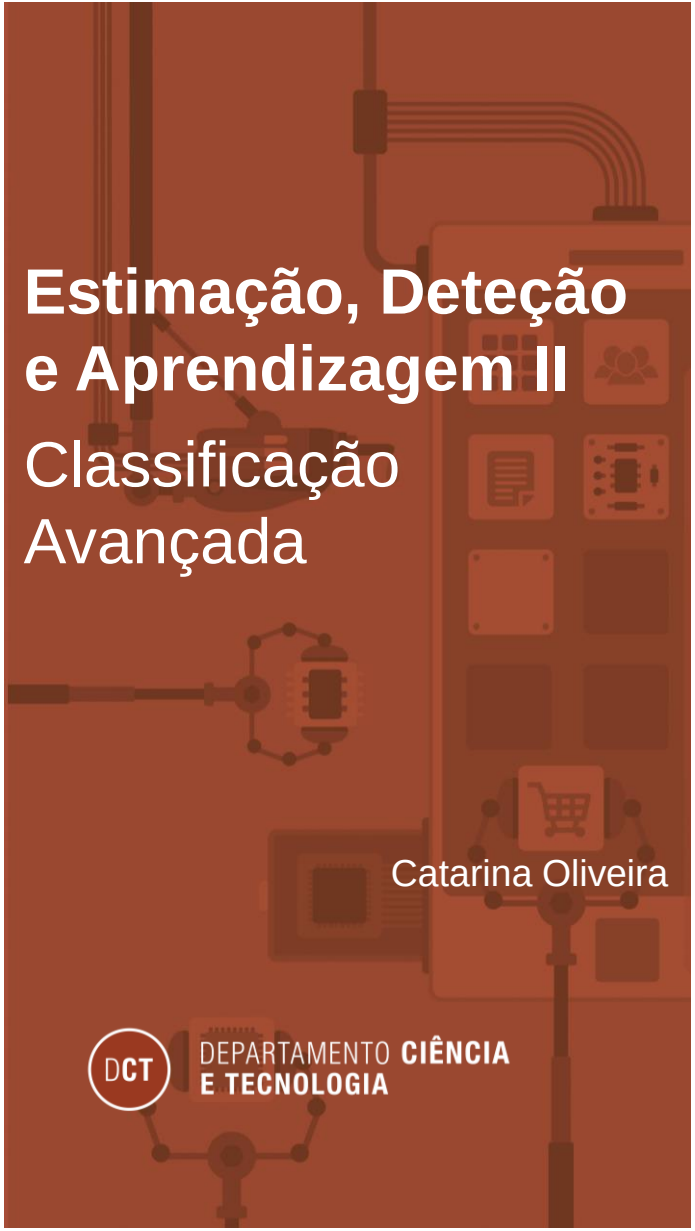


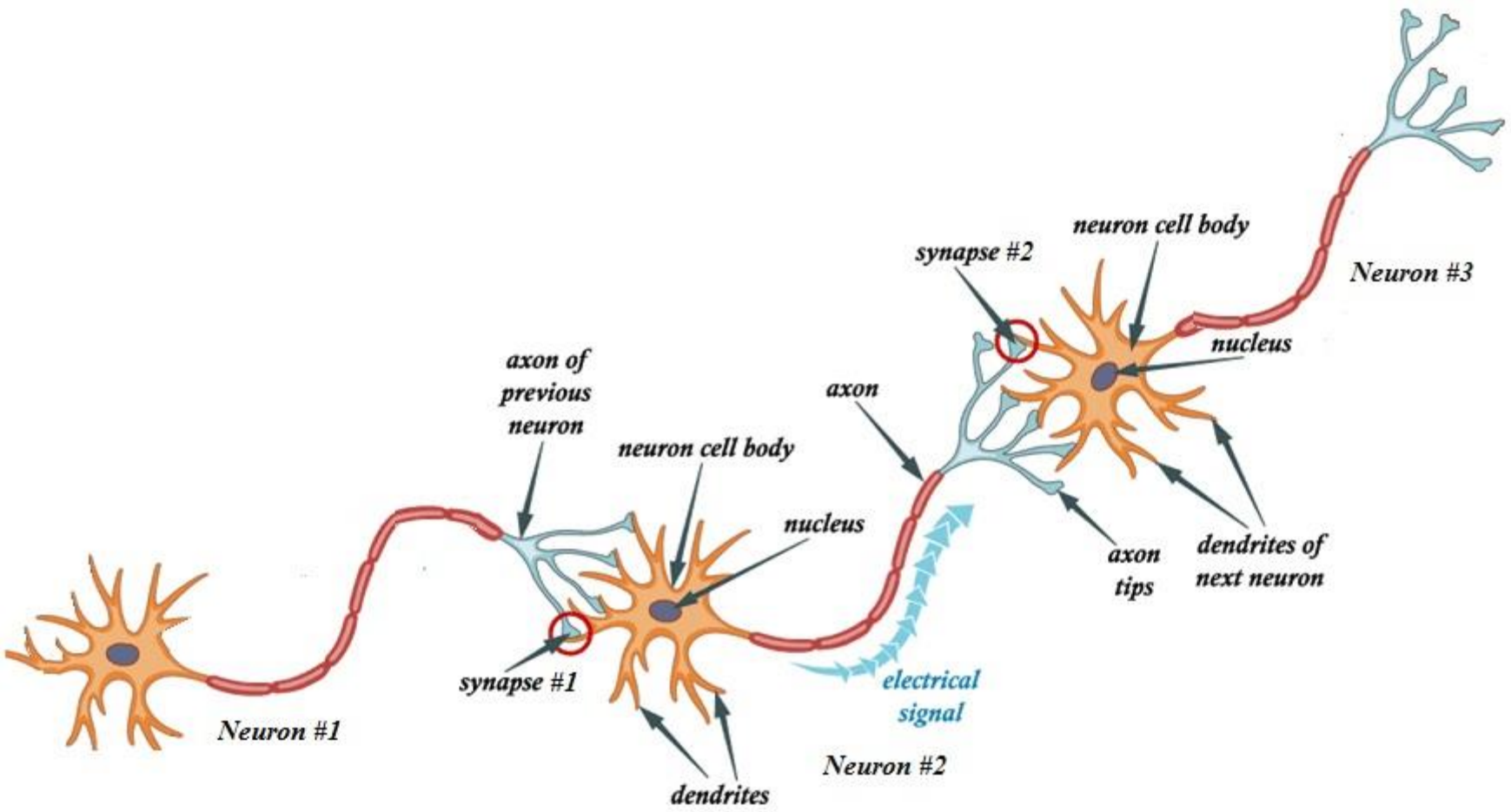
UNIVERSIDADE PORTUCALENSE

CONTEÚDO

1. Redes neuronais
2. Support Vector Machines
3. Vizinho mais próximo
4. Aprendizagem semi-supervisionada
5. Aprendizagem por transferência

