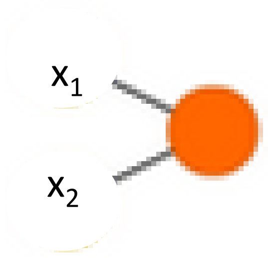


MultiLayer Perceptron



https://sites.icmc.usp.br/andre/research/neural/

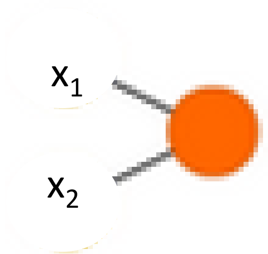
Propagação: fluxo do sinal sempre para frente



Redes Feed Forward

Rede Perceptron

- Perceptron é uma rede **muito simples**.
- Quando constituída de apenas um neurônio é chamada de perceptron elementar.
- Possui apenas **uma camada de neurônios**.
- Pode ter várias entradas e várias saídas.
- Trabalha com valores discretos e contínuos tanto para as entradas quanto para as saídas.
- Historicamente quando há valores contínuos, as redes com essas características eram chamadas de Adaline.
- <u>Limitação</u>: só consegue resolver problemas linearmente separáveis.

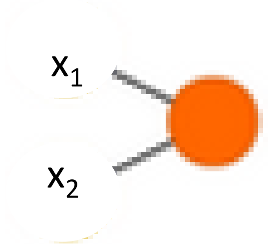


Redes Feed Forward

Rede Perceptron - Topologia

Como definir a quantidade valores de entrada, de camadas e neurônios da rede?

- Como esta rede tem apenas 1 camada, precisamos apenas definir a os **atributos de entrada** e **quantidade de neurônios** desta única camada.
- Normalmente, define-se um **neurônio para cada classe esperada**.
- Para entender, vamos ver um exemplo...



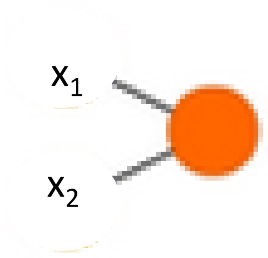
Redes Feed Forward

Rede Perceptron - Topologia

cpf	nome	renda	dívida	classificação do cliente
111...	João	2000	1000	bom
222...	Maria	3000	2000	mau
333...	Pedro	1000	500	mau
444...	Carlos	3000	1500	bom

Entradas: Quais são atributos podem ser usados como entrada para rede ?

Saídas : Em quantas classes do algoritmo deve organizar os clientes?



Redes Feed Forward

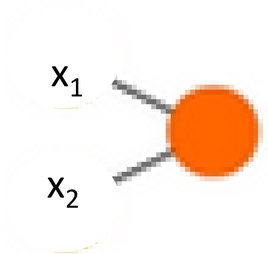
Rede Perceptron - Topologia

cpf	nome	renda	dívida	classificação do cliente
111...	João	2000	1000	bom
222...	Maria	3000	2000	mau
333...	Pedro	1000	500	mau
444...	Carlos	3000	1500	bom

2 atributos de entrada

Entradas: Quais são atributos podem ser usados como entrada para rede ?

Saídas : Em quantas classes do algoritmo deve organizar os clientes?



Redes Feed Forward

Rede Perceptron - Topologia

cpf	nome	renda	dívida	classificação do cliente
111...	João	2000	1000	bom
222...	Maria	3000	2000	mau
333...	Pedro	1000	500	mau
444...	Carlos	3000	1500	bom

2 classes como saida

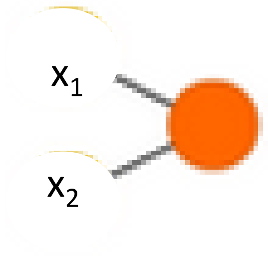
Entradas: Quais são atributos podem ser usados como entrada para rede ?

Saídas : Em quantas classes do algoritmo deve organizar os clientes?

Lembrando:

Redes neurais **trabalham somente com números**, por isso e comum realizar transformações nos dados.

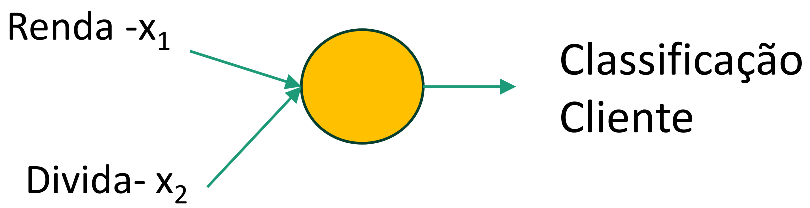
E funcionam melhor com dados normalizados.



Redes Feed Forward

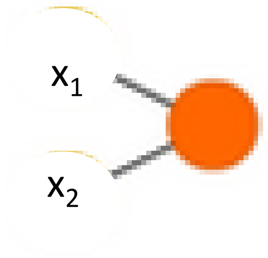
Rede Perceptron - Topologia

renda	dívida	classificação do cliente
0,66	0,5	1
1	1	0
0,33	0,25	0
1	0,75	1



Topologia menos usual.
Cada neurônio e capaz de decidir entre 2 classes:

- Quando a rede devolve 0, significa que não é um cliente bom.
- Quanto a rede devolve 1, significa que é um cliente bom.

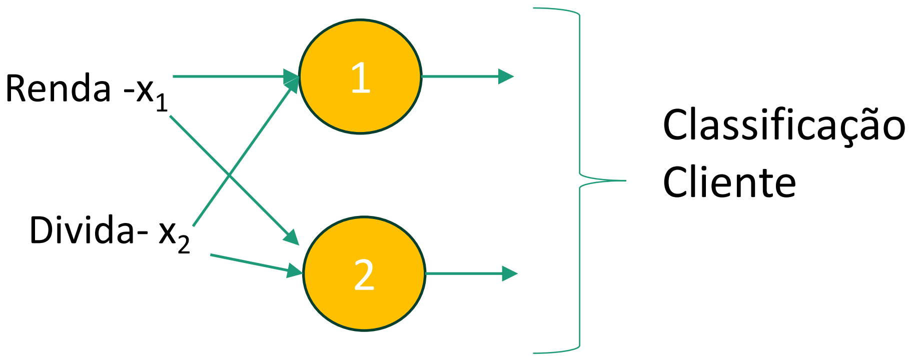


Redes Feed Forward

Rede Perceptron - Topologia

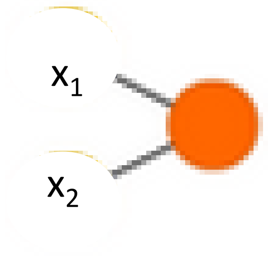
Classificação Cliente

renda	dívida	D1	D2
0,66	0,5	1	0
1	1	0	1
0,33	0,25	0	1
1	0,75	1	0



Topologia mais usual.
Neurônio dedicados para cada classe:

- Haverá uma saída esperada para cada neurônio.
- A saída da rede é dependente dos vários neurônios.
- Pode ser escolhido aquele de maior valor de saída.

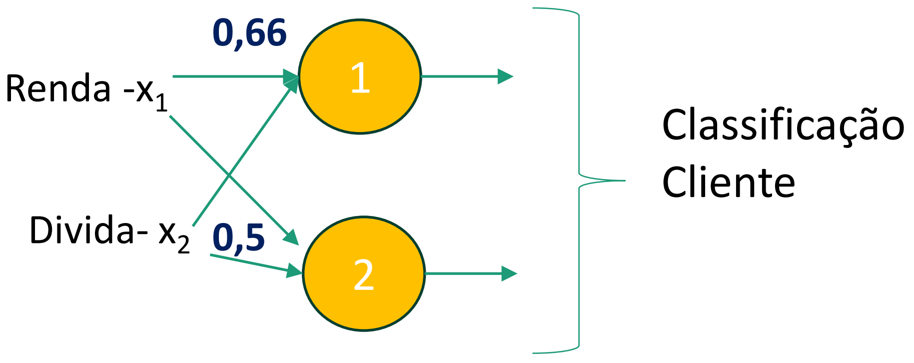


Redes Feed Forward

Rede Perceptron - Topologia

Classificação Cliente

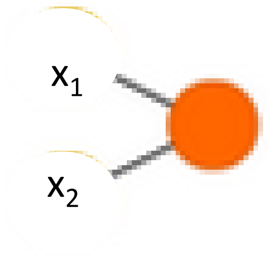
renda	dívida	D1	D2
0,66	0,5	1	0
1	1	0	1
0,33	0,25	0	1
1	0,75	1	0



Topologia mais usual.

Neurônio dedicados para cada classe:

- Haverá uma saída esperada para cada neurônio.
- A saída da rede é dependente dos vários neurônios.
- Pode ser escolhido aquele de maior valor de saída.

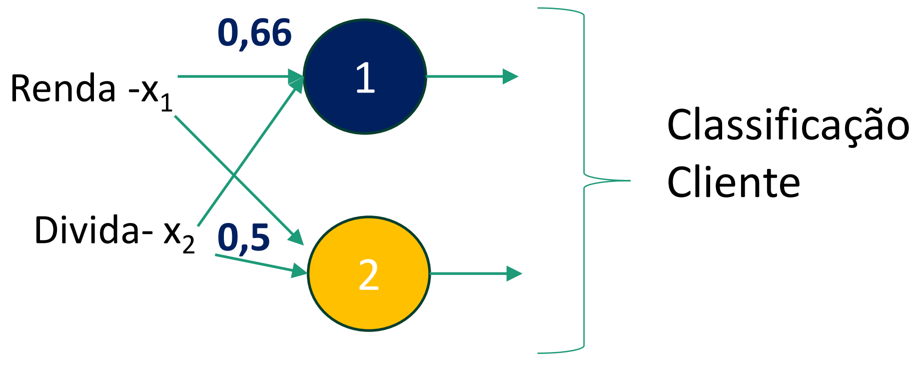


Redes Feed Forward

Rede Perceptron - Topologia

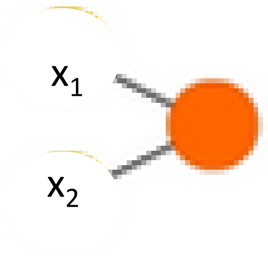
Classificação Cliente

renda	dívida	D1	D2
0,66	0,5	1	0
1	1	0	1
0,33	0,25	0	1
1	0,75	1	0



Topologia mais usual.
Neurônio dedicados para cada classe:

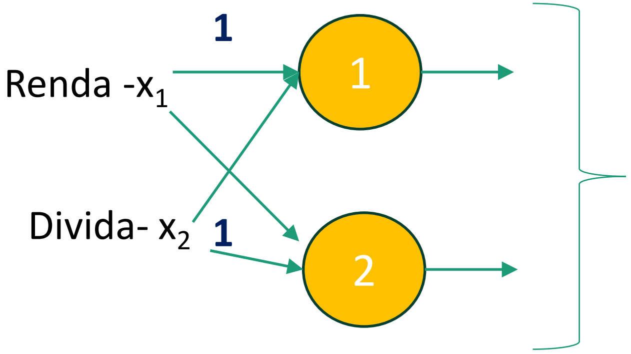
- Haverá uma saída esperada para cada neurônio.
- A saída da rede é dependente dos vários neurônios.
- Pode ser escolhido aquele de maior valor de saída.



Redes Feed Forward

Rede Perceptron - Topologia

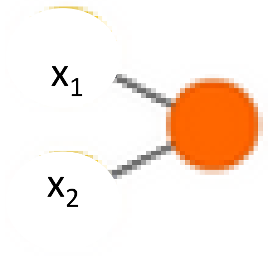
renda	dívida	Classificação Cliente D1	D2
0,66	0,5	1	0
1	1	0	1
0,33	0,25	0	1
1	0,75	1	0



Topologia mais usual.

Neurônio dedicados para cada classe:

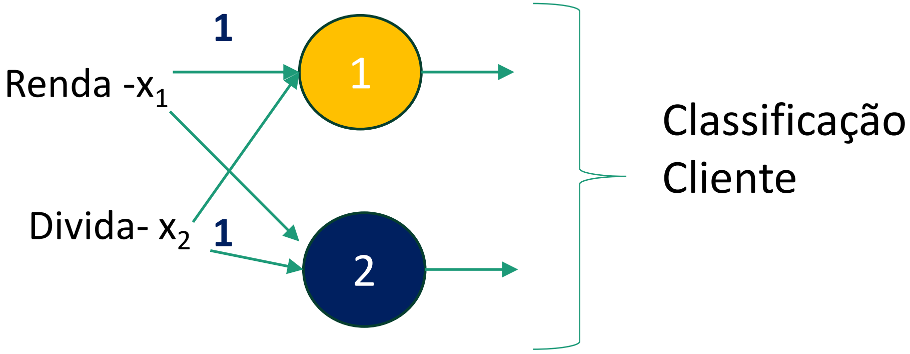
- Haverá uma saída esperada para cada neurônio.
- A saída da rede é dependente dos vários neurônios.
- Pode ser escolhido aquele de maior valor de saída.



Redes Feed Forward

Rede Perceptron - Topologia

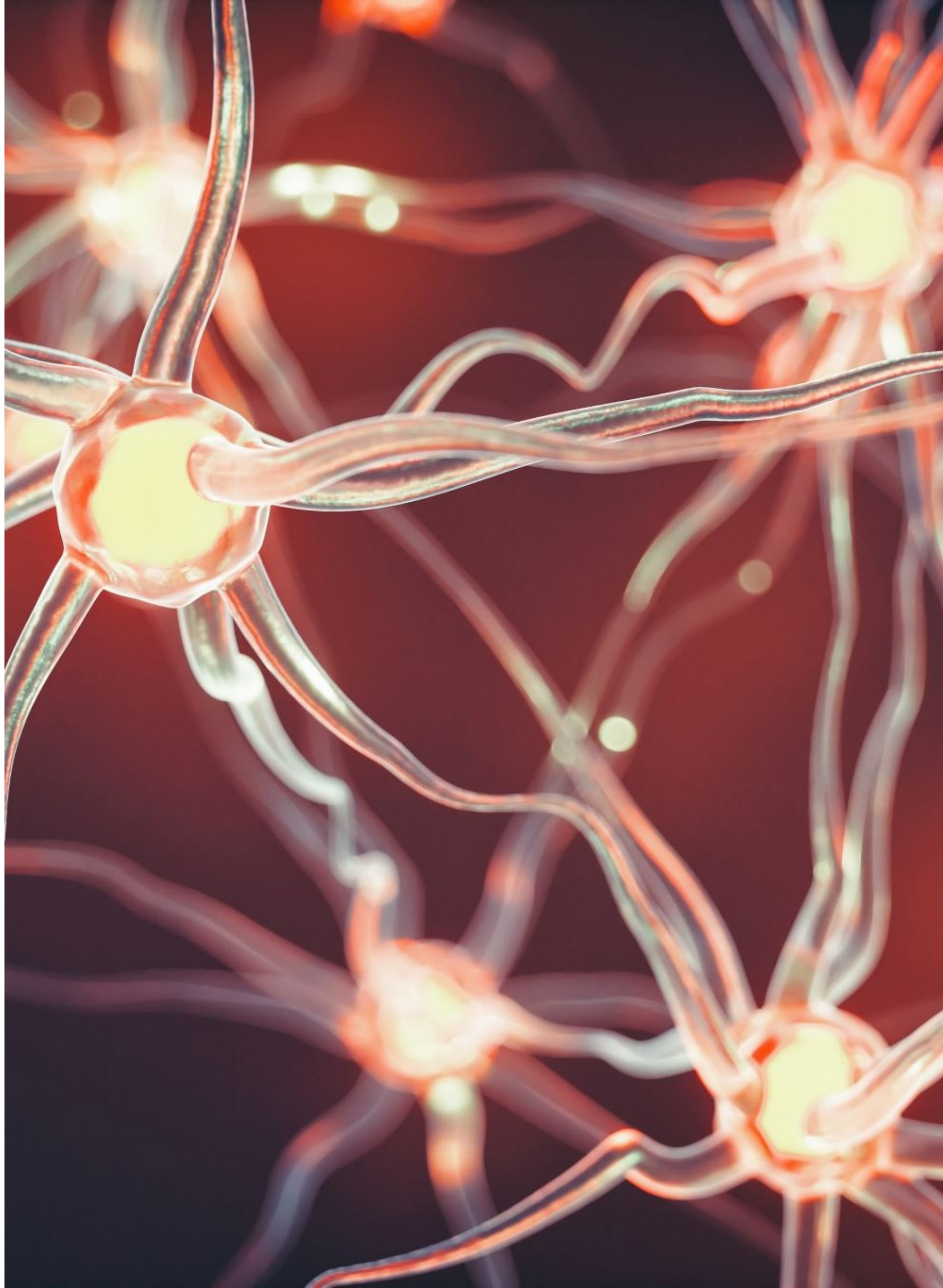
renda	dívida	Classificação Cliente D1	Classificação Cliente D2
0,66	0,5	1	0
1	1	0	1
0,33	0,25	0	1
1	0,75	1	0



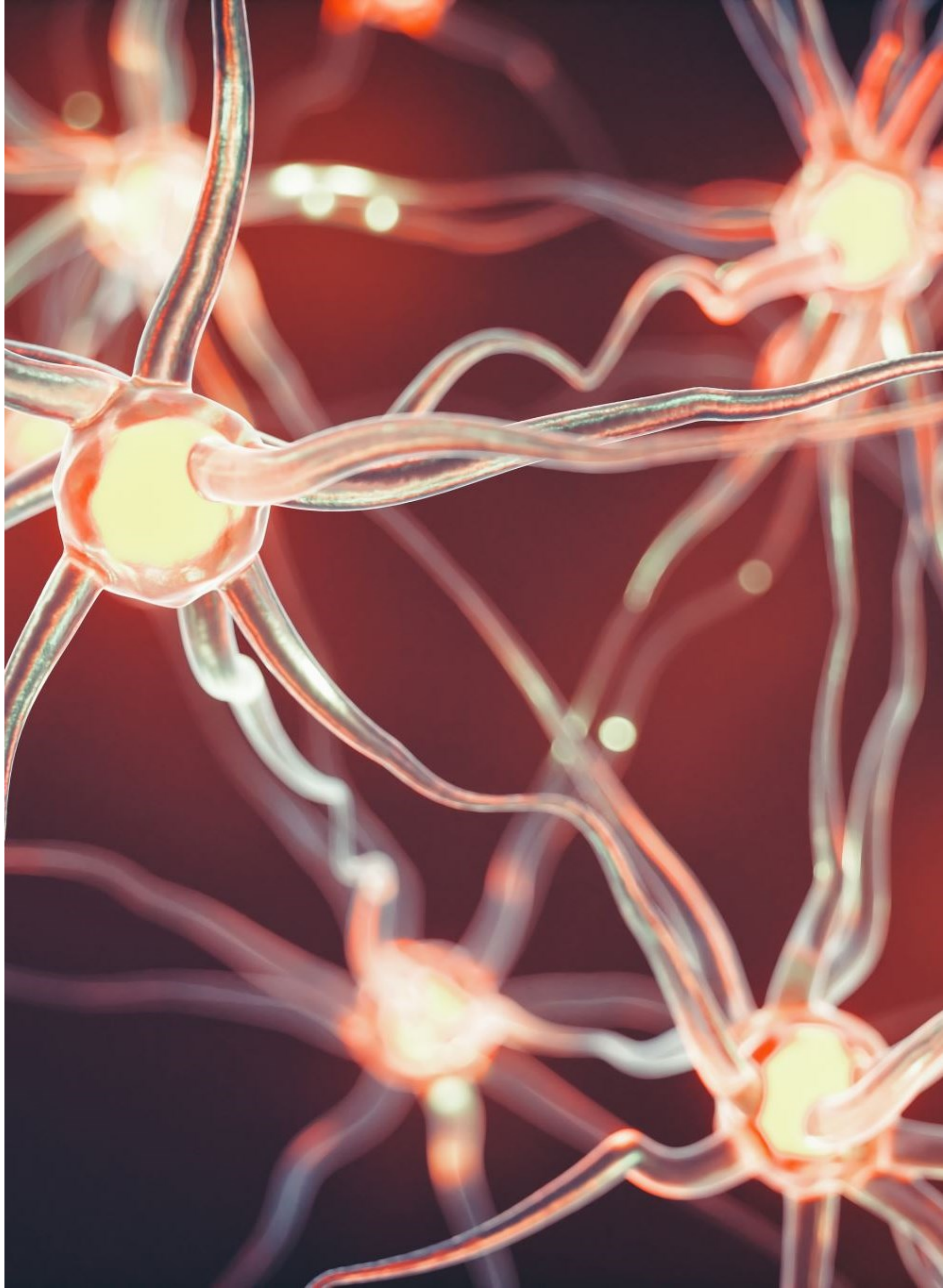
Topologia mais usual.
Neurônio dedicados para cada classe:

- Haverá uma saída esperada para cada neurônio.
- A saída da rede é dependente dos vários neurônios.
- Pode ser escolhido aquele de maior valor de saída.

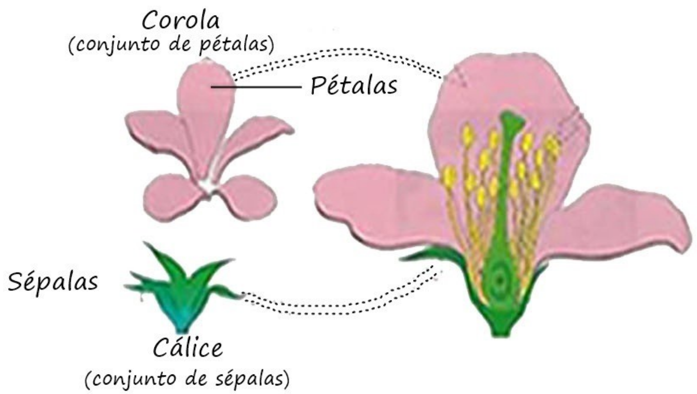
Mais exemplos de topologia ...



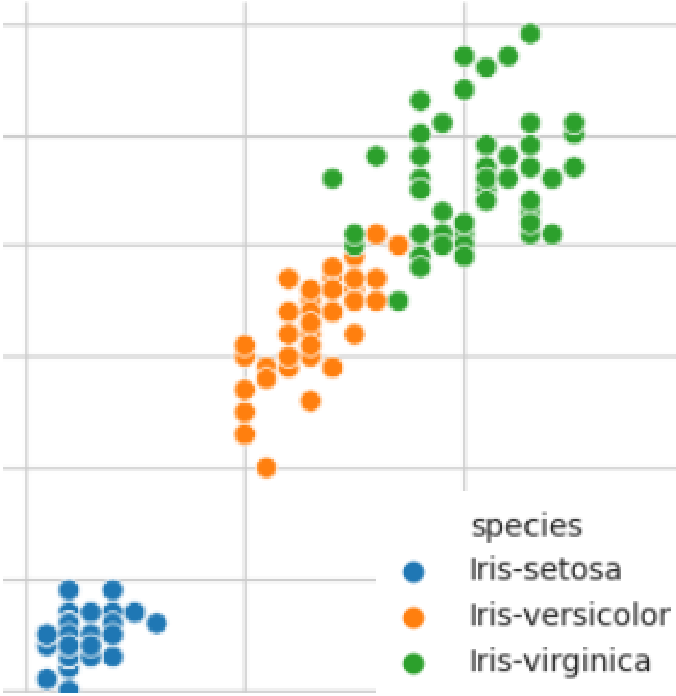
Exemplo 1 – Dados estruturados



Planta Iris

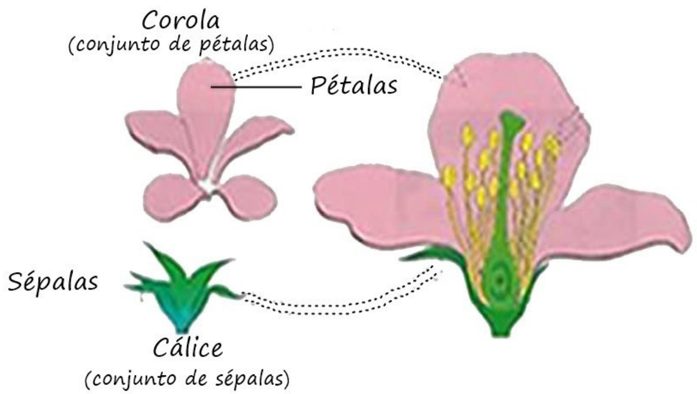


4 atributos de entrada
3 classes: Iris-setosa, Iris-versicolor, Iris-virginica



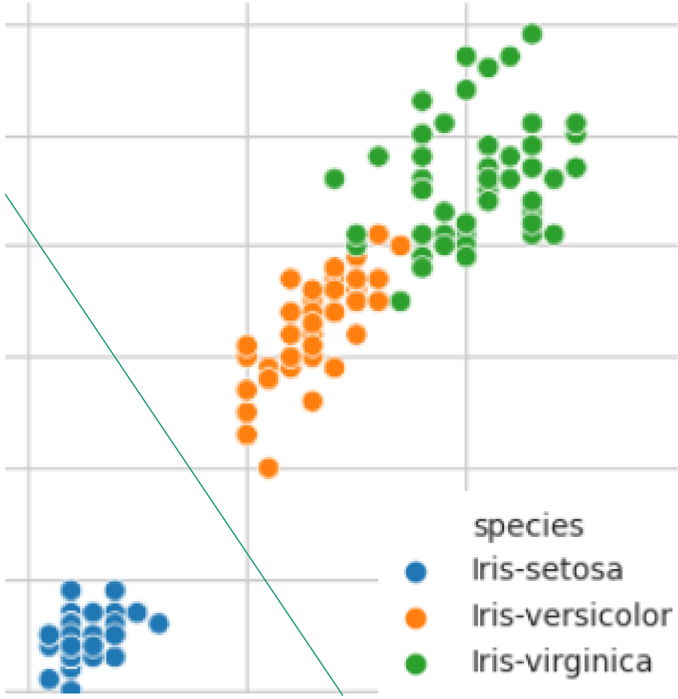
	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
5	5.4	3.9	1.7	0.4	Iris-setosa
6	4.6	3.4	1.4	0.3	Iris-setosa
7	5.0	3.4	1.5	0.2	Iris-setosa
8	4.4	2.9	1.4	0.2	Iris-setosa
9	4.9	3.1	1.5	0.1	Iris-setosa

Planta Iris



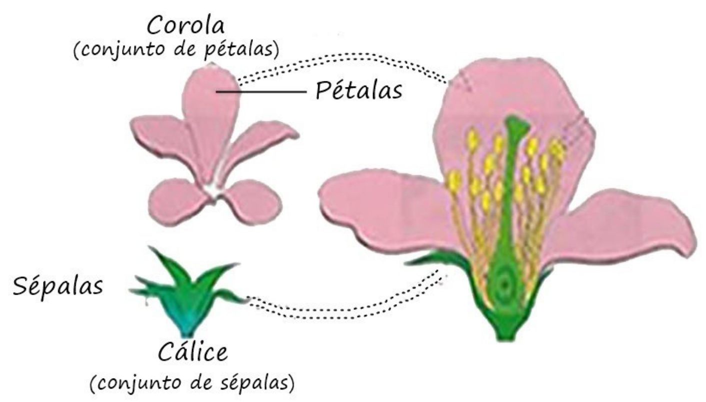
4 atributos de entrada

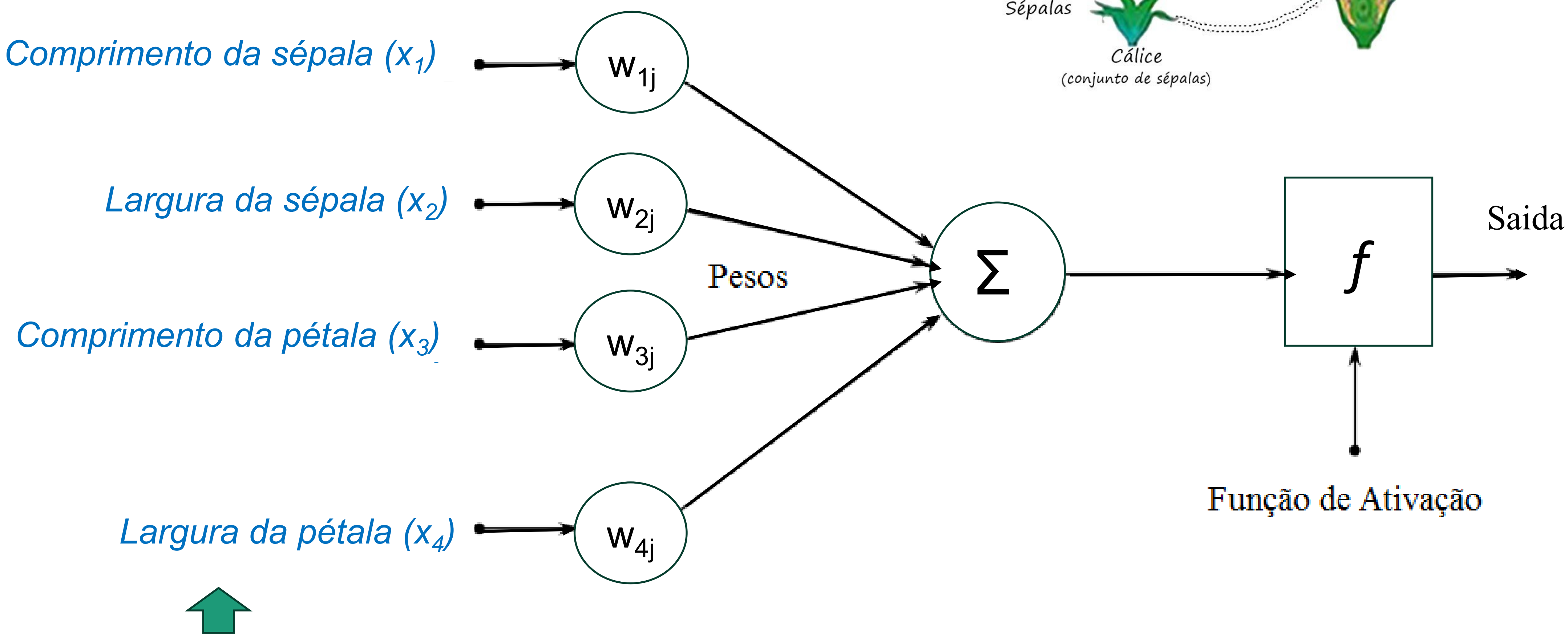
2 classes: Iris-setosa, Nao Iris-setosa



	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
5	5.4	3.9	1.7	0.4	Iris-setosa
6	4.6	3.4	1.4	0.3	Iris-setosa
7	5.0	3.4	1.5	0.2	Iris-setosa
8	4.4	2.9	1.4	0.2	Iris-setosa
9	4.9	3.1	1.5	0.1	Iris-setosa