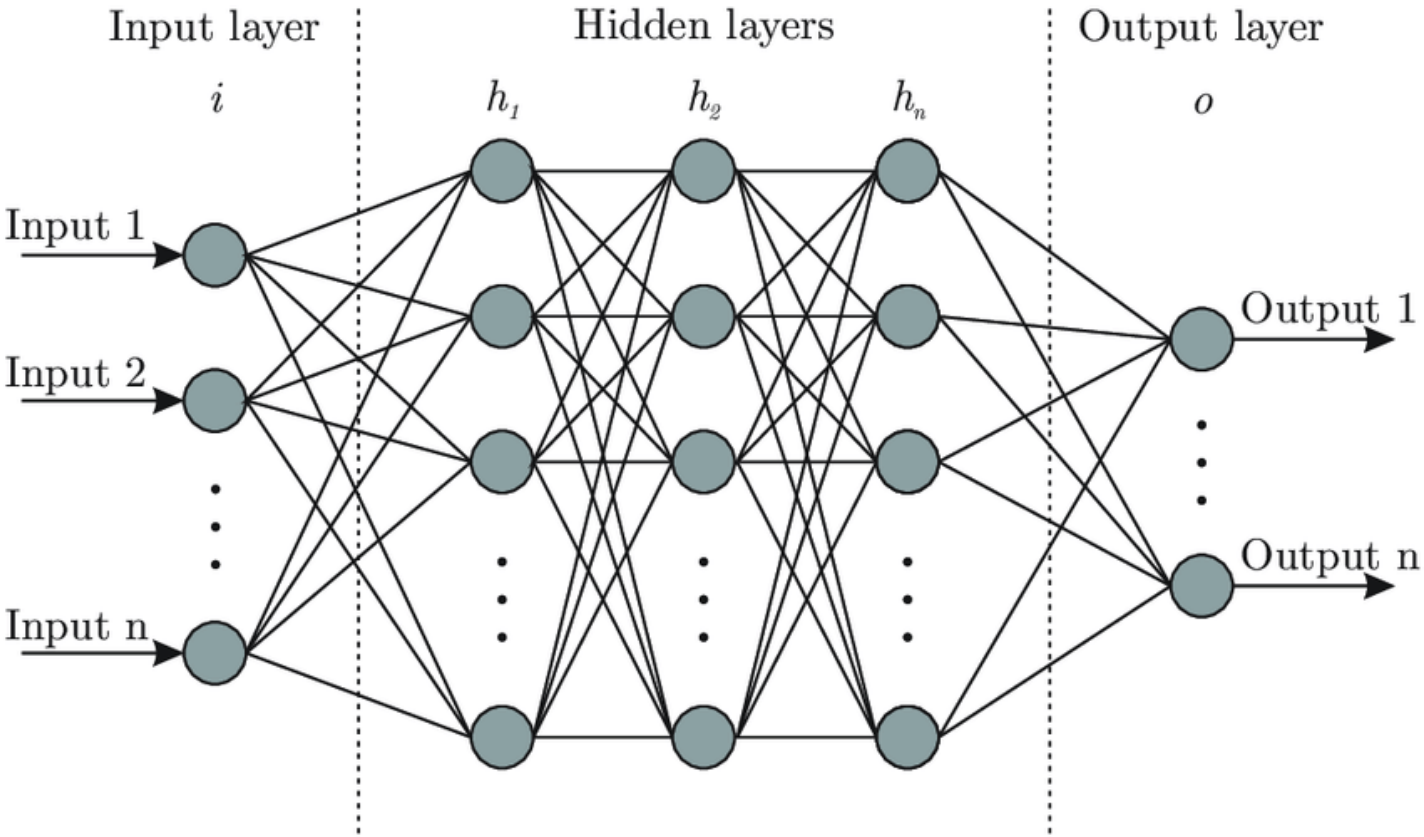
Redes Neuronais
Neural Networks

Redes neuronais: contexto biológico



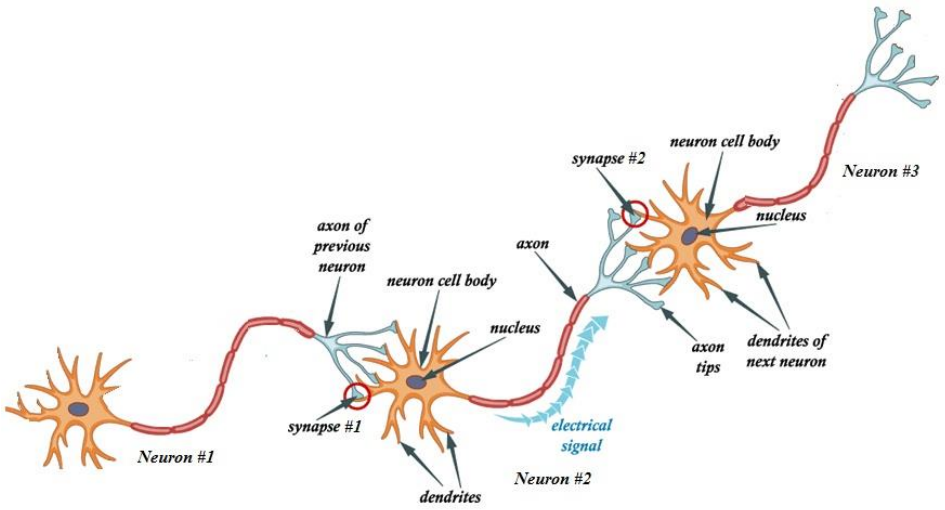
https://www.chegg.com/homework-help/questions-and-answers/let-s-put-together-review-steps-action-potential-events-synapses-fill-following-blanks-tra-q41393959

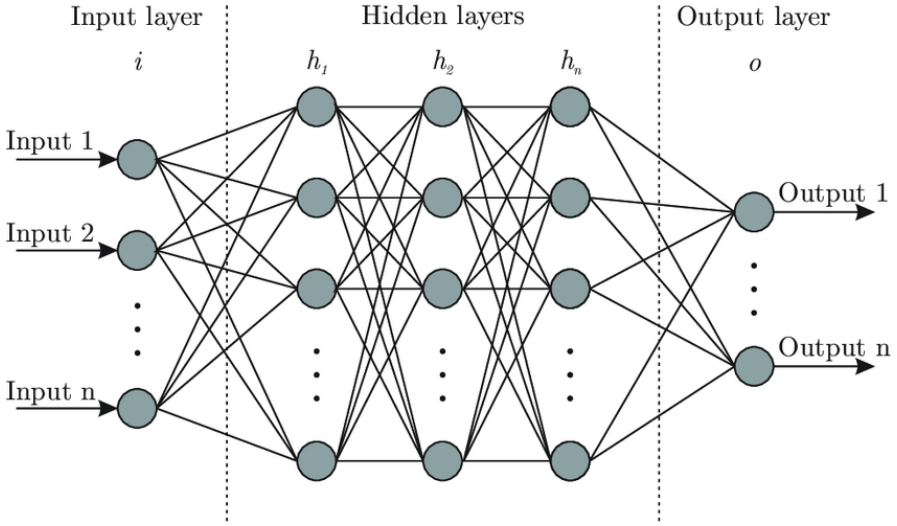
Redes Neuronais: Artificial Neural Networks (ANNs)



Bre, Facundo & Gimenez, Juan & Fachinotti, Víctor. (2017). Prediction of wind pressure coefficients on building surfaces using Artificial Neural Networks. Energy and Buildings. 158. 10.1016/j.enbuild.2017.11.045.

Redes Neuronais: biológicas vs. ANNs



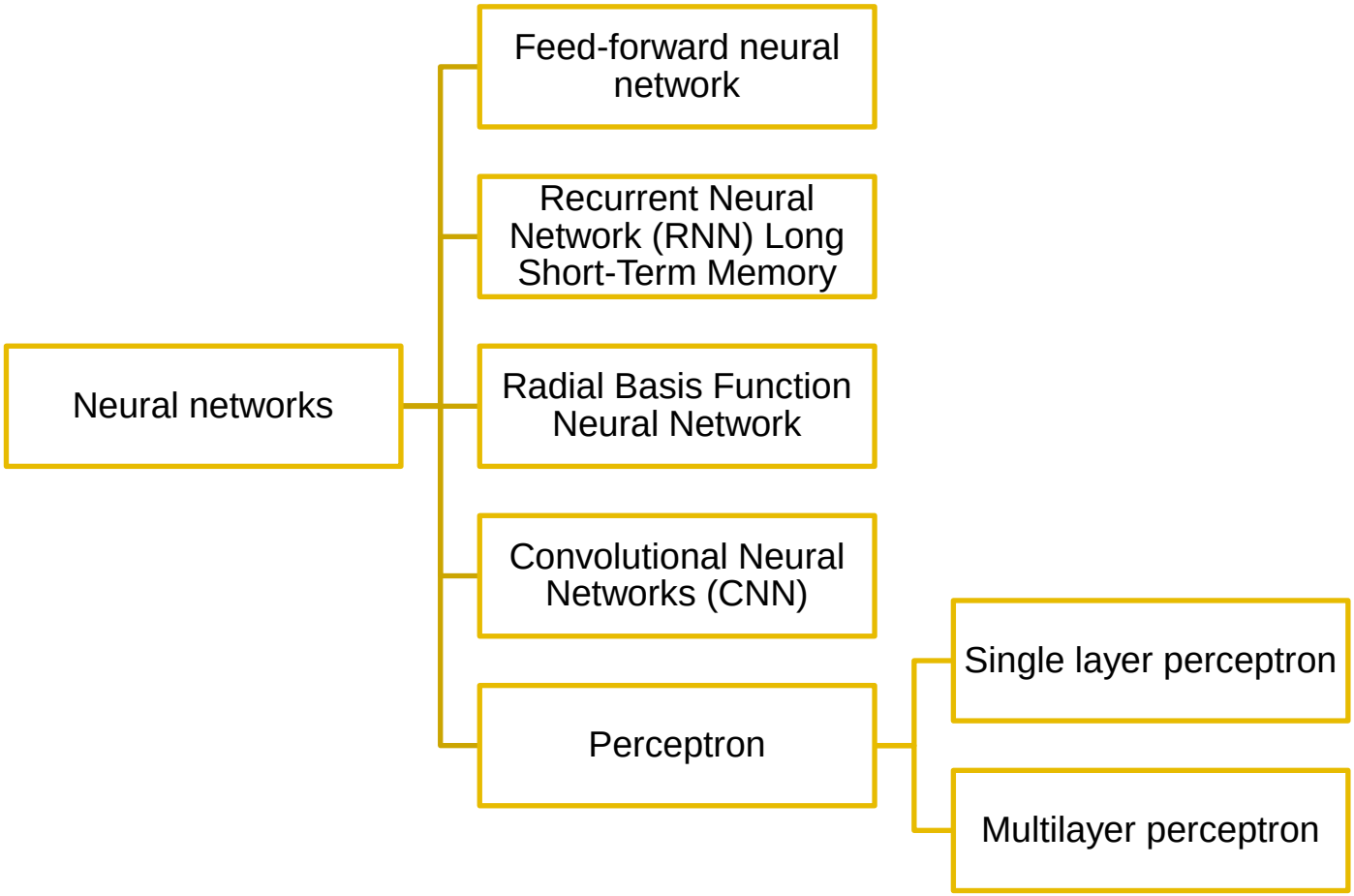


The human brain contains neurons that are composed by several *dendrites* (providing inputs to the neuron) and an *axon* (working as the neuron's output). The neurons are connected and the connections, called *synapses*, allow the communication between neurons. When neurons "fire", they send an electrical impulse that propagates through the cell body, to the *axon* and then the *synapse*. From here, the electrical impulse acts as an input for the subsequent neuron. Each *synapse* has an associated strength and the combination of all the inputs received, when compared to a certain threshold, will define if the neuron will "fire" an electrical impulse to the subsequent neuron.