Planta Iris

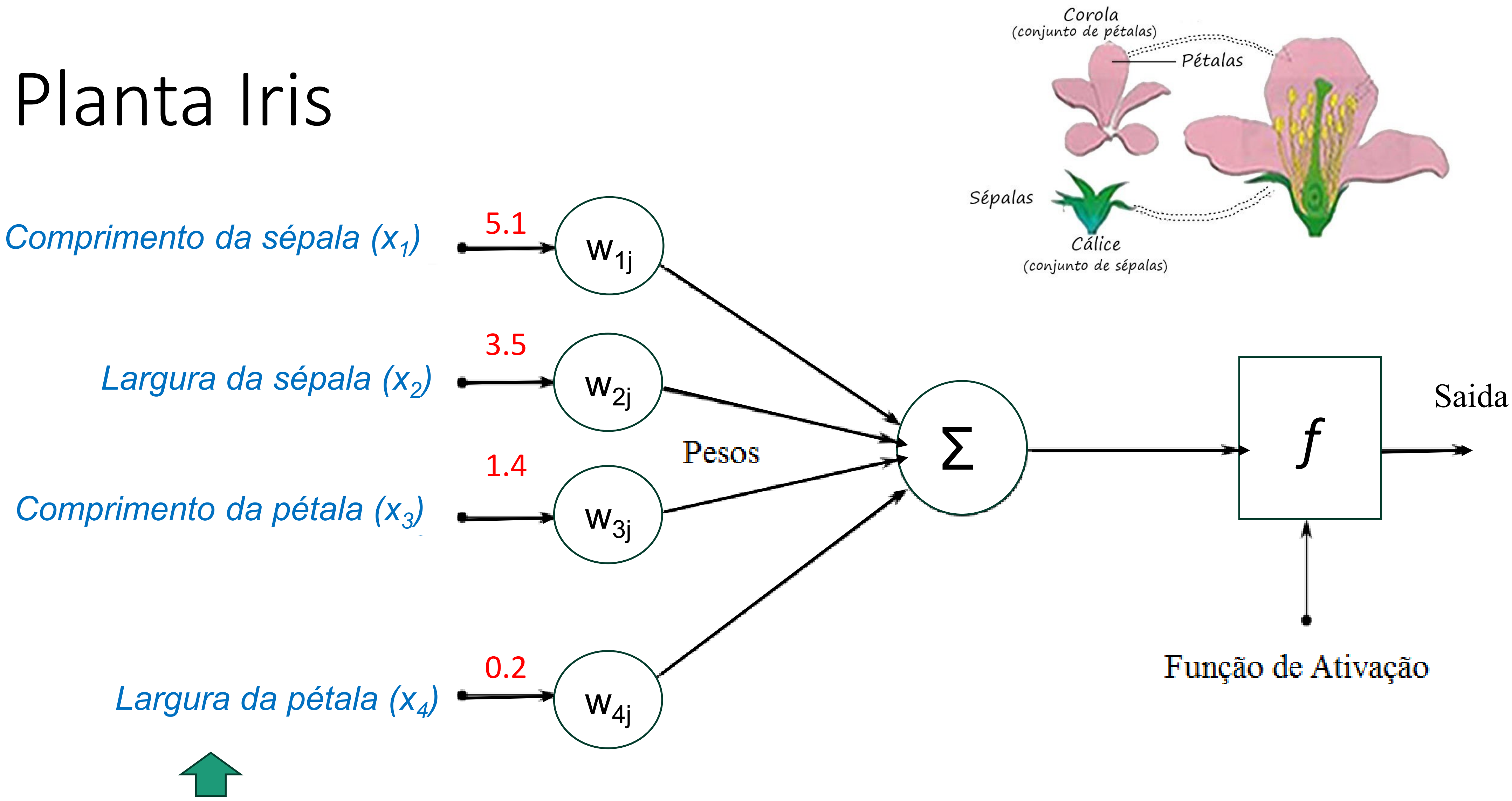




Atributos dos objetos

Entrada com Dados estruturados

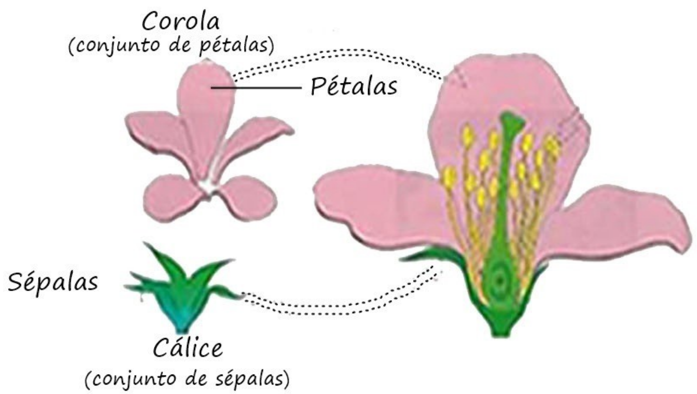
Planta Iris



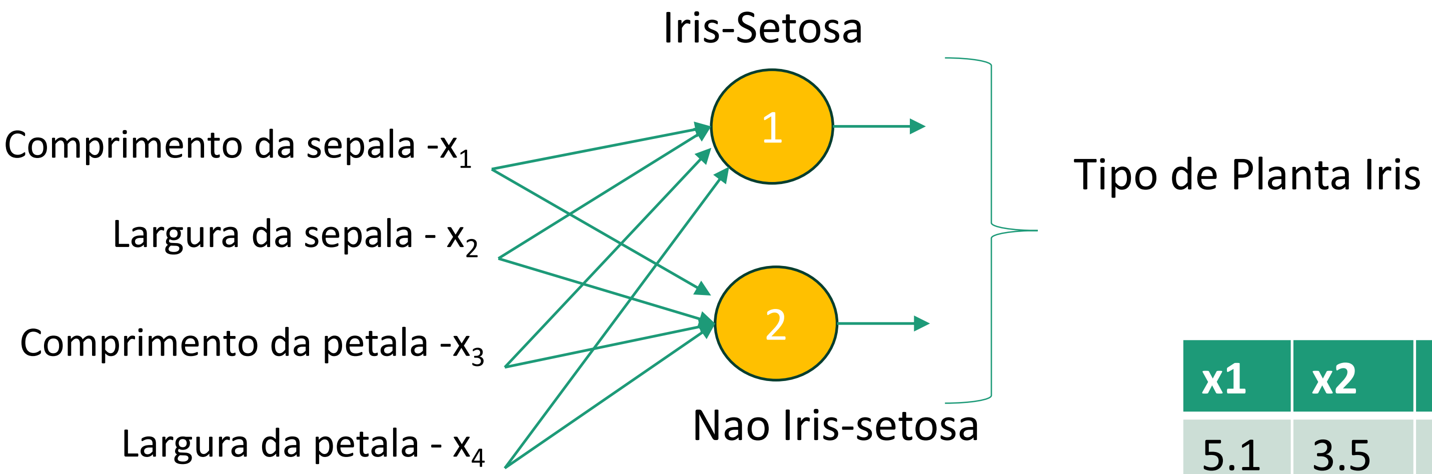
Atributos dos objetos

Entrada com Dados estruturados

Planta Iris



Topologia para 2 classes
Classificador Binario Neural



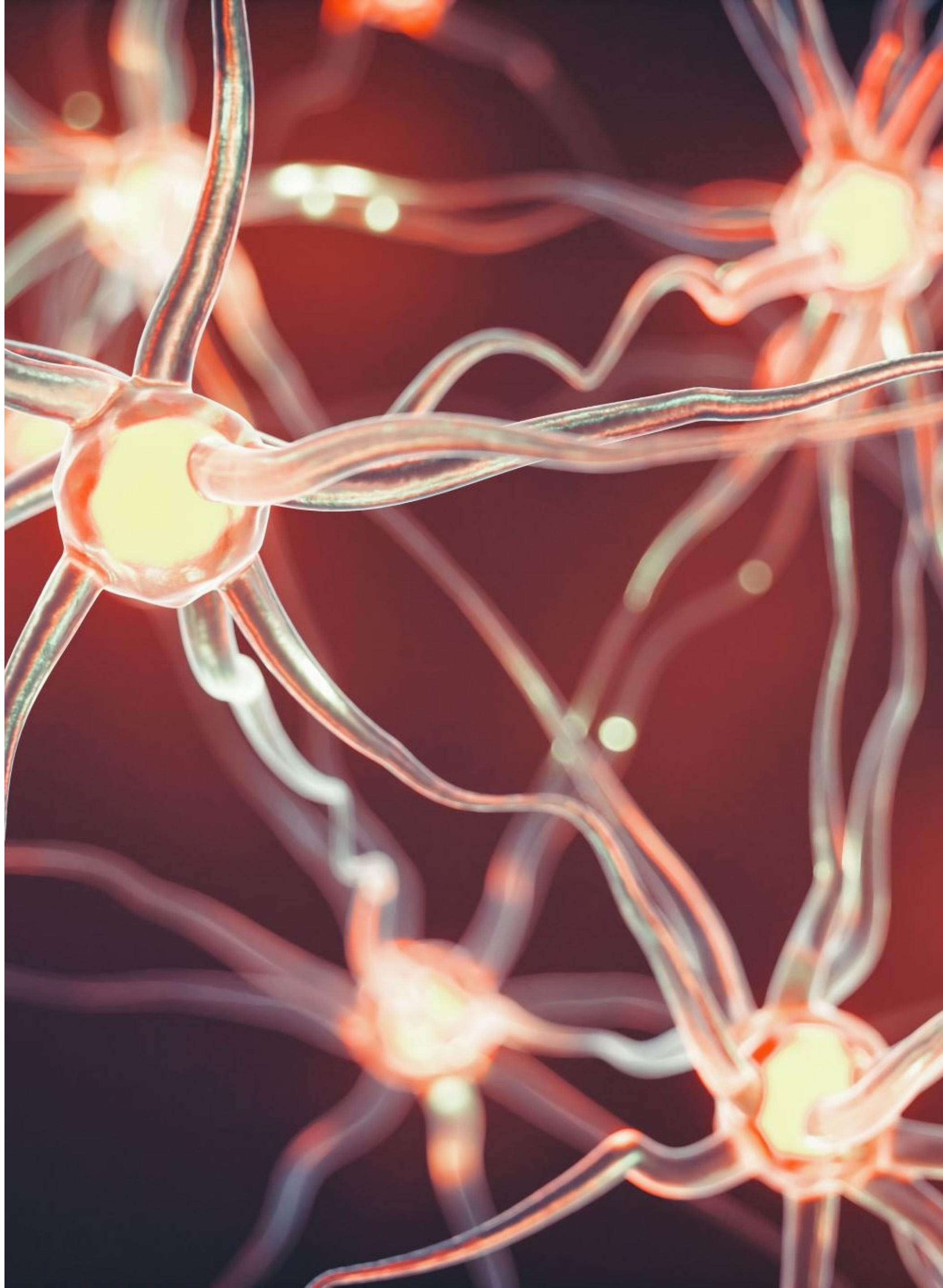
Topologia: 4 x 2

Camada de entrada(dados): 4

Camada de saida (neuronios): 2

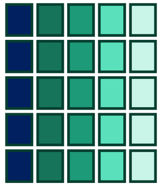
x1	x2	x3	x4	d1	d2	
5.1	3.5	1.4	0.2	1	0	Iris-Setosa
7	3.2	4.7	1.4	0	1	Nao Iris-Setosa
…						

Exemplo 2 – Dados desestruturados

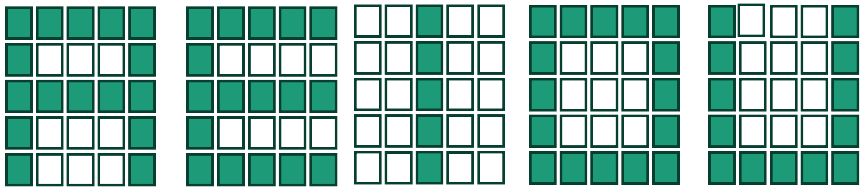


Reconhecedor de Caracteres

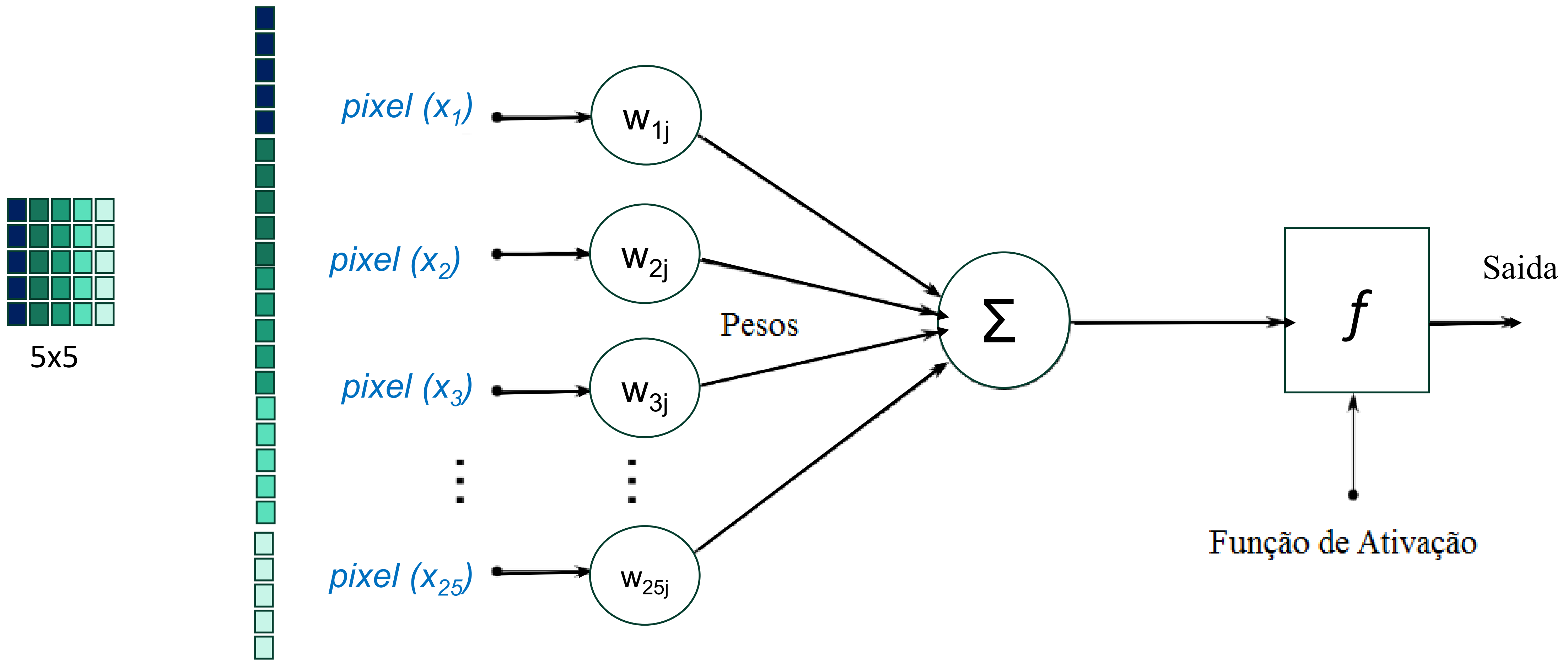
- Considere como entrada de uma imagem de caracteres: 5 x 5.



- Objetivo: identificar as imagens correspondentes a vogais:

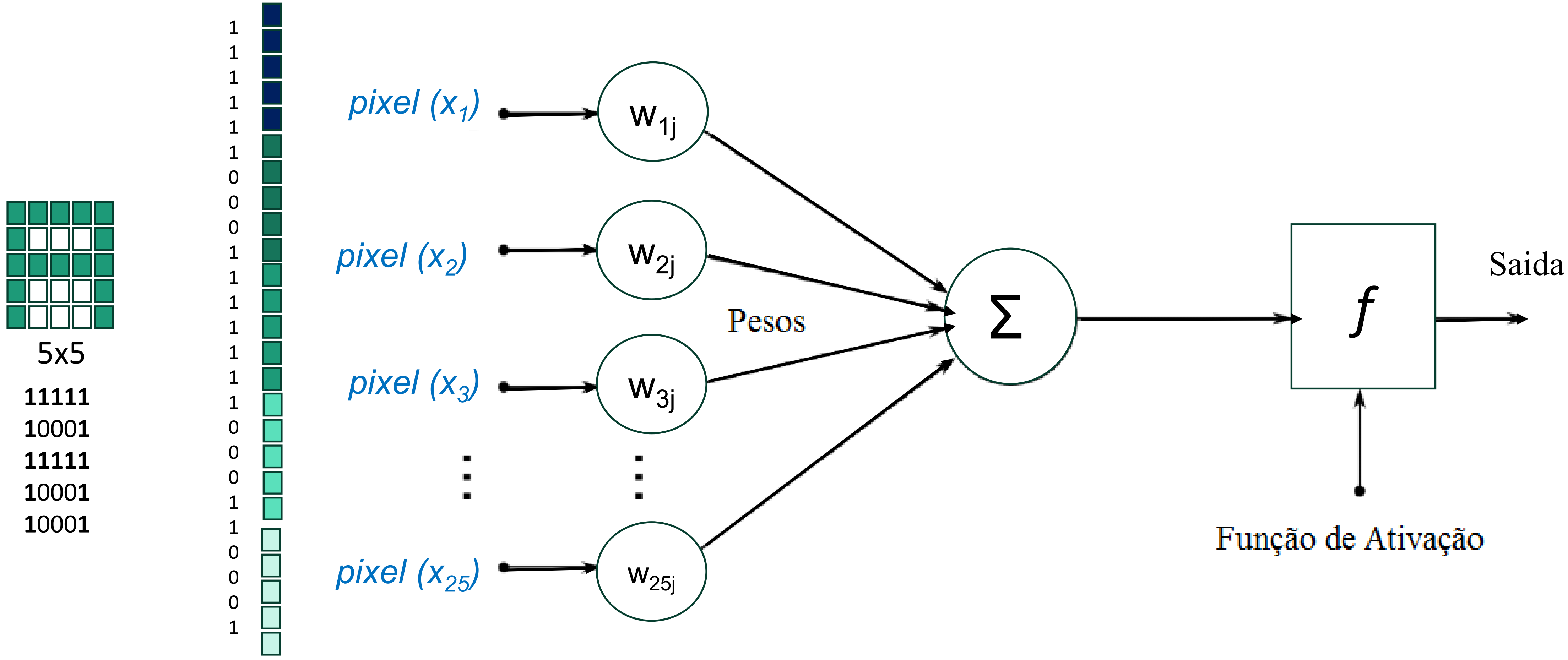


Reconhecedor de Caracteres

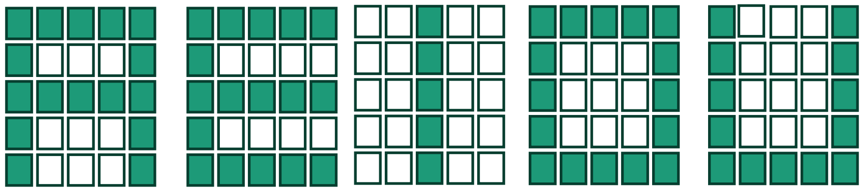


Atributos dos objetos

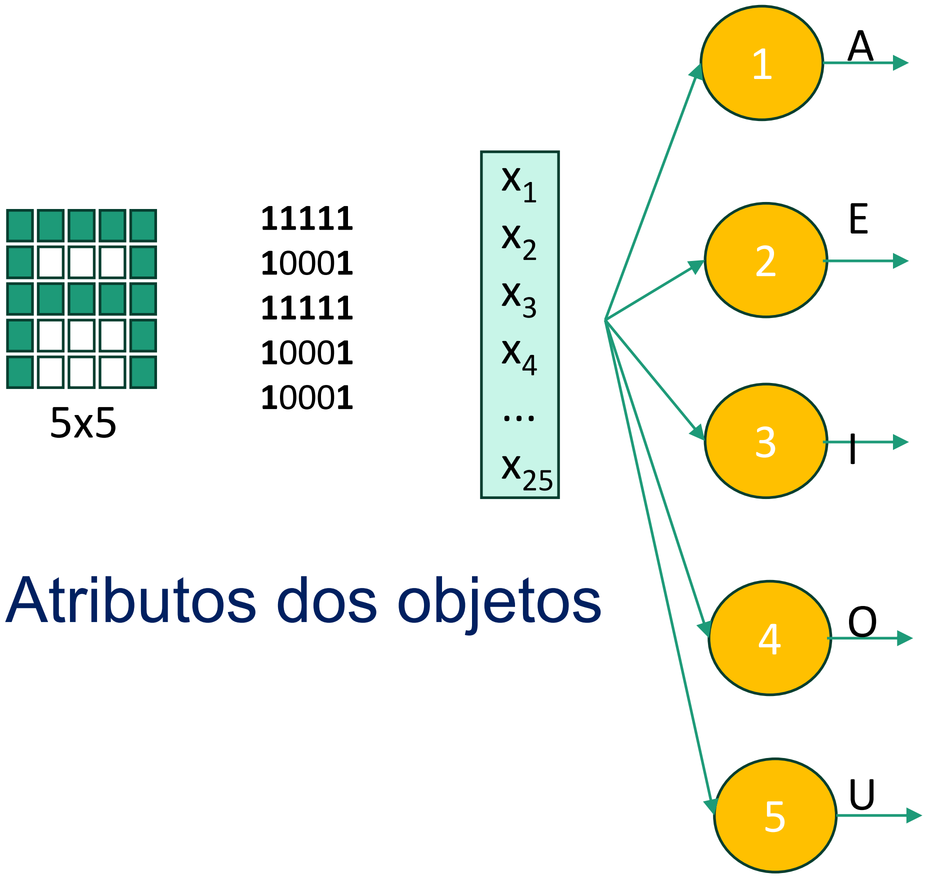
Reconhecedor de Caracteres



Atributos dos objetos

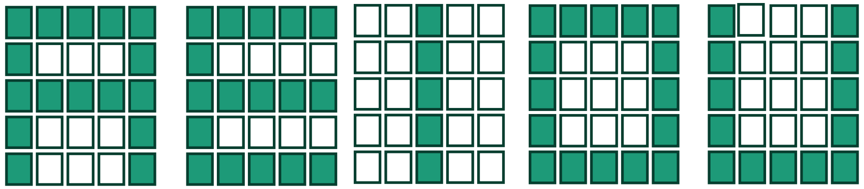


Reconhecedor de Caracteres



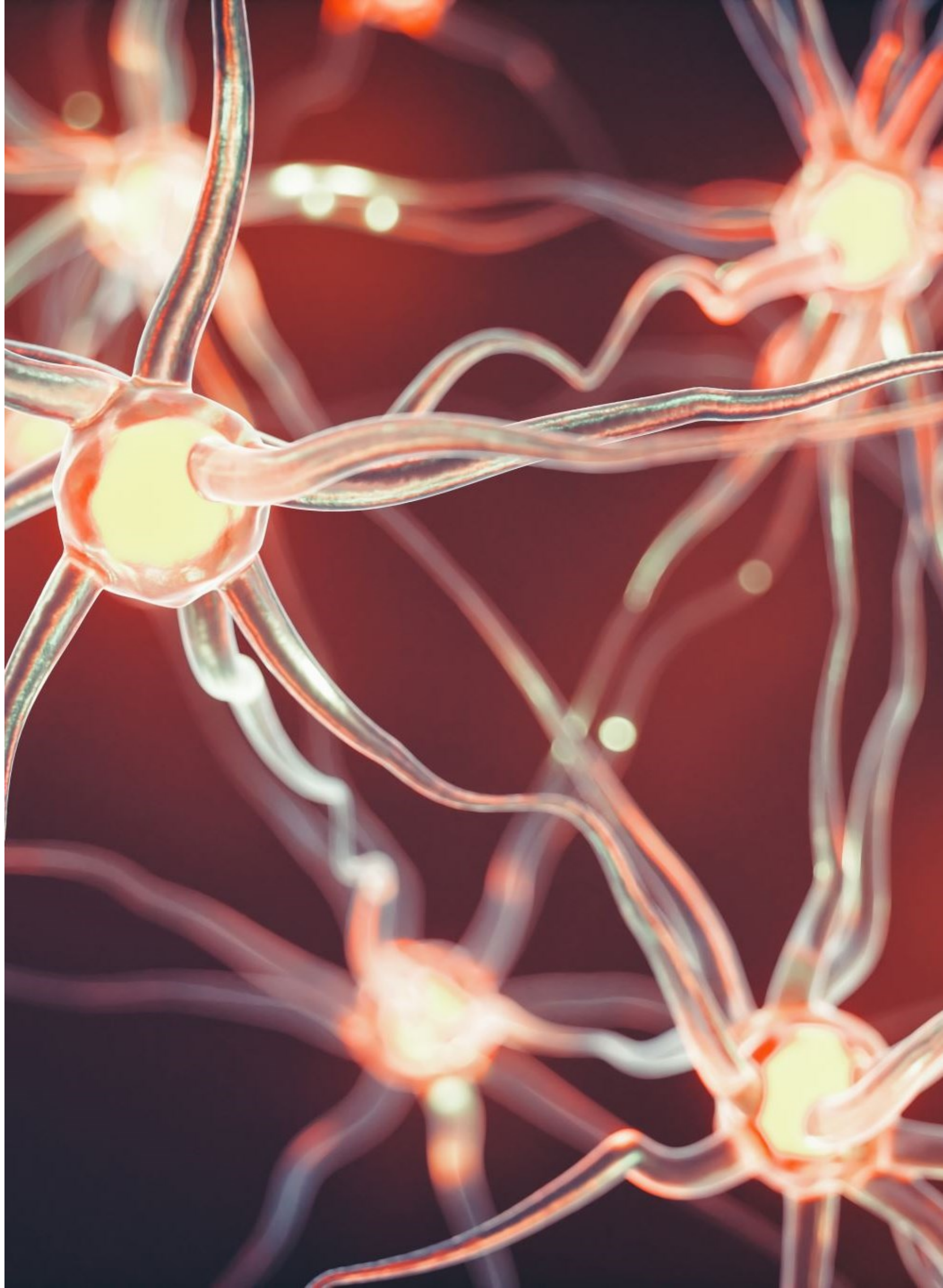
Atributos dos objetos

Topologia: 25 x 5

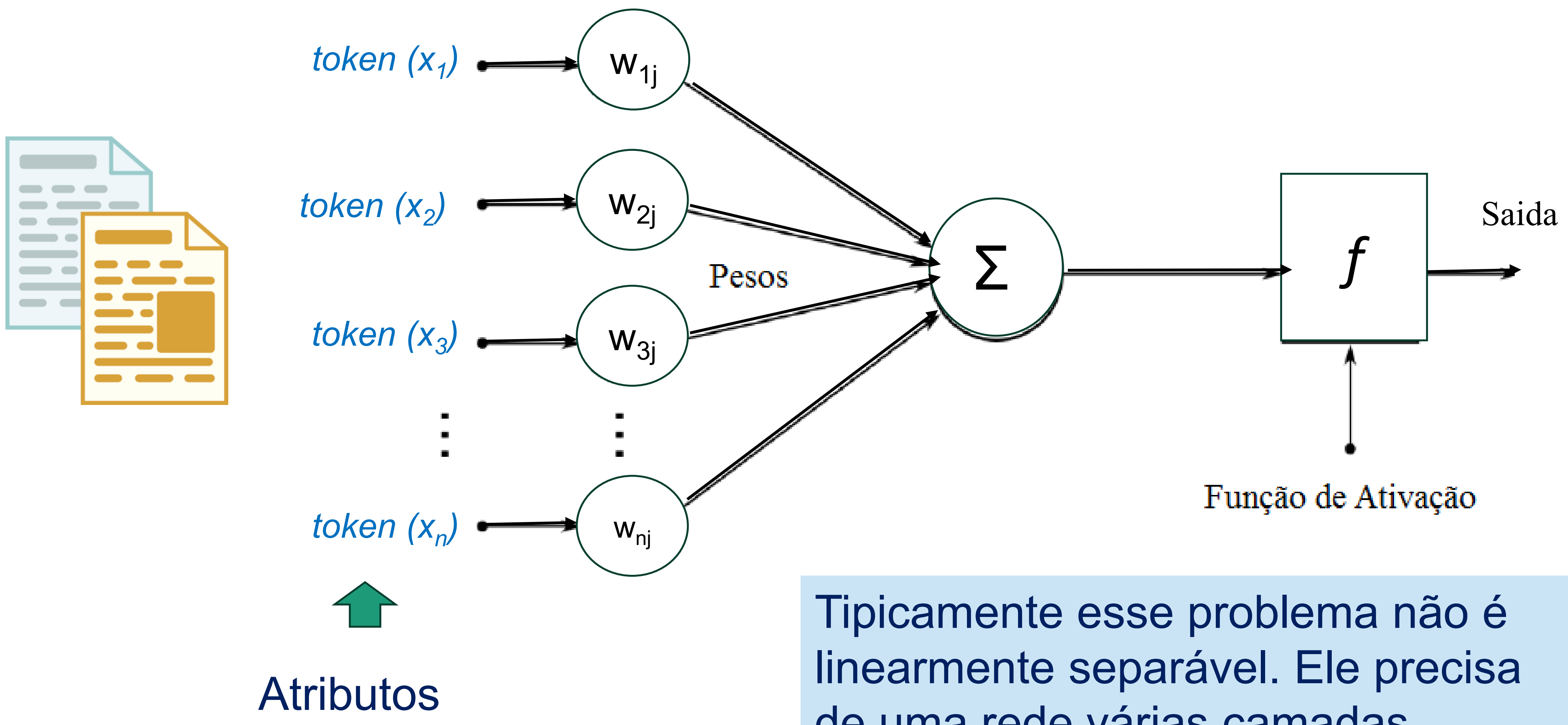


x1,x2,x3,...,x25	d1	d2	d3	d4	d5
1,1,1,1,1,1,0,...	1	0	0	0	0
1,1,1,1,1,1,0,...	0	1	0	0	0
0,0,1,0,0,0,0,...	0	0	1	0	0
1,1,1,1,1,1,0,...	0	0	0	1	0
1,0,0,0,1,1,0,...	0	0	0	0	1
...					

Exemplo 3 – Dados desestruturados



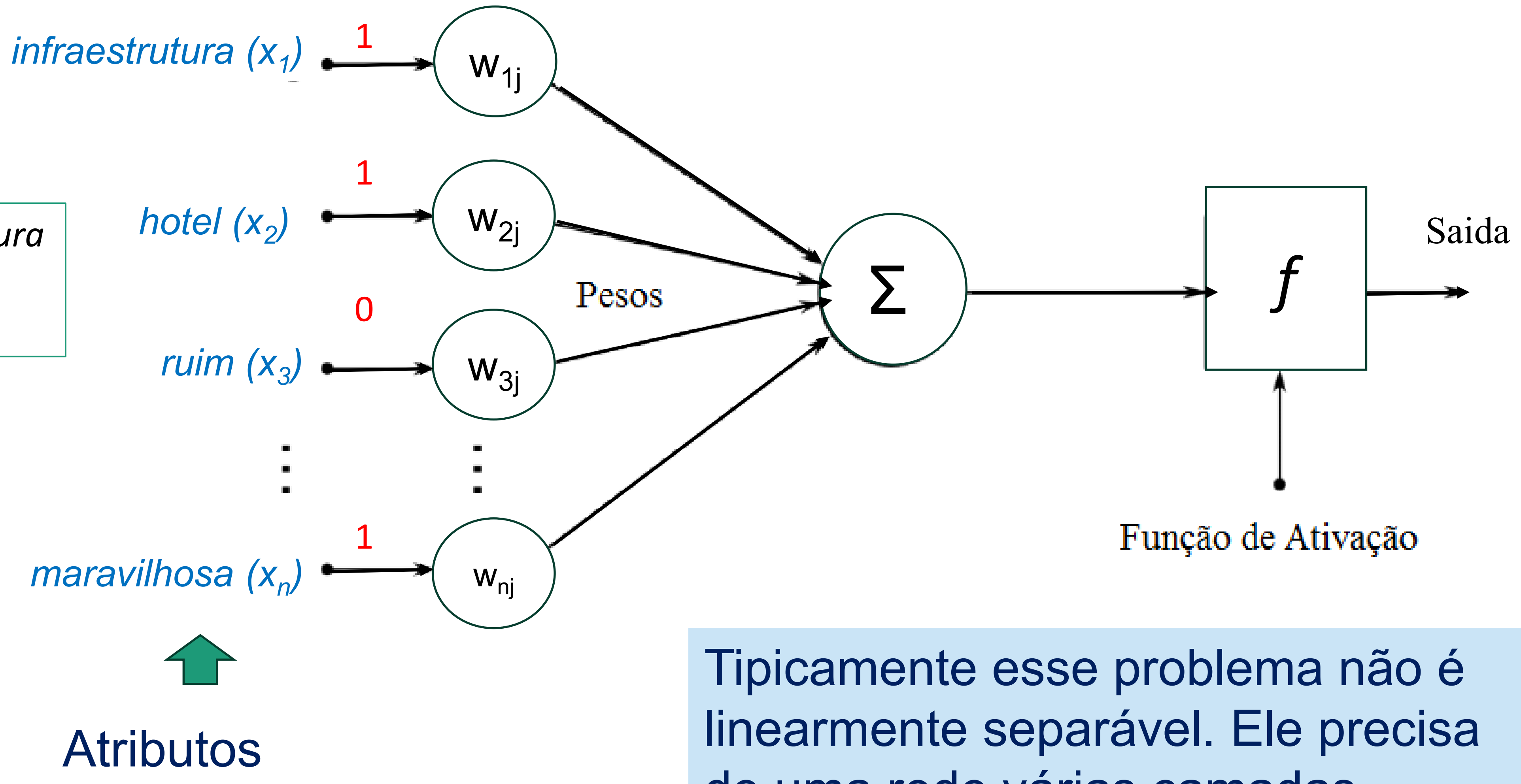
Classificador de sentenças



Tipicamente esse problema não é linearmente separável. Ele precisa de uma rede várias camadas.

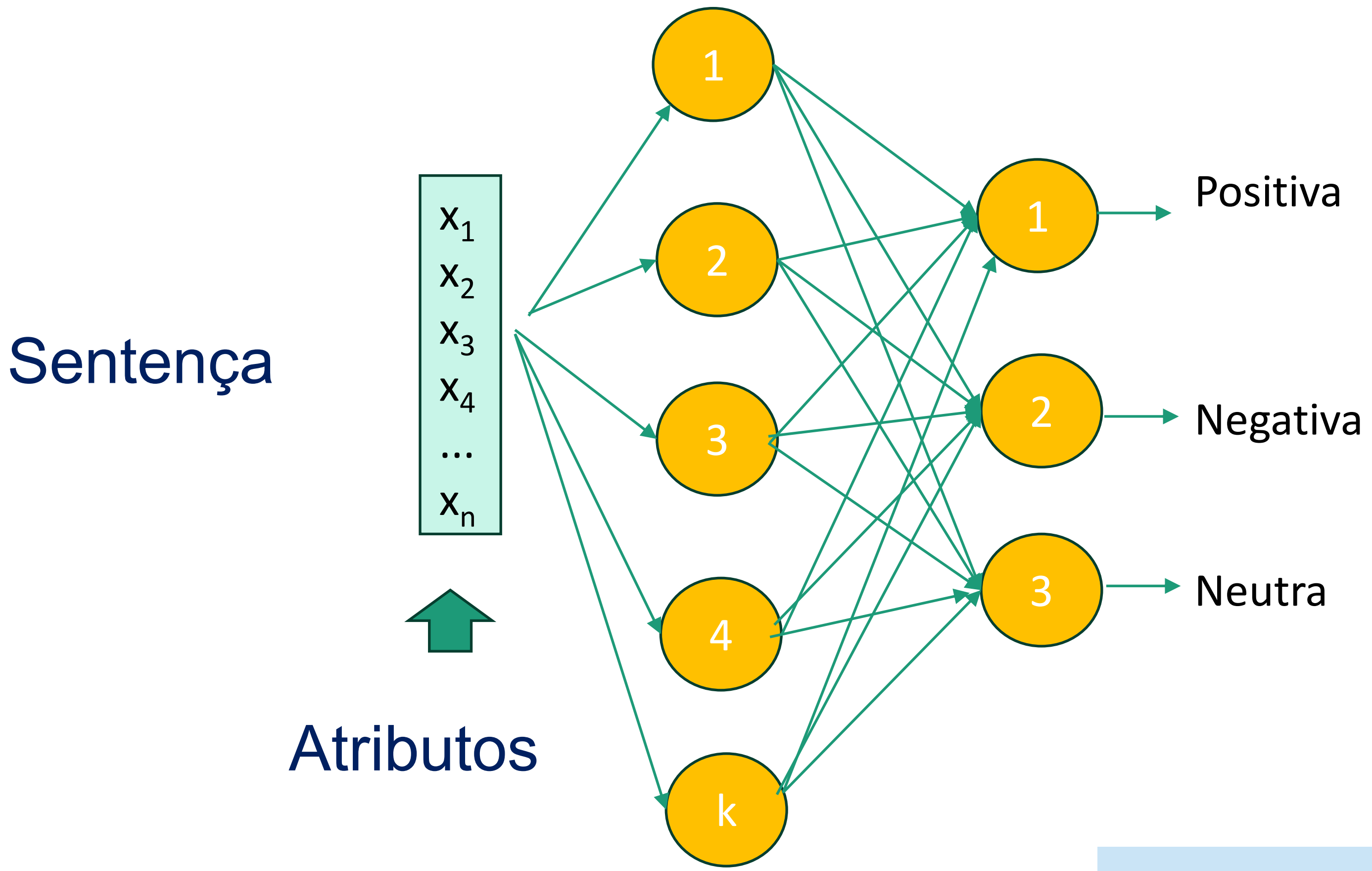
Classificador de sentenças

A infraestrutura do hotel era maravilhosa.



Tipicamente esse problema não é linearmente separável. Ele precisa de uma rede várias camadas.

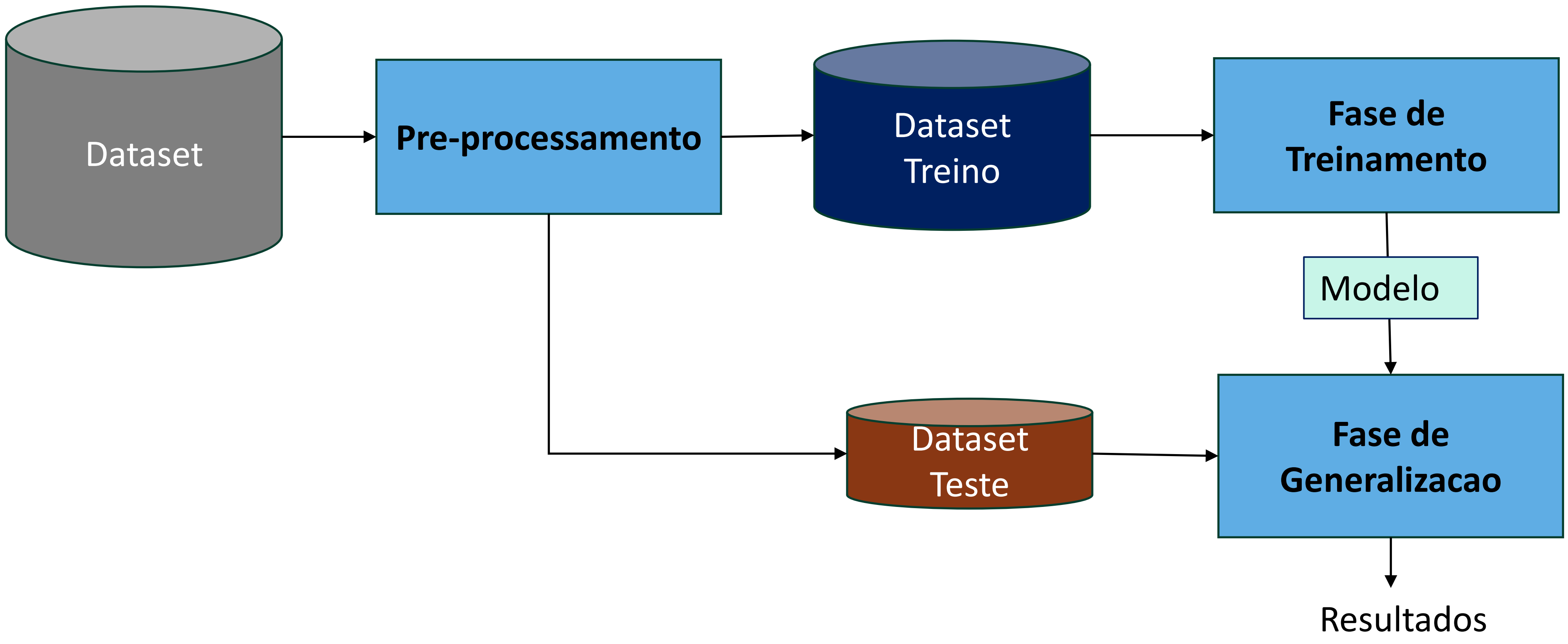
Classificador de sentenças



Veremos mais detalhes dessa topologia na aula sobre MultiLayer Perceptron.

Tipicamente esse problema não é linearmente separável. Ele precisa de uma rede várias camadas.

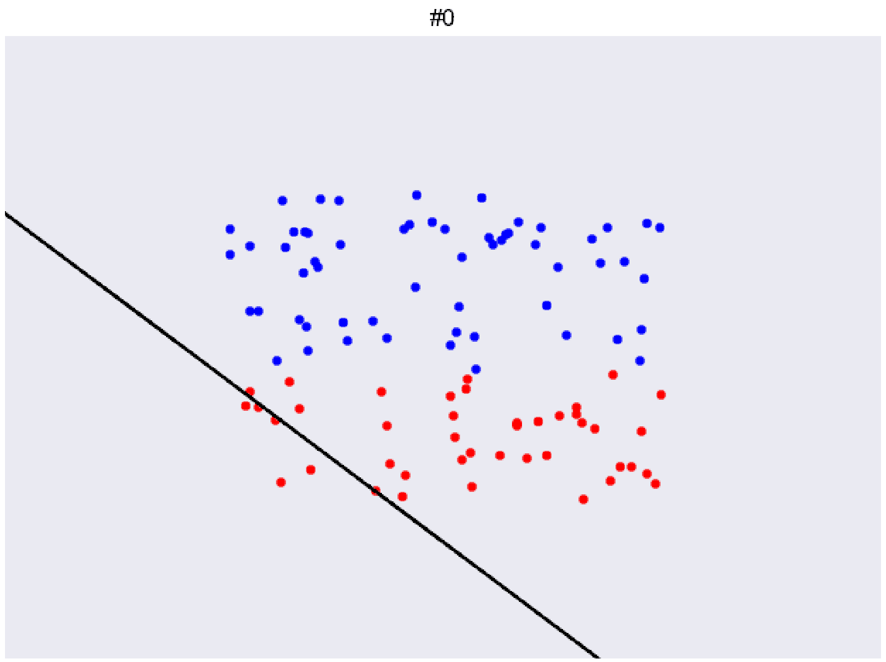
Etapas Básicas de Construção de uma Rede Neural



Rede Perceptron

- Como já mencionado, essas redes só conseguem resolver problemas linearmente separáveis.

- O algoritmo de aprendizado desse tipo de rede, procura os coeficientes que traçam a reta que separa linearmente os dados de uma classe de outra.

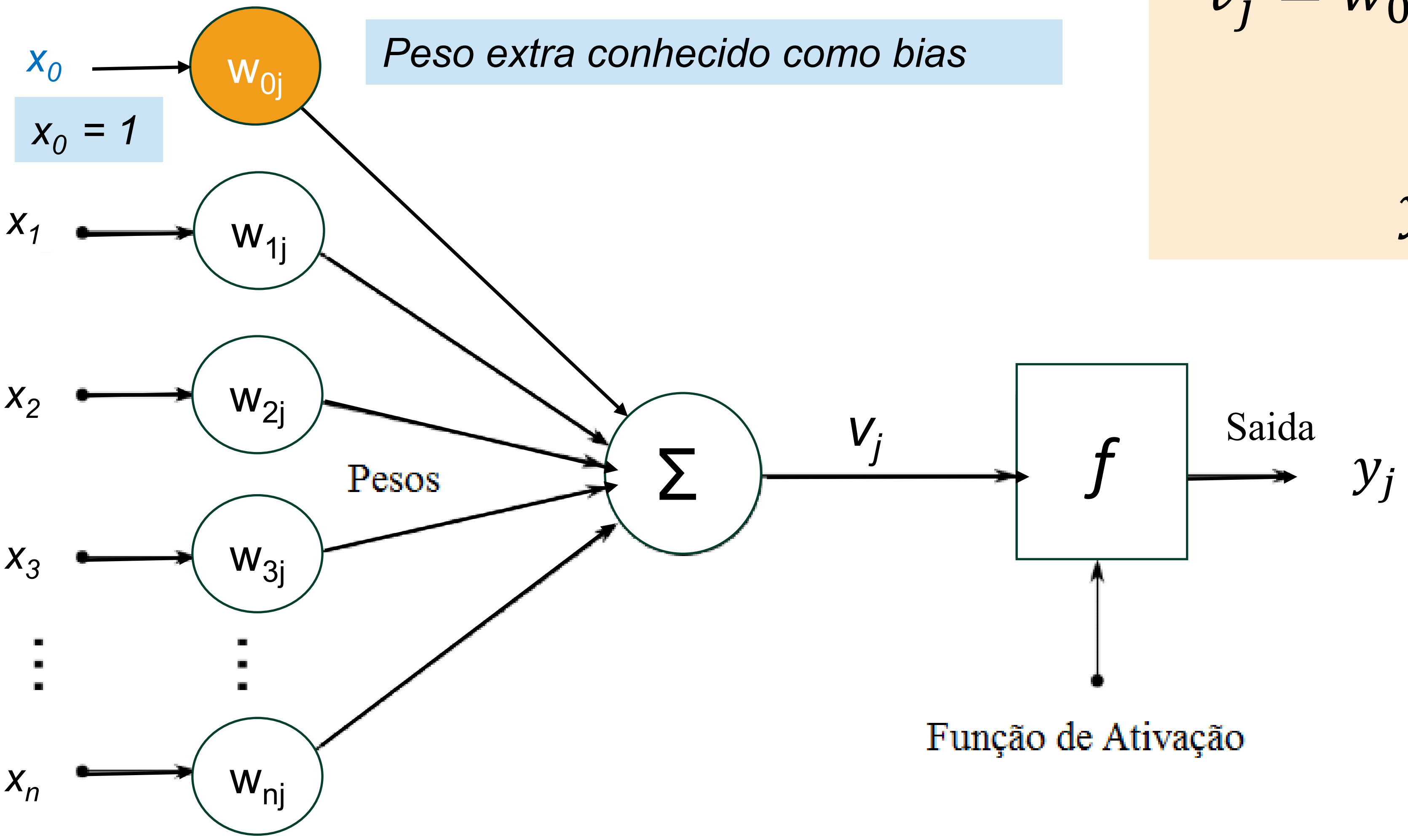


Rede Perceptron

Revisitando um neurônio artificial **j**:

$$v_j = w_{0j} + \sum_{i=1}^{n} (x_i \times w_{ij})$$

$$y_j = f(v_j)$$

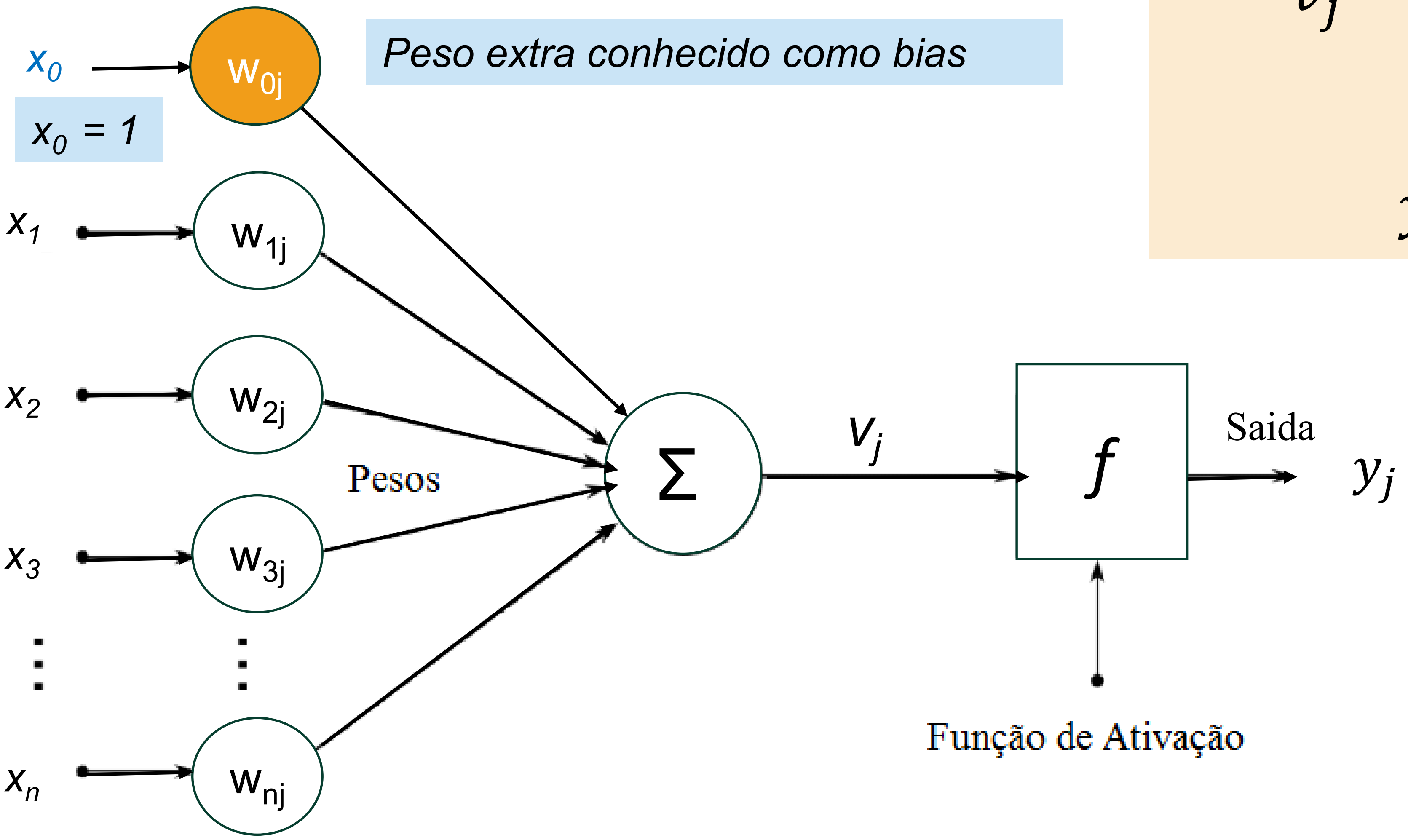


Rede Perceptron

Revisitando um neurônio artificial **j**:

$$v_j = \sum_{i=0}^{n} (x_i \times w_{ij})$$

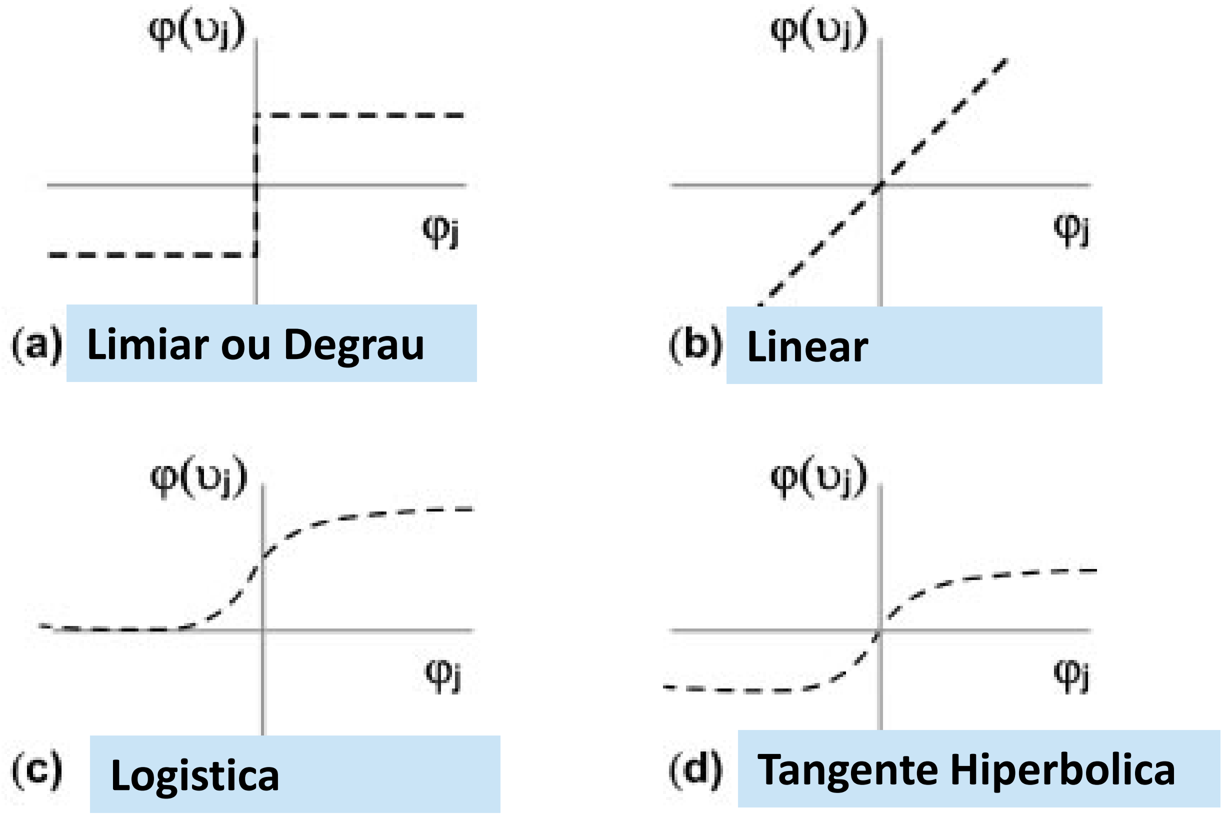
$$y_j = f(v_j)$$



Rede Perceptron

Revisitando um neurônio artificial **j**:

- Existem várias funções de ativação para as redes neurais.
- As clássicas são:
  - Limiar ou degrau
  - Linear
  - Sigmoide: logistica ou tangente hiperbolica
- Geralmente o Perceptron usa a função limiar.

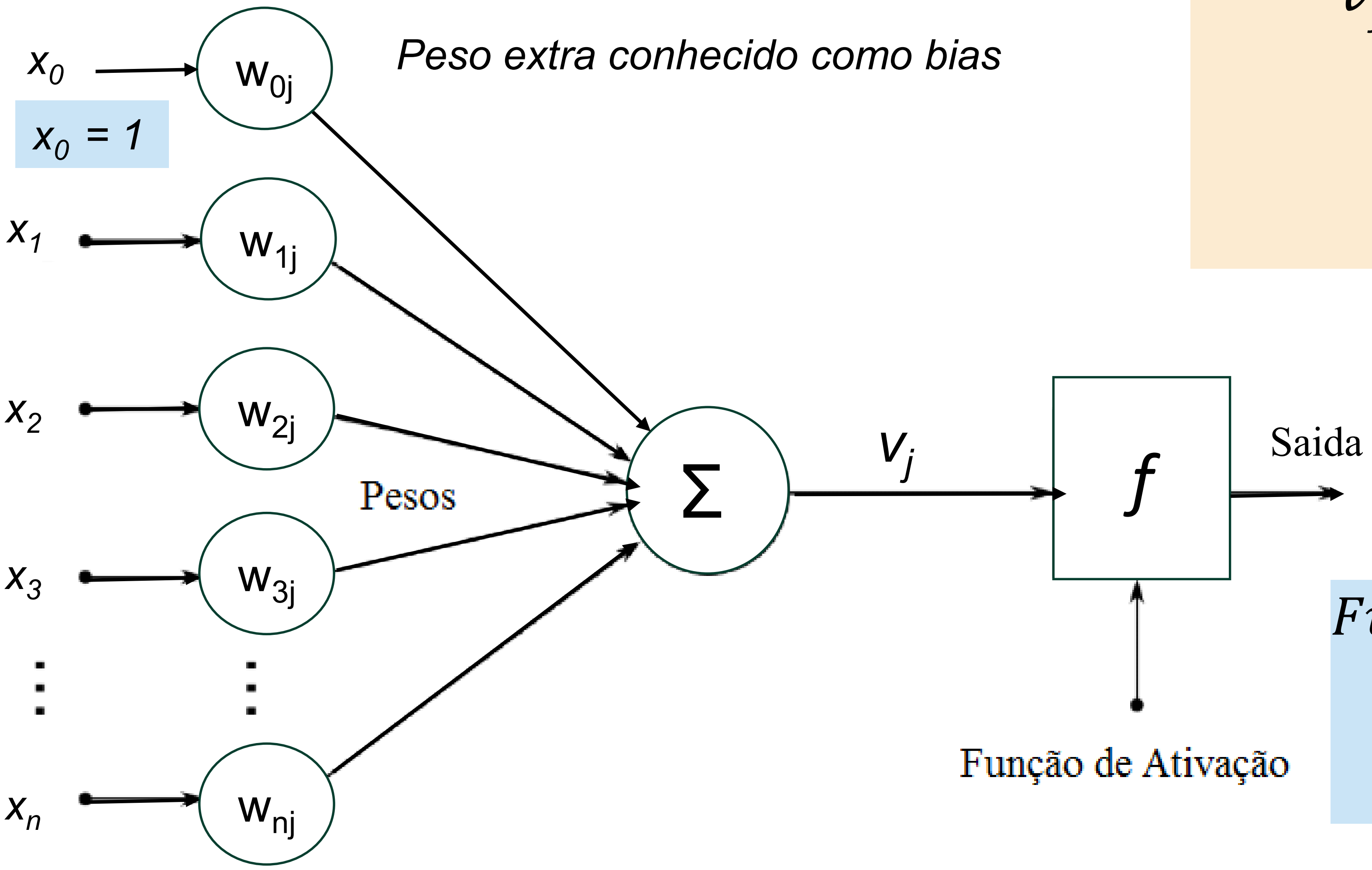


Rede Perceptron

Revisitando um neurônio artificial **j**:

$$v_j = \sum_{i=0}^{n} (x_i \times w_{ij})$$

$$y_j = f(v_j)$$



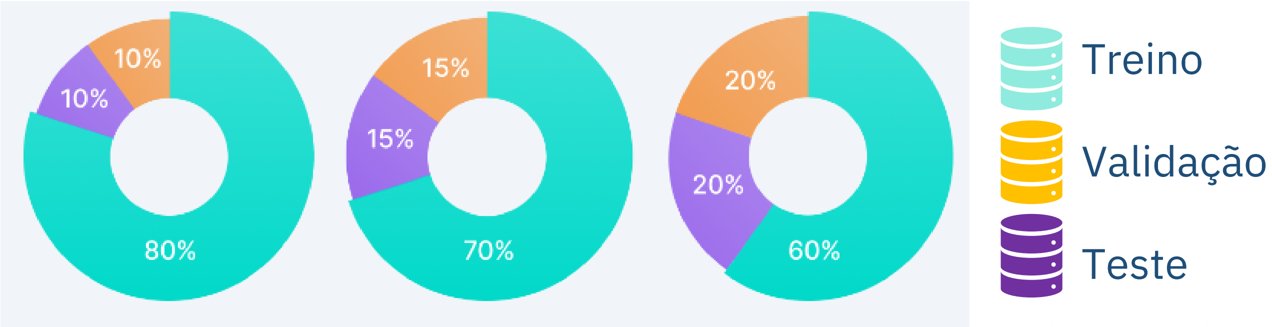
Função de ativação - Limiar

$$f(v_j) = \begin{cases} 1 \ se \ v_j \geq 0 \\ 0 \ se \ v_j < 0 \end{cases}$$

Treinamento de Rede Neurais

Dataset com dados históricos, pre-processado e anotado com classes:

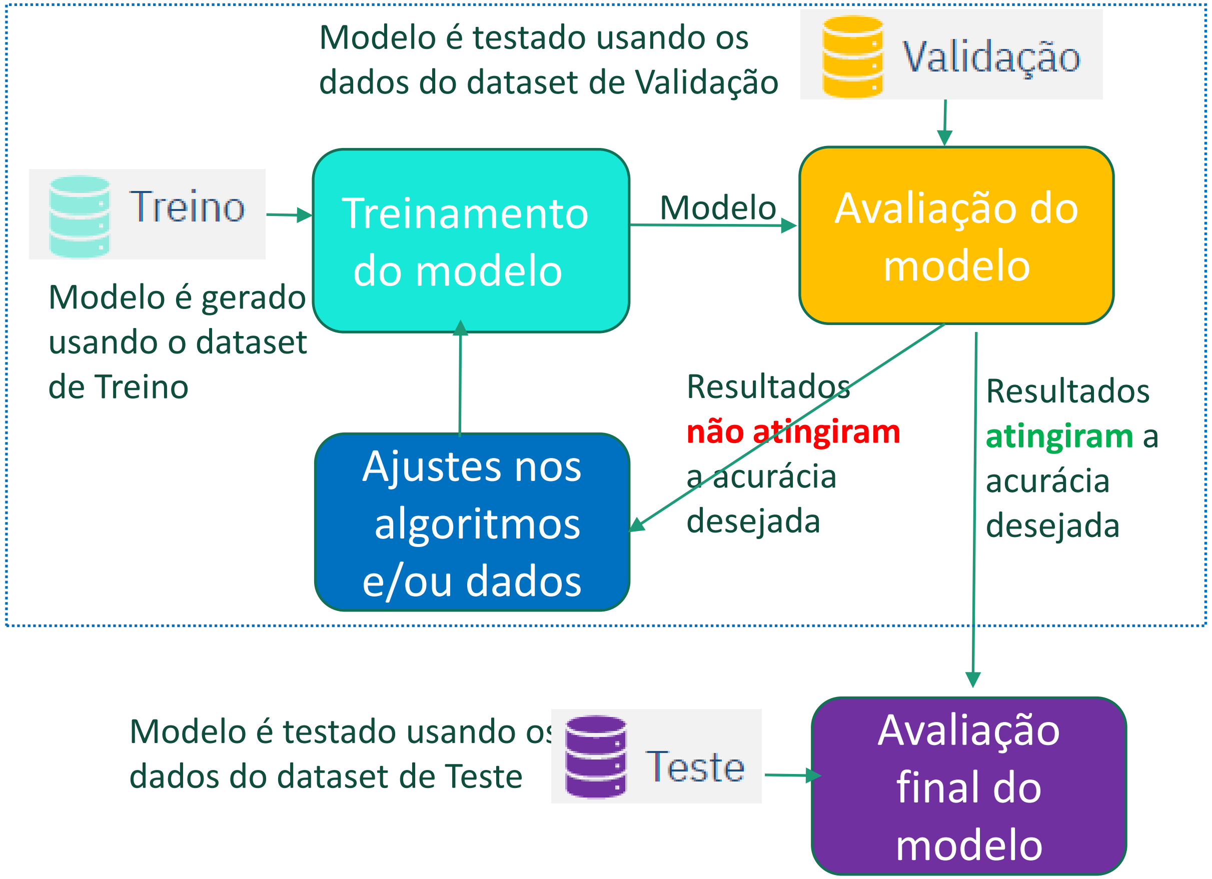
- **Divida o dataset** (harmonicamente, no caso de classes), em conjuntos disjuntos de:
- \- **treino**: usado para o treinamento do modelo;
  \- **validação**: usado para validar o modelo durante o treinamento;
  \- **teste**: usado para verificar a solução, ou seja, a versão final do modelo, após o treinamento .