We focus on the multi-layer perceptron which contains units (*neurons*) organised in layers: the input layer, at least one hidden layer, and the output layer. The units on one layer are connected to the units in the subsequent layer. The output of each unit passes through the connections (*synapses*) to the units in the next layer. This simulates the input and output through *dendrites* and *axon*, respectively. The connections between units have associated weights (*synapse* strength) that influence the impact of the information passed to the next unit.

Each layer has an associated activation function. The input values from the previous layer's units are fed to the activation function, which aggregates them into a single value that is passed onto the following layer's units. The middle layers are said to be hidden because their activation values are not directly accessible from the outside.

Tipos de redes neuronais



Feed-forward Neural Network

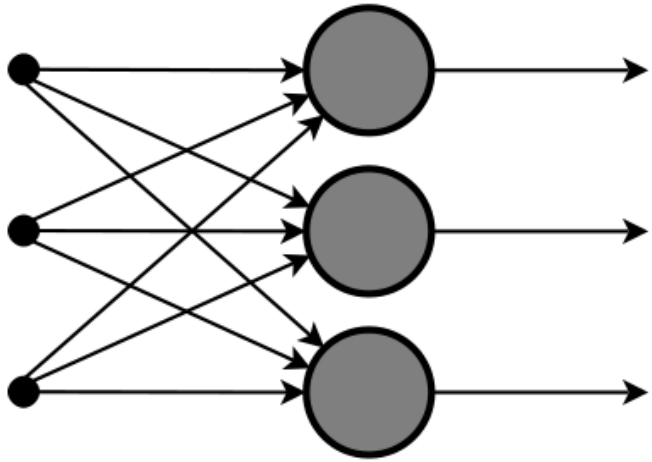
Os dados movem-se apenas numa direção a partir da entrada até atingir a saída.

Ao longo do caminho, é calculada a soma dos produtos dos inputs e pesos.

O resultado final é passado para as saídas para processamento.

Usadas principalmente em **reconhecimento facial e visão computacional**

Estão equipados para lidar com dados que contêm muito ruído.



https://towardsdatascience.com/a-beginners-guide-to-neural-networks-d5cf7e369a13

Recurrent Neural Network (RNN) Long Short-Term Memory

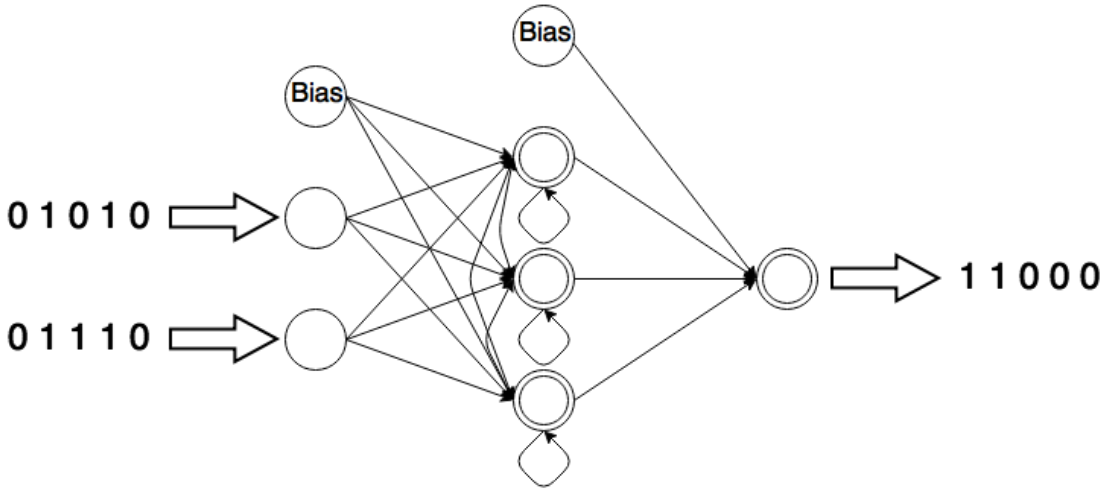
Guarda a saída de uma camada e realimenta a entrada.

A primeira camada é formada como na rede feedforward, sendo calculada a soma dos inputs e dos pesos.

Nas camadas seguintes, o processo recorrente começa: a cada iteração, o nó lembra-se de algumas informações que possuía na iteração anterior. Atua como uma célula de memória enquanto computa e realiza a operação.

A rede começa da mesma forma que a rede feedforward, mas tem memória das informações para poder usar mais tarde.

Muito eficaz para **conversão de voz em texto**



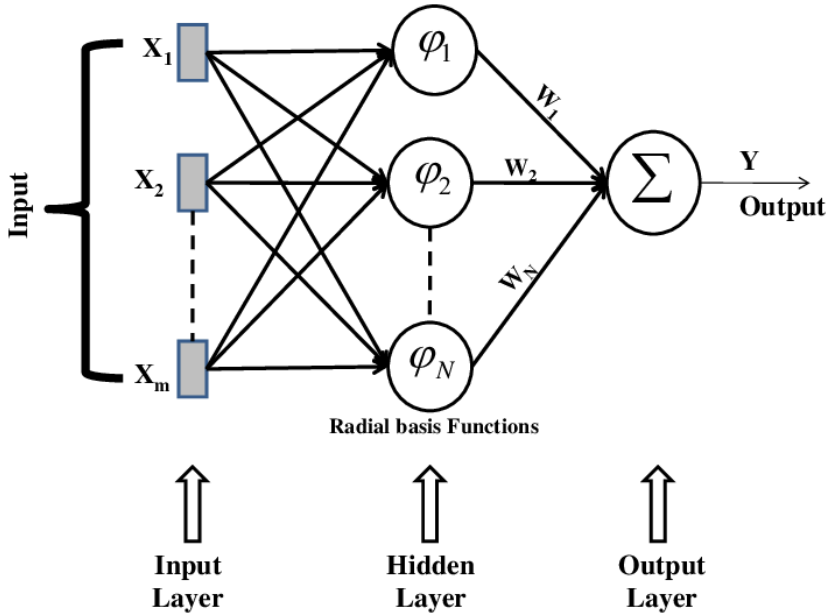
https://towardsdatascience.com/a-beginners-guide-to-neural-networks-d5cf7e369a13

Radial Basis Function Neural Network

É considerada a distância de qualquer ponto em relação ao centro.