Fonte da Figura: https://www.v7labs.com/blog/train-validation-test-set#:~:text=Training%20data%20is%20the%20set,after%20completing%20the%20training%20phase.

Treinamento de Redes Neurais

Como os subconjuntos de treino, validação e teste são usados:



$$\text{Acurácia} = \frac{\text{Número de predições corretas}}{\text{Total de predições}}$$

Treinamento de Redes Neurais

- Os **ciclos de treinamento de uma rede** são medidos em **épocas**.
- Uma <u>época</u> **corresponde a passagem de todos os dados do conjunto de treino uma vez pela rede**.
- Para treinar uma rede são necessárias várias épocas.

Rede Perceptron - Algoritmo de Treinamento

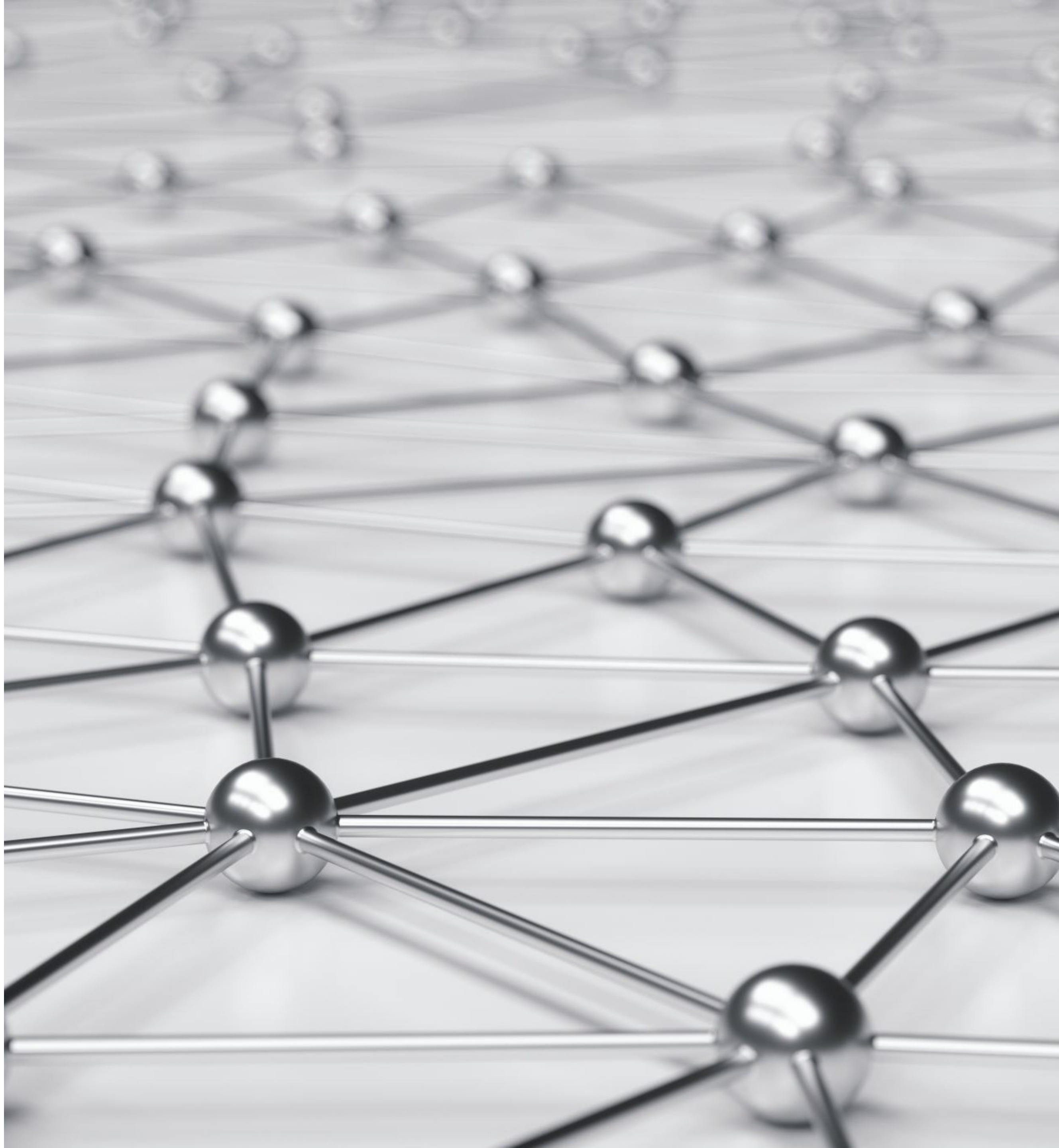
Algoritmo de Treinamento do Perceptron: usa Regra Delta (correção de erro)

- Sendo X = { (dados1;saidaDesejada1); (dados1;saidaDesejada2),...}
- A taxa de aprendizagem η (deve ser positiva).

```
Inicializar todos pesos da rede com zero : wij = 0
Repetir ate encontrar erroGeral=0:
        epocas = epocas +1
                erroGeral = 0
                Para cada neuronio j: 1 a k da rede :
                vj = 0
                Para cada atributo i: 0 a n dos dados de X :
                        vj = vj + xi * wij
                yj = f(vj)
                erroj = dj – yj
                se erroj!=0 entao :
                        delta = η * erroj * xi
                        wij = wj + delta
         erroGeral = erroGeral + abs(erroj)
```

Qual o objetivo do treinamento de uma rede neural?

O que estamos procurando?



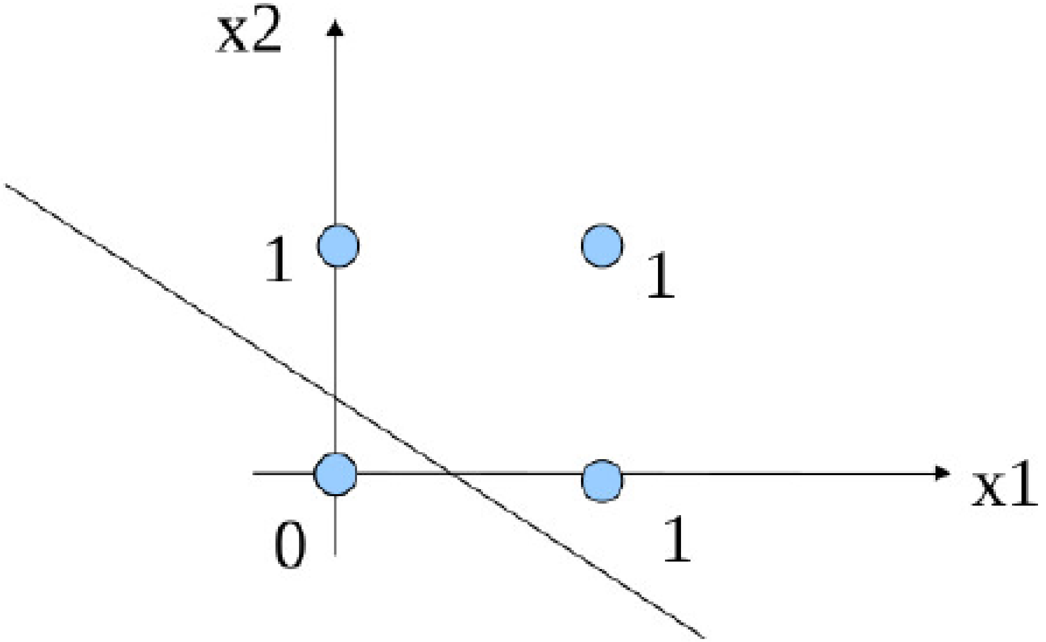
Os **pesos sinápticos**.

São neles que o conhecimento adquirido pela rede fica armazenado.

Rede Perceptron – Limitações

Tabela OR

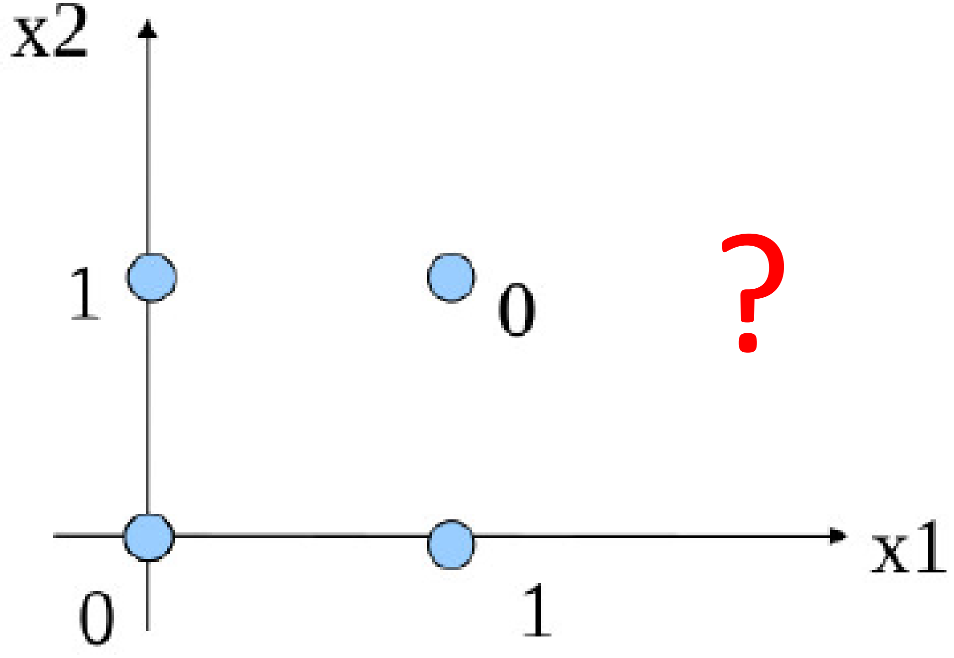
x1	x2	d
0	0	0
0	1	1
1	0	1
1	1	1



Rede Perceptron – Limitações

Tabela XOR

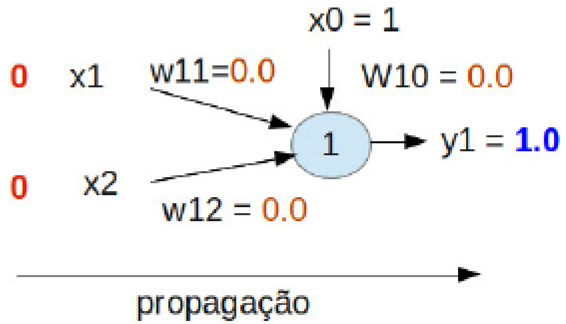
x1	x2	d
0	0	0
0	1	1
1	0	1
1	1	0



Simulando uma Rede Perceptron

- **Pesos** inicializados com zero: $w_{10} = 0, w_{11} = 0, w_{12} = 0$
- limiar: $Q(v_k) = \begin{cases} 1 & se\ v_k \geq 0 \\ 0 & caso\ contrário \end{cases}$, taxa de aprendizagem: $\eta = 1$

x1	x2	d
0	0	0
0	1	1
1	0	1
1	1	1



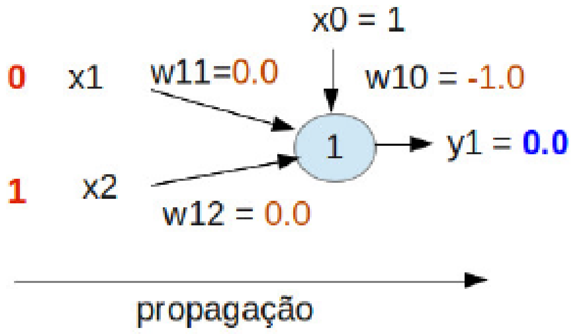
- $v_1 = x_0 \times w_{10} + x_1 \times w_{11} + x_2 \times w_{12}$
- $v_1 = 1 \times 0 + 0 \times 0 + 0 \times 0 = 0$
- $y_1 = Q(v_1) = Q(0) = 1.0$

- Ajuste dos pesos (Aplicando a regra delta)
  - $erro_1 = d_1 - y_1 = 0 - 1.0 = -1.0$
  - $w_{10} = w_{10} + \eta \times erro_1 \times x_0 = 0.0 + 1 \times -1.0 \times 1 = -1.0$
  - $w_{11} = w_{11} + \eta \times erro_1 \times x_1 = 0.0 + 1 \times -1.0 \times 0 = 0.0$
  - $w_{12} = w_{12} + \eta \times erro_1 \times x_2 = 0.0 + 1 \times -1.0 \times 0 = 0.0$

Simulando uma Rede Perceptron

- **Pesos atuais:** $w_{10} = -1.0, w_{11} = 0.0, w_{12} = 0.0$
- limiar: $Q(v_k) = \begin{cases} 1 & se\ v_k \geq 0 \\ 0 & caso\ contrário \end{cases}$, taxa de aprendizagem: $\eta = 1$

x1	x2	d
0	0	0
0	1	1
1	0	1
1	1	1



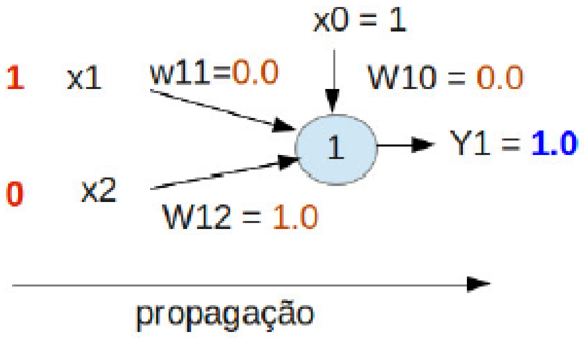
- $v_1 = x_0 \times w_{10} + x_1 \times w_{11} + x_2 \times w_{12}$
- $v_1 = 1 \times -1.0 + 0 \times 0.0 + 1 \times 0.0 = -1.0$
- $y_1 = Q(v_1) = Q(-1.0) = 0.0$

- Ajuste dos pesos (Aplicando a regra delta)
  - $erro_1 = d_1 - y_1 = 1 - 0.0 = 1.0$
  - $w_{10} = w_{10} + \eta \times erro_1 \times x_0 = -1.0 + 1 \times 1.0 \times 1 = 0.0$
  - $w_{11} = w_{11} + \eta \times erro_1 \times x_1 = 0.0 + 1 \times 1.0 \times 0 = 0.0$
  - $w_{12} = w_{12} + \eta \times erro_1 \times x_2 = 0.0 + 1 \times 1.0 \times 1 = 1.0$

Simulando uma Rede Perceptron

- **Pesos atuais:** $w_{10} = 0.0, w_{11} = 0.0, w_{12} = 1.0$
- limiar: $Q(v_k) = \begin{cases} 1 & se\, v_k \geq 0 \\ 0 & caso\ contrário \end{cases}$, taxa de aprendizagem: $\eta = 1$

x1	x2	d
0	0	0
0	1	1
1	0	1
1	1	1



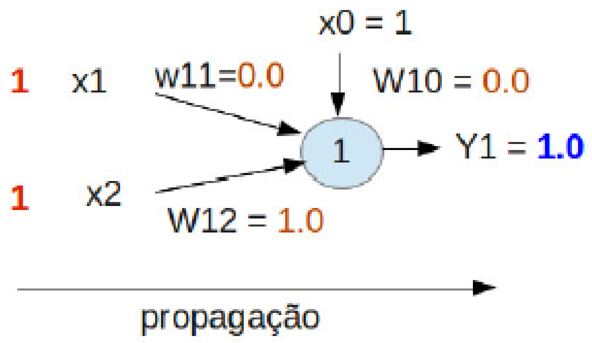
- $v_1 = x_0 \times w_{10} + x_1 \times w_{11} + x_2 \times w_{12}$
- $v_1 = 1 \times 0.0 + 1 \times 0.0 + 0 \times 1.0 = 0.0$
- $y_1 = Q(v_1) = Q(0.0) = 1.0$

- Ajuste dos pesos (Aplicando a regra delta)
  - $erro_1 = d_1 - y_1 = 1 - 1.0 = 0.0$
    - quando o erro é zero, não há ajuste de pesos.

Simulando uma Rede Perceptron

- **Pesos atuais:** $w_{10} = 0.0, w_{11} = 0.0, w_{12} = 1.0$
- limiar: $Q(v_k) = \begin{cases} 1 & se\ v_k \geq 0 \\ 0 & caso\ contr\acute{a}rio \end{cases}$, taxa de aprendizagem: $\eta = 1$

x1	x2	d
0	0	0
0	1	1
1	0	1
1	1	1



- $v_1 = x_0 \times w_{10} + x_1 \times w_{11} + x_2 \times w_{12}$
- $v_1 = 1 \times 0.0 + 1 \times 0.0 + 1 \times 1.0 = 1.0$
- $y_1 = Q(v_1) = Q(1.0) = 1.0$

- Ajuste dos pesos (Aplicando a regra delta)
  - $erro_1 = d_1 - y_1 = 1 - 1.0 = 0.0$
    - quando o erro é zero, não há ajuste de pesos.

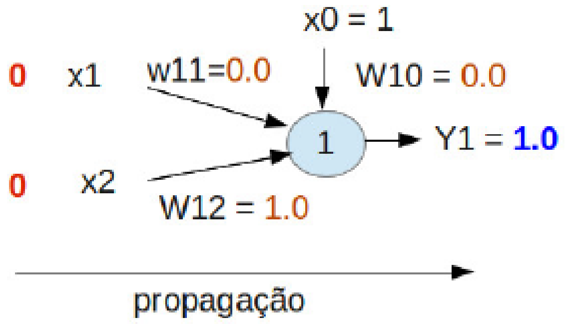
Simulando uma Rede Perceptron

- Erro Acumulado para época 1 :
  - Erro Geral:
    - $= abs(erroEntrada1) + abs(erroEntrada2) + abs(erroEntrada3) + abs(erroEntrada4) =$
    - $= abs(-1.0) + abs(1.0) + abs(0.0) + abs(0.0)$
    - $= 2.0$
- Como ainda há erro, o algoritmo prossegue ...

Simulando uma Rede Perceptron

- **Pesos atuais:** $w_{10} = 0.0, w_{11} = 0.0, w_{12} = 1.0$
- limiar: $Q(v_k) = \begin{cases} 1 & se\, v_k \geq 0 \\ 0 & caso\, contr\acute{a}rio \end{cases}$, taxa de aprendizagem: $\eta = 1$

x1	x2	d
0	0	0
0	1	1
1	0	1
1	1	1



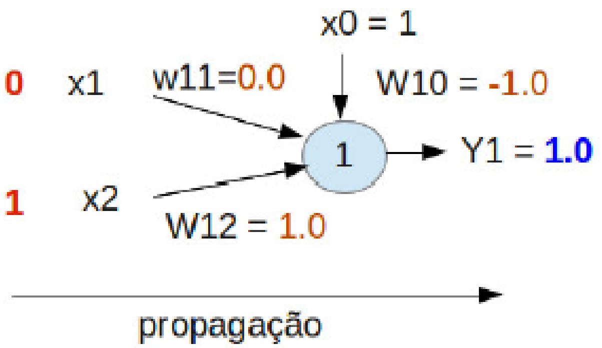
- $v_1 = x_0 \times w_{10} + x_1 \times w_{11} + x_2 \times w_{12}$
- $v_1 = 0 \times 0.0 + 0 \times 1.0 + 1 \times 0.0 = 0.0$
- $y_1 = Q(v_1) = Q(0.0) = 1.0$

- Ajuste dos pesos (Aplicando a regra delta)
  - $erro_1 = d_1 - y_1 = 0 - 1.0 = -1.0$
  - $w_{10} = w_{10} + \eta \times erro_1 \times x_0 = 0.0 + 1 \times -1.0 \times 1 = -1.0$
  - $w_{11} = w_{11} + \eta \times erro_1 \times x_1 = 0.0 + 1 \times -1.0 \times 0 = 0.0$
  - $w_{12} = w_{12} + \eta \times erro_1 \times x_2 = 1.0 + 1 \times -1.0 \times 0 = 1.0$

Simulando uma Rede Perceptron

- **Pesos atuais:** $w_{10} = -1.0, w_{11} = 0.0, w_{12} = 1.0$
- limiar: $Q(v_k) = \begin{cases} 1 & se\ v_k \geq 0 \\ 0 & caso\ contrário \end{cases}$, taxa de aprendizagem: $\eta = 1$

x1	x2	d
0	0	0
0	1	1
1	0	1
1	1	1



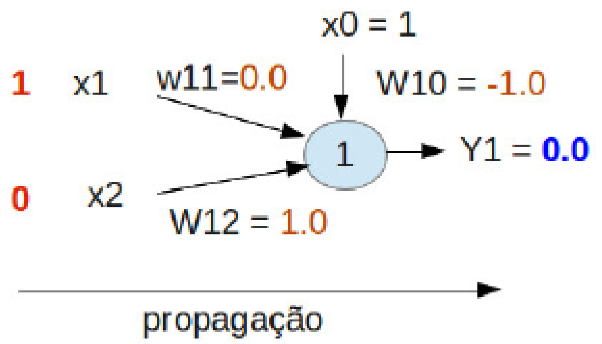
- $v_1 = x_0 \times w_{10} + x_1 \times w_{11} + x_2 \times w_{12}$
- $v_1 = 1 \times -1.0 + 0 \times 0.0 + 1 \times 1.0 = 0.0$
- $y_1 = Q(v_1) = Q(0.0) = 1.0$

- Ajuste dos pesos (Aplicando a regra delta)
  - $erro_1 = d_1 - y_1 = 1 - 1.0 = 0.0$
    - quando o erro é zero, não há ajuste de pesos.

Simulando uma Rede Perceptron

- **Pesos atuais:** $w_{10} = -1.0, w_{11} = 0.0, w_{12} = 1.0$
- limiar: $Q(v_k) = \begin{cases} 1 & se\ v_k \geq 0 \\ 0 & caso\ contr\acute{a}rio \end{cases}$, taxa de aprendizagem: $\eta = 1$

x1	x2	d
0	0	0
0	1	1
1	0	1
1	1	1



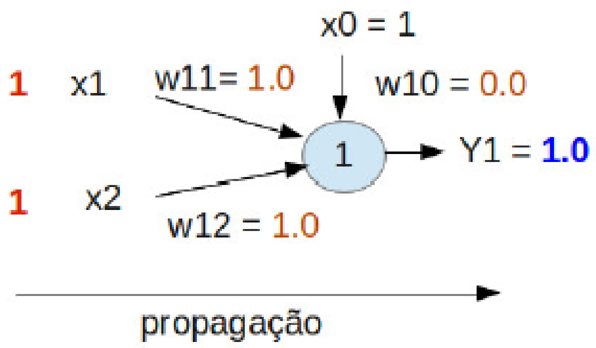
- $v_1 = x_0 \times w_{10} + x_1 \times w_{11} + x_2 \times w_{12}$
- $v_1 = 1 \times -1.0 + 1 \times 0.0 + 0 \times 1.0 = -1.0$
- $y_1 = Q(v_1) = Q(-1.0) = 0.0$

- Ajuste dos pesos (Aplicando a regra delta)
  - $erro_1 = d_1 - y_1 = 1 - 0.0 = 1.0$
  - $w_{10} = w_{10} + \eta \times erro_1 \times x_0 = -1.0 + 1 \times 1.0 \times 1 = 0.0$
  - $w_{11} = w_{11} + \eta \times erro_1 \times x_1 = 0.0 + 1 \times 1.0 \times 1 = 1.0$
  - $w_{12} = w_{12} + \eta \times erro_1 \times x_2 = 1.0 + 1 \times 1.0 \times 0 = 1.0$

Simulando uma Rede Perceptron

- **Pesos atuais:** $w_{10} = 0.0, w_{11} = 1.0, w_{12} = 1.0$
- limiar: $Q(v_k) = \begin{cases} 1 & se\ v_k \geq 0 \\ 0 & caso\ contrário \end{cases}$, taxa de aprendizagem: $\eta = 1$

x1	x2	d
0	0	0
0	1	1
1	0	1
1	1	1



- $v_1 = x_0 \times w_{10} + x_1 \times w_{11} + x_2 \times w_{12}$
- $v_1 = 1 \times 0.0 + 1 \times 1.0 + 1 \times 1.0 = 2.0$
- $y_1 = Q(v_1) = Q(2.0) = 1.0$

- Ajuste dos pesos (Aplicando a regra delta)
  - $erro_1 = d_1 - y_1 = 1 - 1.0 = 0.0$
    - quando o erro é zero, não há ajuste de pesos.

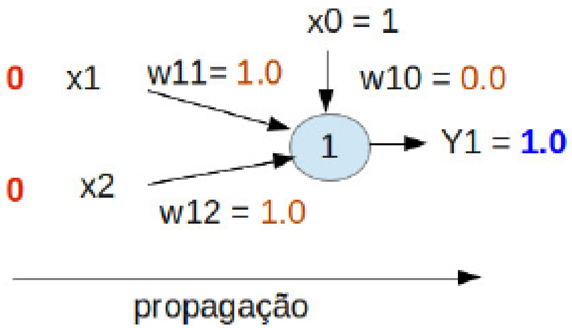
Simulando uma Rede Perceptron

- Erro Acumulado para época 2 :
    - Erro Geral:
        - $= abs(erroEntrada1) + abs(erroEntrada2) + abs(erroEntrada3) + abs(erroEntrada4) =$
        - $= abs(-1.0) + abs(0.0) + abs(1.0) + abs(0.0)$
        - $= 2.0$
- Como ainda há erro, o algoritmo prossegue ...

Simulando uma Rede Perceptron

- **Pesos atuais:** $w_{10} = 0.0, w_{11} = 1.0, w_{12} = 1.0$
- limiar: $Q(v_k) = \begin{cases} 1 & se\ v_k \geq 0 \\ 0 & caso\ contr\acute{a}rio \end{cases}$, taxa de aprendizagem: $\eta = 1$

x1	x2	d
0	0	0
0	1	1
1	0	1
1	1	1



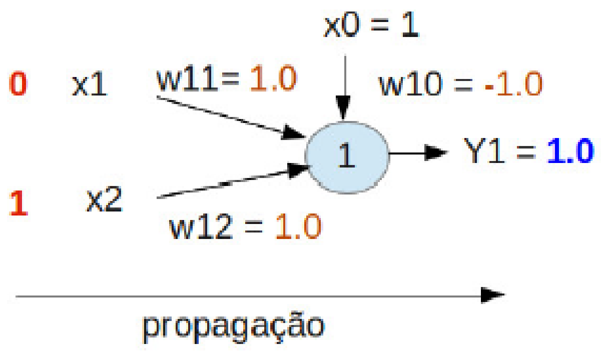
- $v_1 = x_0 \times w_{10} + x_1 \times w_{11} + x_2 \times w_{12}$
- $v_1 = 1 \times 0.0 + 0 \times 1.0 + 0 \times 1.0 = 0.0$
- $y_1 = Q(v_1) = Q(0.0) = 1.0$

- Ajuste dos pesos (Aplicando a regra delta)
  - $erro_1 = d_1 - y_1 = 0 - 1.0 = -1.0$
  - $w_{10} = w_{10} + \eta \times erro_1 \times x_0 = 0.0 + 1 \times -1.0 \times 1 = -1.0$
  - $w_{11} = w_{11} + \eta \times erro_1 \times x_1 = 1.0 + 1 \times -1.0 \times 0 = 1.0$
  - $w_{12} = w_{12} + \eta \times erro_1 \times x_2 = 1.0 + 1 \times -1.0 \times 0 = 1.0$

Simulando uma Rede Perceptron

- **Pesos atuais:** $w_{10} = -1.0, w_{11} = 1.0, w_{12} = 1.0$
- limiar: $Q(v_k) = \begin{cases} 1 & se\ v_k \geq 0 \\ 0 & caso\ contrário \end{cases}$, taxa de aprendizagem: $\eta = 1$

x1	x2	d
0	0	0
0	1	1
1	0	1
1	1	1



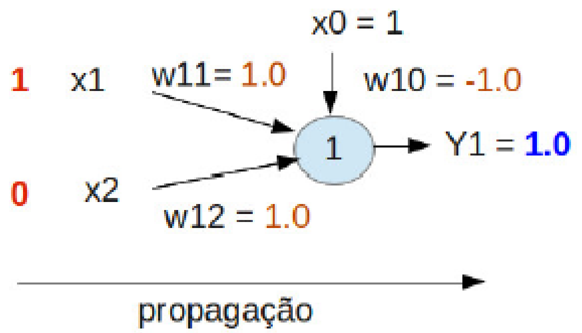
- $v_1 = x_0 \times w_{10} + x_1 \times w_{11} + x_2 \times w_{12}$
- $v_1 = 1 \times -1.0 + 0 \times 1.0 + 1 \times 1.0 = 0.0$
- $y_1 = Q(v_1) = Q(0.0) = 1.0$

- Ajuste dos pesos (Aplicando a regra delta)
  - $erro_1 = d_1 - y_1 = 1 - 1.0 = 0.0$
    - quando o erro é zero, não há ajuste de pesos.

Simulando uma Rede Perceptron

- **Pesos atuais:** $w_{10} = -1.0, w_{11} = 1.0, w_{12} = 1.0$
- limiar: $Q(v_k) = \begin{cases} 1 & se\ v_k \geq 0 \\ 0 & caso\ contrário \end{cases}$, taxa de aprendizagem: $\eta = 1$

x1	x2	d
0	0	0
0	1	1
1	0	1
1	1	1



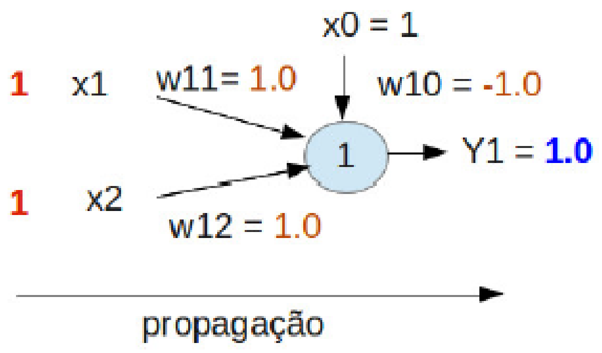
- $v_1 = x_0 \times w_{10} + x_1 \times w_{11} + x_2 \times w_{12}$
- $v_1 = 1 \times -1.0 + 1 \times 1.0 + 0 \times 1.0 = 0.0$
- $y_1 = Q(v_1) = Q(0.0) = 1.0$

- Ajuste dos pesos (Aplicando a regra delta)
  - $erro_1 = d_1 - y_1 = 1 - 1.0 = 0.0$
    - quando o erro é zero, não há ajuste de pesos.

Simulando uma Rede Perceptron

- **Pesos atuais:** $w_{10} = -1.0, w_{11} = 1.0, w_{12} = 1.0$
- limiar: $Q(v_k) = \begin{cases} 1 & se\ v_k \geq 0 \\ 0 & caso\ contrário \end{cases}$, taxa de aprendizagem: $\eta = 1$

x1	x2	d
0	0	0
0	1	1
1	0	1
1	1	1



- $v_1 = x_0 \times w_{10} + x_1 \times w_{11} + x_2 \times w_{12}$
- $v_1 = 1 \times -1.0 + 1 \times 1.0 + 1 \times 1.0 = 1.0$
- $y_1 = Q(v_1) = Q(1.0) = 1.0$

- Ajuste dos pesos (Aplicando a regra delta)
  - $erro_1 = d_1 - y_1 = 1 - 1.0 = 0.0$
    - quando o erro é zero, não há ajuste de pesos.

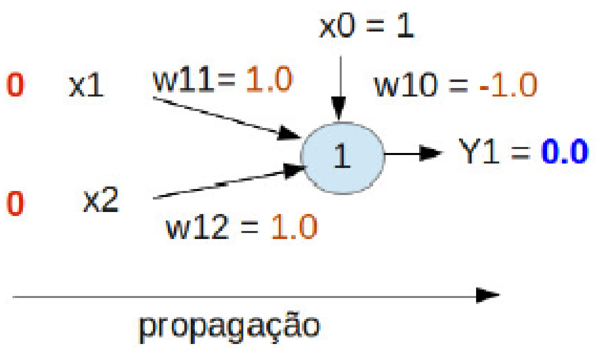
Simulando uma Rede Perceptron

- Erro Acumulado para época 3 :
  - Erro Geral:
    - $= abs(erroEntrada1) + abs(erroEntrada2) + abs(erroEntrada3) + abs(erroEntrada4) =$
    - $= abs(-1.0) + abs(0.0) + abs(0.0) + abs(0.0)$
    - $= 1.0$
- Como ainda há erro, o algoritmo prossegue ...

Simulando uma Rede Perceptron

- **Pesos atuais:** $w_{10} = -1.0, w_{11} = 1.0, w_{12} = 1.0$
- limiar: $Q(v_k) = \begin{cases} 1 & se\ v_k \geq 0 \\ 0 & caso\ contrário \end{cases}$, taxa de aprendizagem: $\eta = 1$

x1	x2	d
0	0	0
0	1	1
1	0	1
1	1	1



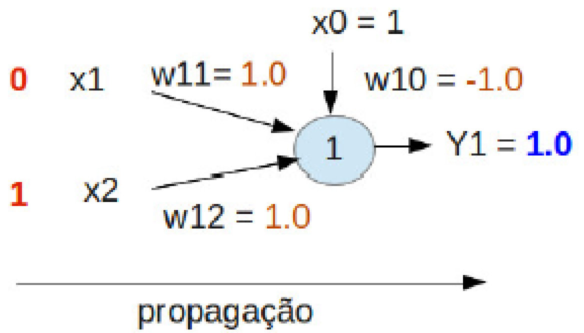
- $v_1 = x_0 \times w_{10} + x_1 \times w_{11} + x_2 \times w_{12}$
- $v_1 = 1 \times -1.0 + 0 \times 1.0 + 0 \times 1.0 = -1.0$
- $y_1 = Q(v_1) = Q(-1.0) = 0.0$

- Ajuste dos pesos (Aplicando a regra delta)
  - $erro_1 = d_1 - y_1 = 0 - 0.0 = 0.0$
    - quando o erro é zero, não há ajuste de pesos.

Simulando uma Rede Perceptron

- **Pesos atuais:** $w_{10} = -1.0, w_{11} = 1.0, w_{12} = 1.0$
- limiar: $Q(v_k) = \begin{cases} 1 & se\ v_k \geq 0 \\ 0 & caso\ contrário \end{cases}$, taxa de aprendizagem: $\eta = 1$

x1	x2	d
0	0	0
0	1	1
1	0	1
1	1	1



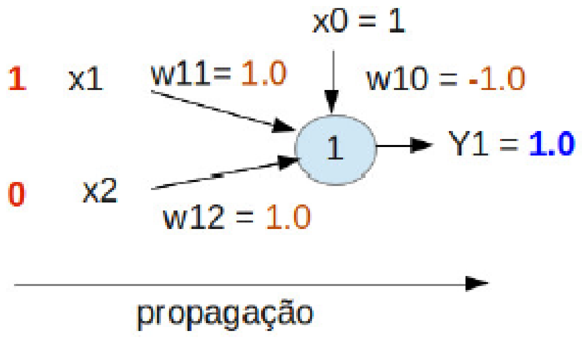
- $v_1 = x_0 \times w_{10} + x_1 \times w_{11} + x_2 \times w_{12}$
- $v_1 = 1 \times -1.0 + 0 \times 1.0 + 1 \times 1.0 = 0.0$
- $y_1 = Q(v_1) = Q(0.0) = 1.0$

- Ajuste dos pesos (Aplicando a regra delta)
  - $erro_1 = d_1 - y_1 = 1 - 1.0 = 0.0$
    - quando o erro é zero, não há ajuste de pesos.