Tem duas camadas: interna e externa. A camada interna tem os recursos combinados com a função de base radial (função cujo valor depende da distância entre a entrada e um ponto fixo). Em seguida, a saída desses recursos é utilizada ao calcular a mesma saída na próxima iteração.

Usada principalmente em **sistemas de restauração de energia.**



https://towardsdatascience.com/a-beginners-guide-to-neural-networks-d5cf7e369a13

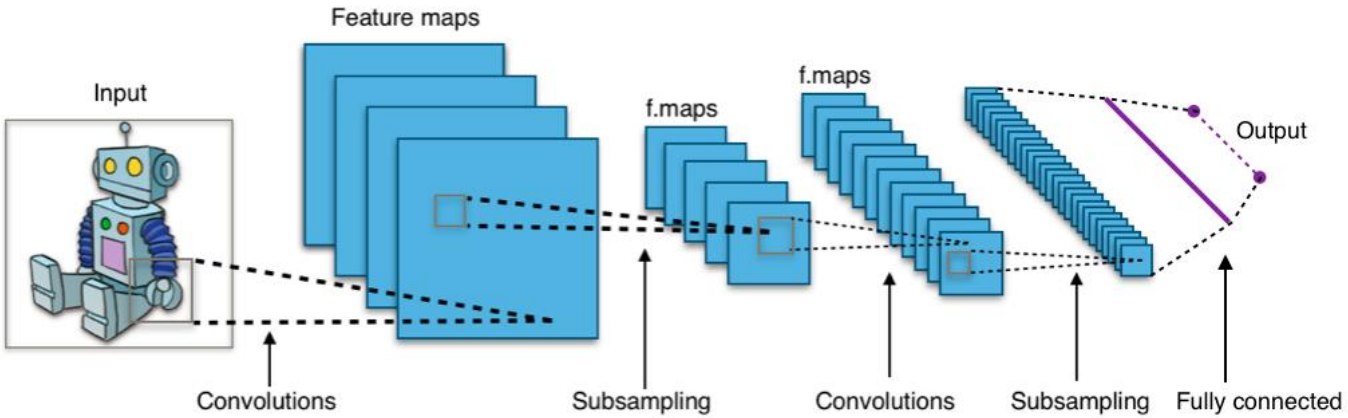
Convolutional Neural Networks (CNN)

Objetivo principal: extrair características da imagem de entrada.

A convolução preserva a relação espacial entre os pixels, extraindo as características da imagem usando pequenos quadrados de dados de entrada.

Composta por uma ou mais camadas convolucionais totalmente conectadas.

Usadas em **visão computacional**, **reconhecimento de objetos** (ex: veículos autónomos).



https://towardsdatascience.com/a-beginners-guide-to-neural-networks-d5cf7e369a13

Perceptron

Algoritmo para aprendizagem supervisionada de classificadores binários

Permite que os neurónios aprendam e processem elementos no conjunto de treino, um de cada vez.

Executa cálculos para decidir se um input pertence ou não a alguma classe específica.

Há dois tipos principais de perceptrons:

- single layer perceptrons (uma única camada)
- multilayer perceptrons (múltiplas camadas)

São classificados como redes neurais feed-forward: movem-se apenas numa direção.

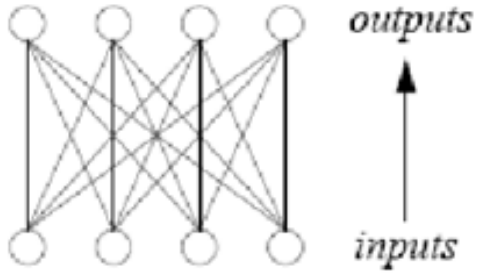
Single layer perceptron

Não tem conhecimento prévio de nenhuma entrada: os pesos iniciais são atribuídos aleatoriamente.

Soma todas as entradas ponderadas (pesos) e, se a soma estiver acima de um threshold, os perceptrons estão ativados.

Os valores de input são apresentados ao perceptron e:

- se o output previsto for igual ao desejado, o desempenho é considerado satisfatório e não são feitas alterações nos pesos.
- se o output previsto não corresponder ao desejado, os pesos precisam de ser ajustados para reduzir o erro.



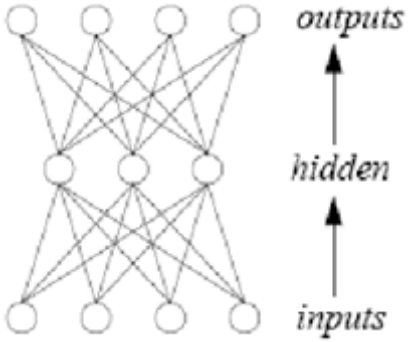
https://towardsdatascience.com/a-beginners-guide-to-neural-networks-d5cf7e369a13

Multilayer perceptron

Tem a mesma estrutura do *single layer perceptron*, mas com uma ou mais camadas escondidas adicionadas.