Simulando uma Rede Perceptron

- **Pesos atuais:** $w_{10} = -1.0, w_{11} = 1.0, w_{12} = 1.0$
- limiar: $Q(v_k) = \begin{cases} 1 & se\ v_k \geq 0 \\ 0 & caso\ contr\acute{a}rio \end{cases}$, taxa de aprendizagem: $\eta = 1$

x1	x2	d
0	0	0
0	1	1
1	0	1
1	1	1



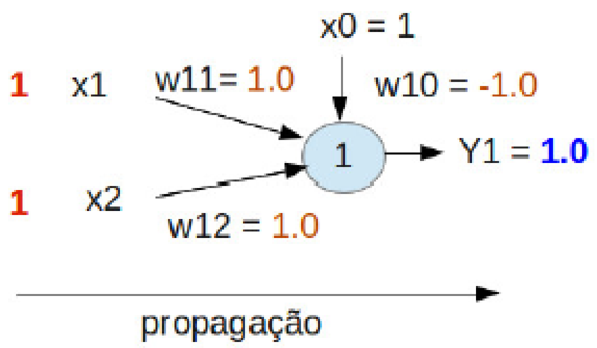
- $v_1 = x_0 \times w_{10} + x_1 \times w_{11} + x_2 \times w_{12}$
- $v_1 = 1 \times -1.0 + 1 \times 1.0 + 0 \times 1.0 = 0.0$
- $y_1 = Q(v_1) = Q(0.0) = 1.0$

- Ajuste dos pesos (Aplicando a regra delta)
  - $erro_1 = d_1 - y_1 = 1 - 1.0 = 0.0$
    - quando o erro é zero, não há ajuste de pesos.

Simulando uma Rede Perceptron

- **Pesos atuais:** $w_{10} = -1.0, w_{11} = 1.0, w_{12} = 1.0$
- limiar: $Q(v_k) = \begin{cases} 1 & se\ v_k \geq 0 \\ 0 & caso\ contr\acute{a}rio \end{cases}$, taxa de aprendizagem: $\eta = 1$

x1	x2	d
0	0	0
0	1	1
1	0	1
1	1	1



- $v_1 = x_0 \times w_{10} + x_1 \times w_{11} + x_2 \times w_{12}$
- $v_1 = 1 \times -1.0 + 1 \times 1.0 + 1 \times 1.0 = 1.0$
- $y_1 = Q(v_1) = Q(1.0) = 1.0$

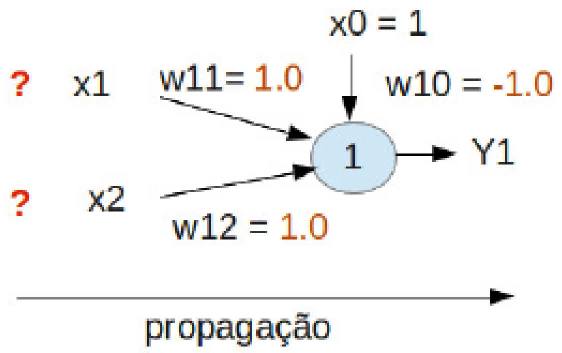
- Ajuste dos pesos (Aplicando a regra delta)
  - $erro_1 = d_1 - y_1 = 1 - 1.0 = 0.0$
    - quando o erro é zero, não há ajuste de pesos.

Simulando uma Rede Perceptron

- Erro Acumulado para época 4 :
  - Erro Geral:
    - $= abs(erroEntrada1) + abs(erroEntrada2) + abs(erroEntrada3) + abs(erroEntrada4) =$
    - $= abs(0.0) + abs(0.0) + abs(0.0) + abs(0.0)$
    - $= 0.0$
  - O algoritmo pára.
- Resultado:
  - O algoritmo encontrou os pesos adequados: $w_{10} = -1.0, w_{11} = 1.0, w_{12} = 1.0$

Simulando uma Rede Perceptron
Fase de Generalização

- A rede está pronta para ser testada (ou usada).
  1. Carrega-se, na topologia da rede, os pesos encontrados na fase de treinamento.



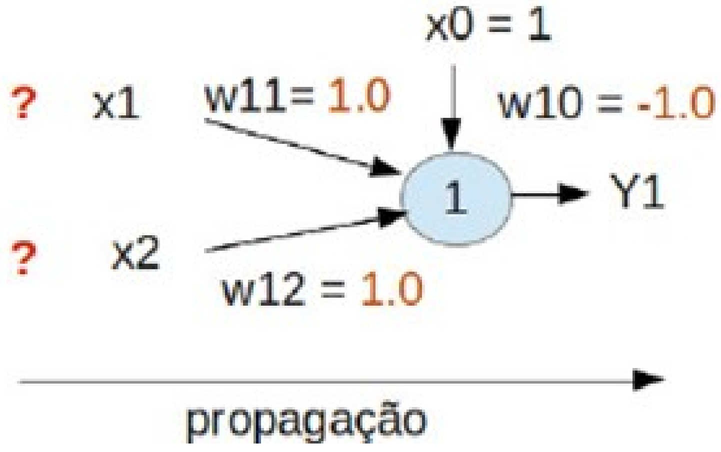
Modelo = Corresponde a uma instancia do Algoritmo de IA gerada a partir dos dados de treinamento

  2. Esta fase implementa apenas a propagação. Informa-se novos valores para as entradas e a rede gera a saída correspondente.

Simulando uma Rede Perceptron
Fase de Generalização

Testando o modelo:

x1	x2
0	0
0	1
1	0
1	1



$$v_j = \sum_{i=0}^{n} (x_i \times w_{ij})$$

$$y_j = f(v_j)$$

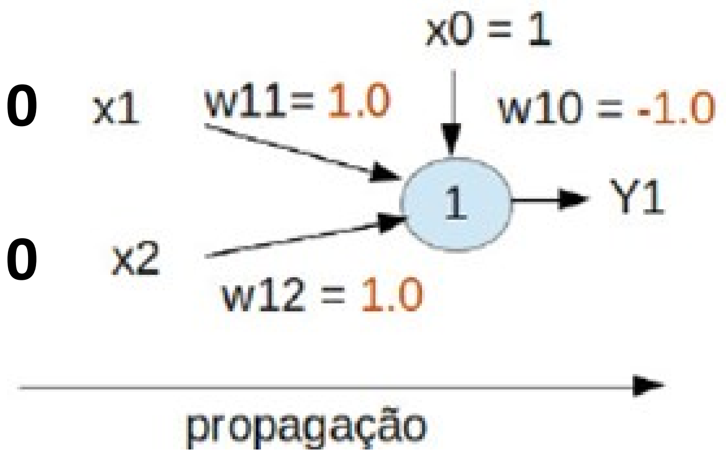
Função de ativação - Limiar

$$f(v_j) = \begin{cases} 1 \; se \; v_j \geq 0 \\ 0 \; se \; v_j < 0 \end{cases}$$

Simulando uma Rede Perceptron
Fase de Generalização

Testando o modelo:

x1	x2
0	0
0	1
1	0
1	1



v1 = 1 x -1.0 +
0 x 1.0
0 x 1.0
= -1
Y1 = Q(-1) = **0**

$$v_j = \sum_{i=0}^{n} (x_i \times w_{ij})$$

$$y_j = f(v_j)$$

Função de ativação - Limiar

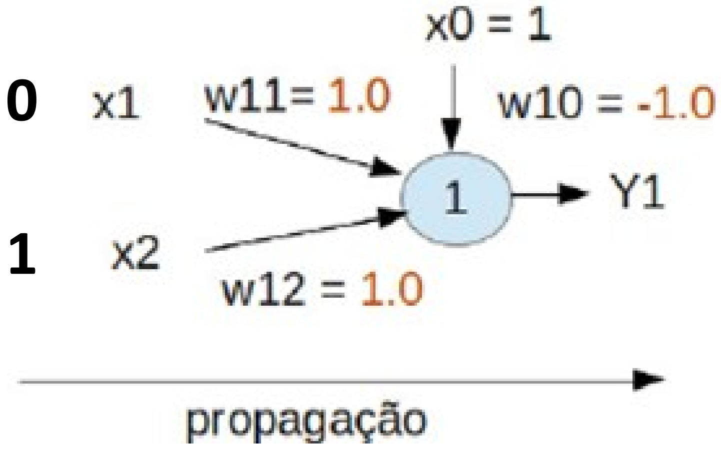
$$f(v_j) = \begin{cases} 1 \; se \; v_j \geq 0 \\ 0 \; se \; v_j < 0 \end{cases}$$

Simulando uma Rede Perceptron

Fase de Generalização

Testando o modelo:

x1	x2
0	0
0	1
1	0
1	1



v1 = 1 x -1.0 +
0 x 1.0
1 x 1.0
= 0

Y1 = Q(0) = **1**

$$v_j = \sum_{i=0}^{n} (x_i \times w_{ij})$$

$$y_j = f(v_j)$$

Função de ativação - Limiar

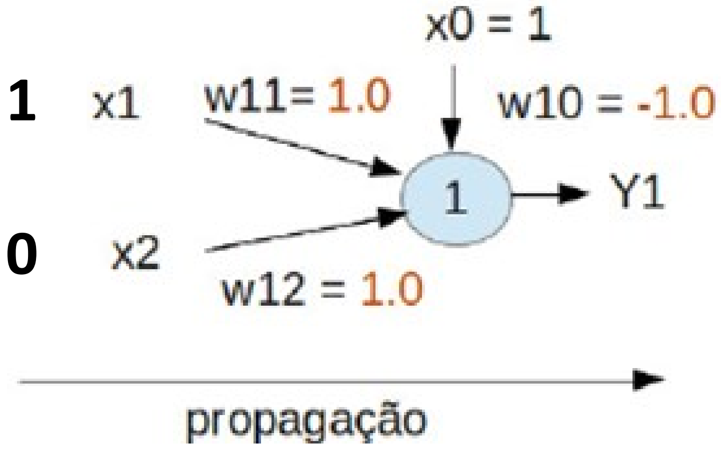
$$f(v_j) = \begin{cases} 1 \ se \ v_j \geq 0 \\ 0 \ se \ v_j < 0 \end{cases}$$

Simulando uma Rede Perceptron

Fase de Generalização

Testando o modelo:

x1	x2
0	0
0	1
1	0
1	1



v1 = 1 x -1.0 +
1 x 1.0
0 x 1.0
= 0

Y1 = Q(0) = **1**

$$v_j = \sum_{i=0}^{n} (x_i \times w_{ij})$$

$$y_j = f(v_j)$$

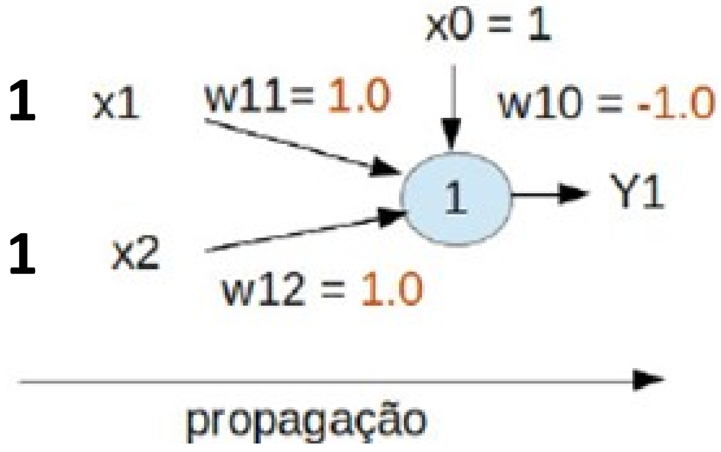
Função de ativação - Limiar

$$f(v_j) = \begin{cases} 1 \; se \; v_j \geq 0 \\ 0 \; se \; v_j < 0 \end{cases}$$

Simulando uma Rede Perceptron
Fase de Generalização

Testando o modelo:

x1	x2
0	0
0	1
1	0
1	1



v1 = 1 x -1.0 +
1 x 1.0
1 x 1.0
= 1

Y1 = Q(1) = **1**

$$v_j = \sum_{i=0}^{n} (x_i \times w_{ij})$$

$$y_j = f(v_j)$$

Função de ativação - Limiar

$$f(v_j) = \begin{cases} 1 \; se \; v_j \geq 0 \\ 0 \; se \; v_j < 0 \end{cases}$$

Exercicios

- **Atividade 1**: Implemente em Python o algoritmo de treinamento da rede Perceptron para as tabelas:
  - OR
  - AND
  - XOR

Exercicios

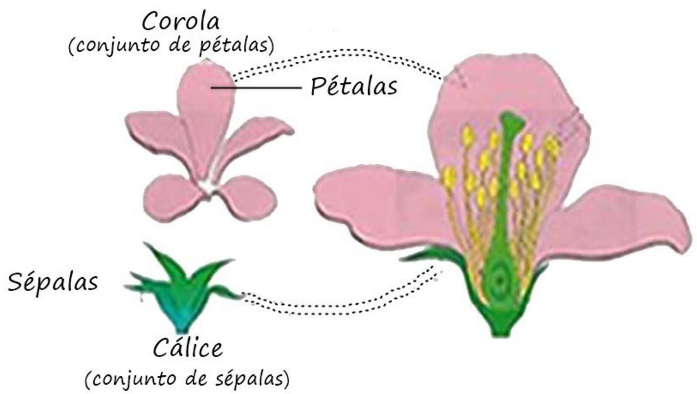
- **Atividade 2**: Considere os dados da tabela, selecione os atributos mais adequados, faca o pre-processamento. E, a seguir, use uma rede perceptron para classificar os clientes. Use a mesma topologia da atividade anterior.

Cliente	*Renda*	*Divida*	*Classe*
101	2800	550	bom
102	1300	500	mau
103	1400	80	bom
104	500	200	mau
105	1100	270	mau
106	1800	450	bom
107	2400	650	bom
108	1950	600	bom
109	450	70	mau
110	2750	730	bom
111	850	90	mau
112	1300	200	mau
113	2100	750	bom
114	900	300	mau
115	2700	250	bom
116	1600	500	mau
117	1900	150	bom
118	2500	800	bom
119	1600	700	mau
120	2300	500	bom
121	2100	250	bom

Exercicios

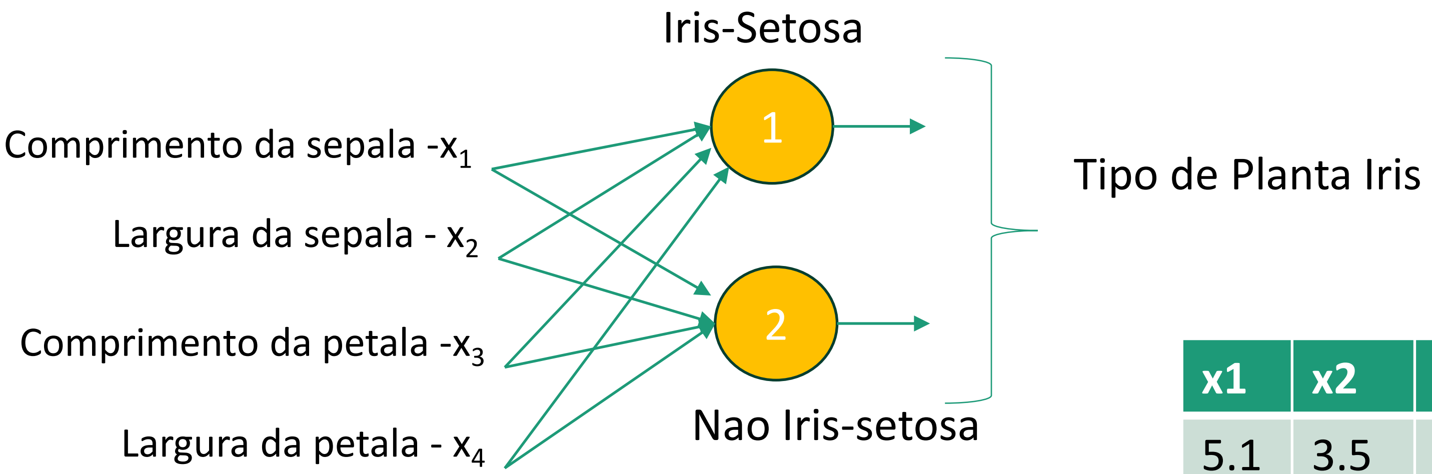
- **Atividade 3**: Implemente em Python o algoritmo de treinamento da rede Perceptron como classificador binário para Planta Iris. Calcule também a acurácia.

Planta Iris



Topologia para 2 classes

Classificador Binario Neural



Topologia: 4 x 2

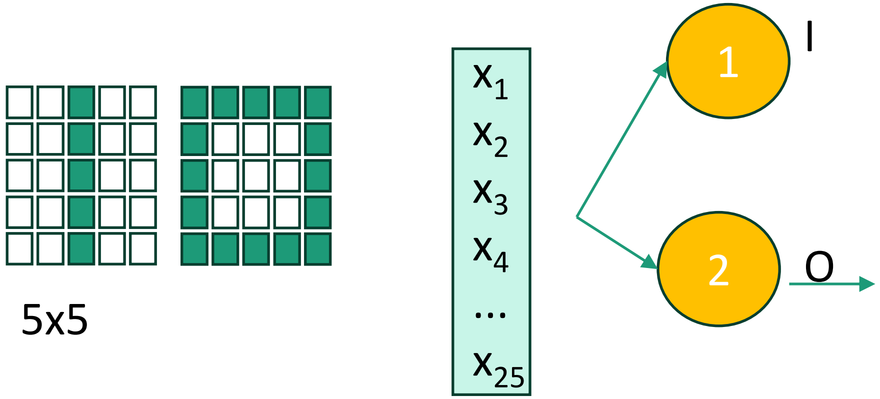
Camada de entrada(dados): 4

Camada de saida (neuronios): 2

x1	x2	x3	x4	d1	d2	
5.1	3.5	1.4	0.2	1	0	Iris-Setosa
7	3.2	4.7	1.4	0	1	Nao Iris-Setosa
…						

Exercicios

- **Atividade 4**: Implemente em Python o algoritmo de treinamento da rede Perceptron para reconhecer as letras . Teste usando letras com ruido.



Atributos dos objetos

Topologia: 25 x 2