O algoritmo consiste em duas fases:

- a fase *forward*, onde as ativações são propagadas da camada de entrada para a camada de saída
- a fase *backward*, onde o erro é propagado para trás de modo a modificar os pesos e valores de *bias*.



https://towardsdatascience.com/a-beginners-guide-to-neural-networks-d5cf7e369a13

Informação extra

Para mais informação sobre redes neuronais, ver:

https://towardsdatascience.com/understanding-neural-networks-19020b758230

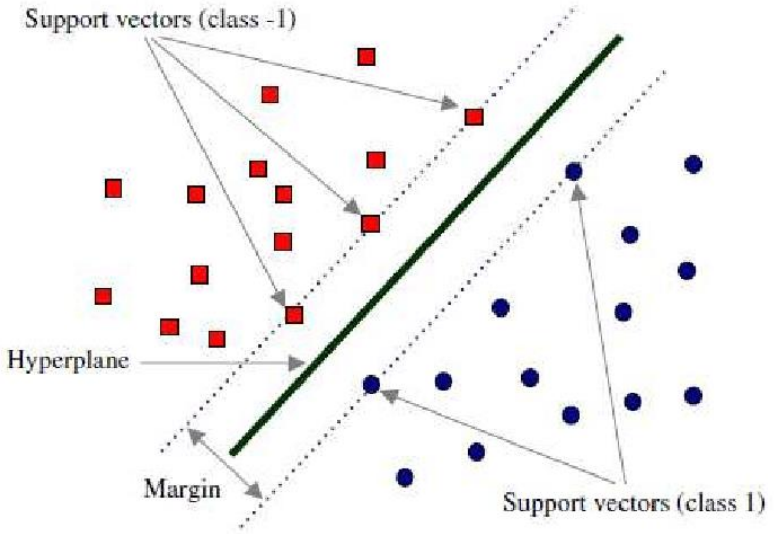
Deep Learning

Redes neuronais com muitas camadas escondidas

Support Vector Machines

Support Vector machines: conceitos

Objetivo: encontrar a melhor separação das classes



Conceitos:

- **Hyperplane**:
  - 1d: ponto que melhor separa as classes
  - 2d: reta que melhor separa as classes
  - 3d: plano que melhor separa as classes
- **Support vectors**: pontos mais próximos do hyperplane
- **Margin**: distância entre os pontos mais próximos de cada classe
- **Kernel**: função matemática utilizada para transformar os dados de input para outro formato (ex: linear, nonlinear, polinomial, …) e quantifica a semelhança (distância) entre duas observações

Support Vector machines: parâmetros

- **Gamma**: define a influencia de um ponto singular
  - Valores mais altos: baixa influência
  - Valores mais baixos: alta influência
- **C** (regularization parameter): controla o tradeoff:
  - fronteira de decisão mais smooth (valores mais baixos)
  - classificar pontos de treino corretamente (valores mais altos).

class `sklearn.svm.` **SVC**(*, *C=1.0, kernel='rbf', degree=3, gamma='scale', coef0=0.0, shrinking=True, probability=False, tol=0.001, cache_size=200, class_weight=None, verbose=False, max_iter=- 1, decision_function_shape='ovr', break_ties=False, random_state=None*) [source]

https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

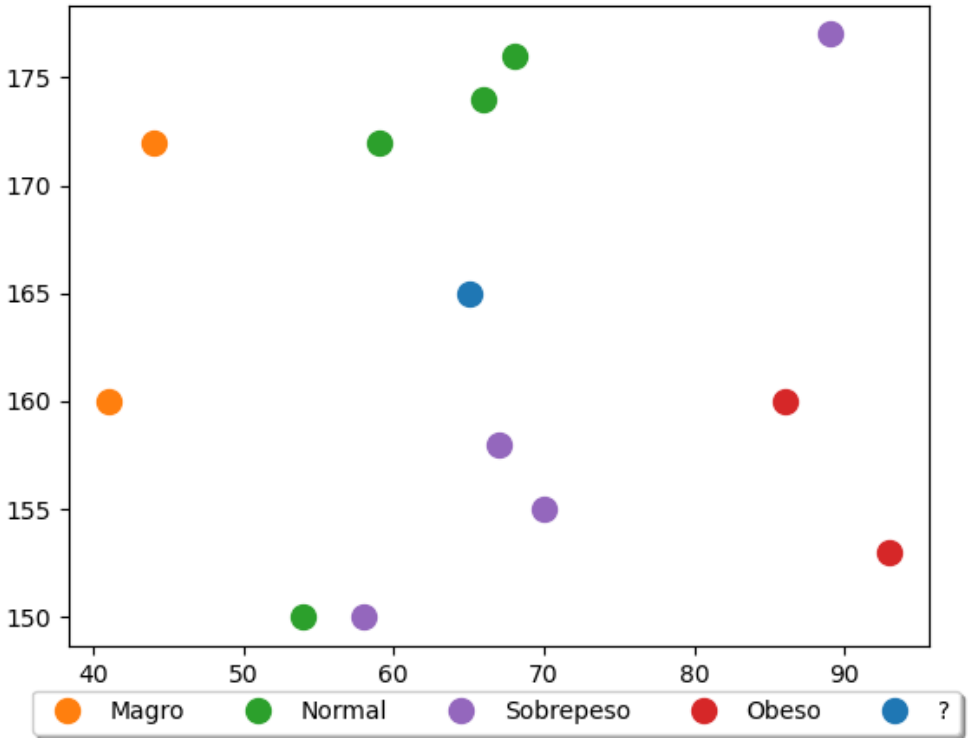
Vantagens e desvantagens

Vantagens	Desvantagens
• Versátil para funções de *kernel* específicas • Eficiente em termos de memória • Eficaz quando o número de dimensões é maior do que o número de amostras	• Sujeito a erros e *overfitting* ao lidar com dados com ruído (ex: pontos sobrepostos com o mesmo *label*) • Longo tempo de computação ao lidar com conjuntos de dados muito grandes • Não fornece uma explicação probabilística dos resultados

Vizinho mais próximo
Nearest Neightbor

Problema

Peso	Altura	IMC
41	160	Magro
44	172	Magro
54	150	Normal
58	150	Sobrepeso
59	172	Normal
66	174	Normal
67	158	Sobrepeso
68	176	Normal
70	155	Sobrepeso
86	160	Obeso
89	177	Sobrepeso
93	153	Obeso
65	**165**	**?**



Algoritmo K Nearest Neighbors

Input:

- O: Conjunto de observações com *label*
- N: Nova observação sem *label*
- K: número de vizinhos a considerar

Passos:

1. Calcular a semelhança de *N* a O_i
2. Obter os *K* vizinhos mais próximos
3. Determinar *label(N)* de acordo com *labels* dos vizinhos
    - Classificação: moda
    - Regressão: média

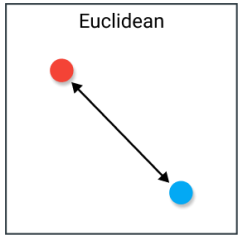
Output:

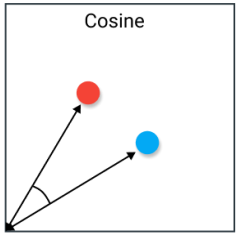
- *Label(N)*

Cálculo da semelhança

Euclidean

$$d(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$



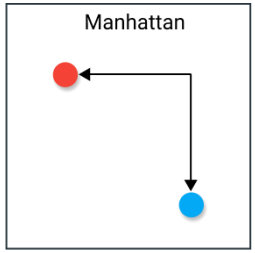


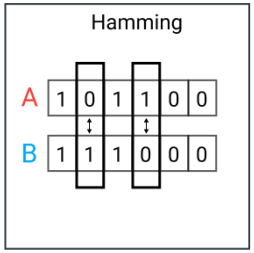
Cosine similarity (ângulo entre vetores)

$$similarity = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|}$$

Manhattan

$$d(x,y) = \sum_{i=1}^{n}|x_i - y_i|$$



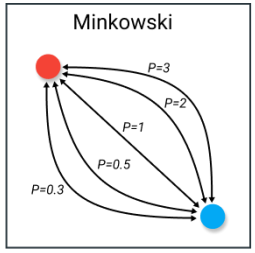


Hamming (distância entre strings)

Número de carateres diferentes entre as strings

Minkowski

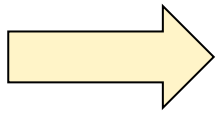
$$d(x,y) = \left(\sum_{i=1}^{n}|x_i - y_i|^p\right)^{\frac{1}{p}}$$



Medidas de distância: https://towardsdatascience.com/9-distance-measures-in-data-science-918109d069fa

No exemplo

Peso	Altura	IMC	Distância
67	158	Sobrepeso	7,3
66	174	Normal	9,1
59	172	Normal	9,2
70	155	Sobrepeso	11,2
68	176	Normal	11,4
58	150	Sobrepeso	16,6
54	150	Normal	18,6
86	160	Obeso	21,6
44	172	Magro	22,1
41	160	Magro	24,5
89	177	Sobrepeso	26,8
93	153	Obeso	30,5
65	**165**	**?**	

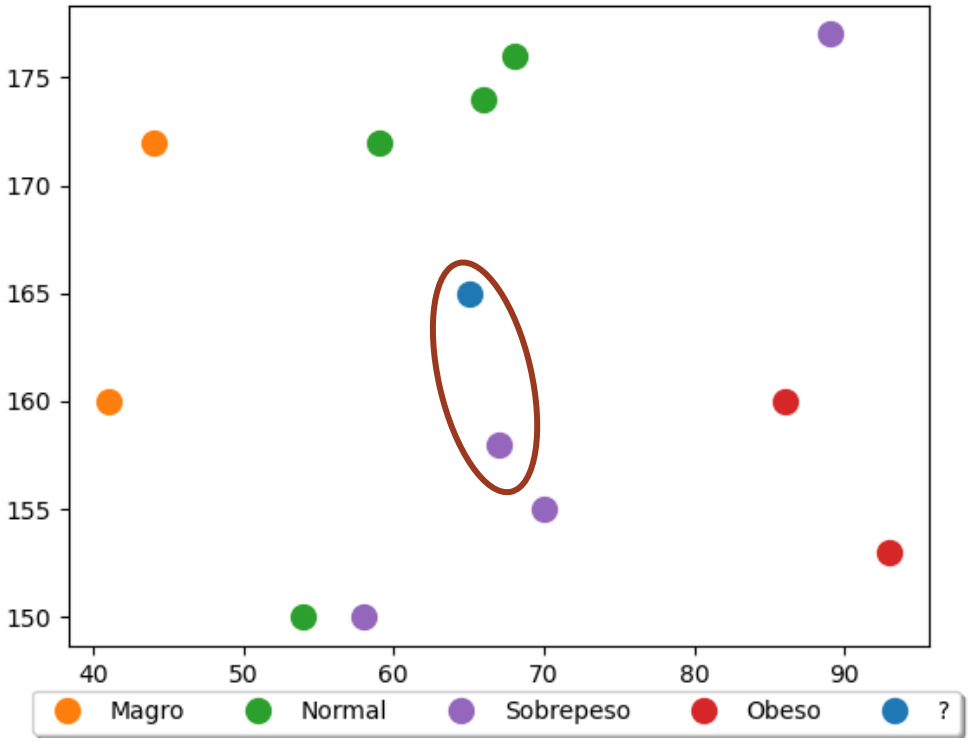


Peso	Altura	IMC	Distância	K
67	158	Sobrepeso	7,3	1
66	174	Normal	9,1	2
59	172	Normal	9,2	3
70	155	Sobrepeso	11,2	4
68	176	Normal	11,4	5
58	150	Sobrepeso	16,6	6
54	150	Normal	18,6	7
86	160	Obeso	21,6	8
44	172	Magro	22,1	9
41	160	Magro	24,5	10
89	177	Sobrepeso	26,8	11
93	153	Obeso	30,5	12
65	**165**	**?**		

No exemplo

K=1

Peso	Altura	IMC	Distância	K
67	158	Sobrepeso	7,3	1
66	174	Normal	9,1	2
59	172	Normal	9,2	3
70	155	Sobrepeso	11,2	4
68	176	Normal	11,4	5
58	150	Sobrepeso	16,6	6
54	150	Normal	18,6	7
86	160	Obeso	21,6	8
44	172	Magro	22,1	9
41	160	Magro	24,5	10
89	177	Sobrepeso	26,8	11
93	153	Obeso	30,5	12
65	**165**	**?**		

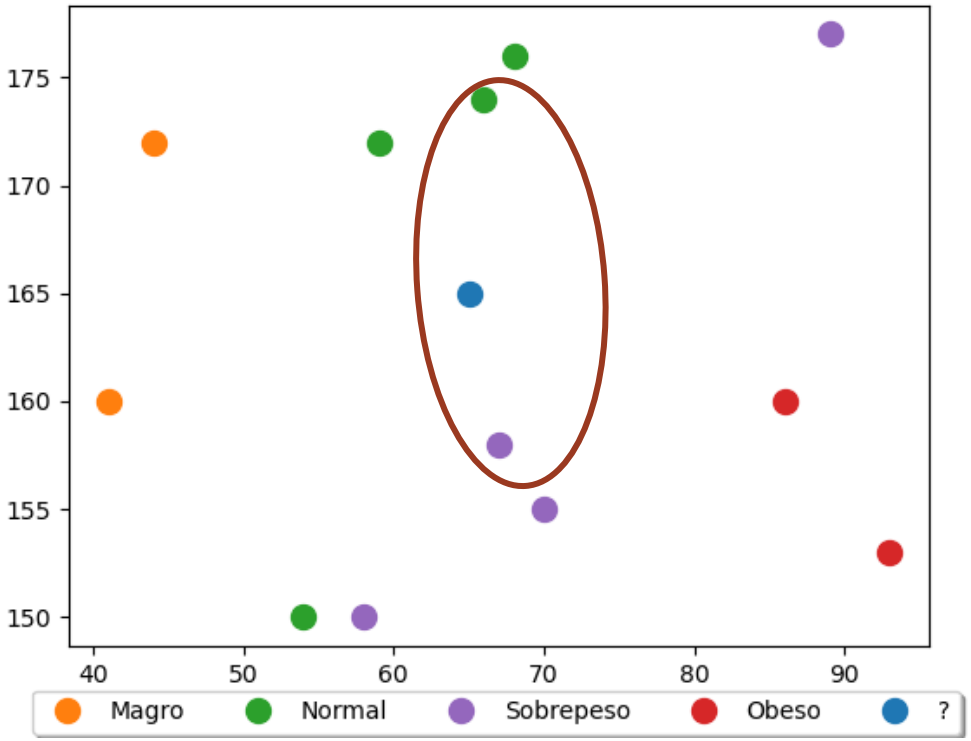


Label: Sobrepeso

No exemplo

K=2

Peso	Altura	IMC	Distância	K
67	158	Sobrepeso	7,3	1
66	174	Normal	9,1	2
59	172	Normal	9,2	3
70	155	Sobrepeso	11,2	4
68	176	Normal	11,4	5
58	150	Sobrepeso	16,6	6
54	150	Normal	18,6	7
86	160	Obeso	21,6	8
44	172	Magro	22,1	9
41	160	Magro	24,5	10
89	177	Sobrepeso	26,8	11
93	153	Obeso	30,5	12
65	**165**	**?**		



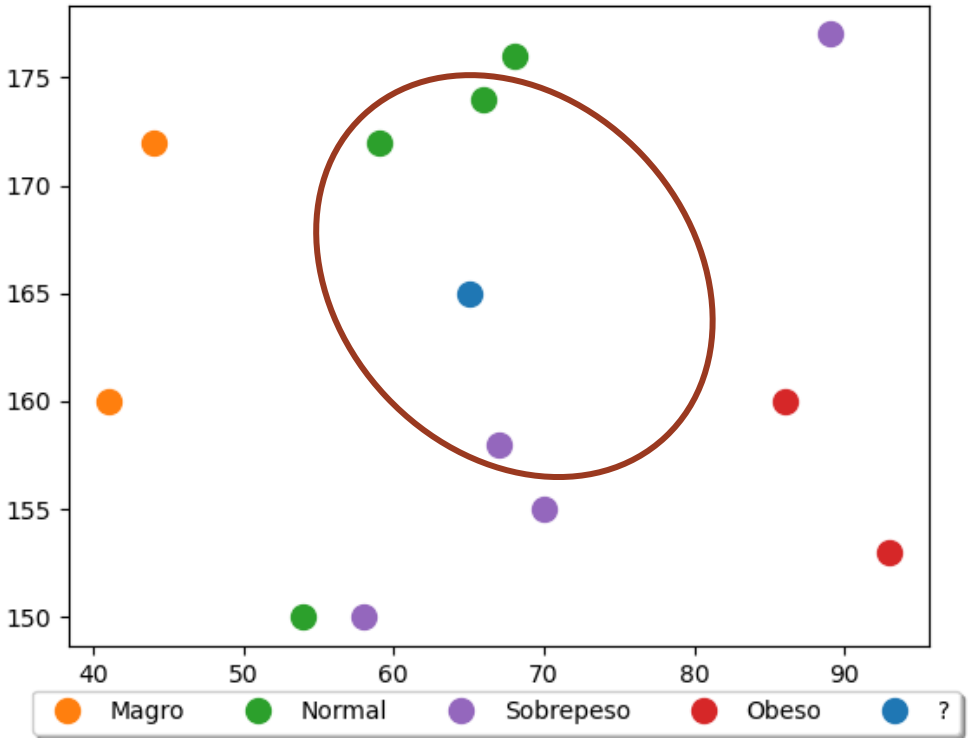
Label: Normal / Sobrepeso

No exemplo

K=3

Peso	Altura	IMC	Distância	K
67	158	Sobrepeso	7,3	1
66	174	Normal	9,1	2
59	172	Normal	9,2	3
70	155	Sobrepeso	11,2	4
68	176	Normal	11,4	5
58	150	Sobrepeso	16,6	6
54	150	Normal	18,6	7
86	160	Obeso	21,6	8
44	172	Magro	22,1	9
41	160	Magro	24,5	10
89	177	Sobrepeso	26,8	11
93	153	Obeso	30,5	12
65	**165**	**?**		

Label: Normal

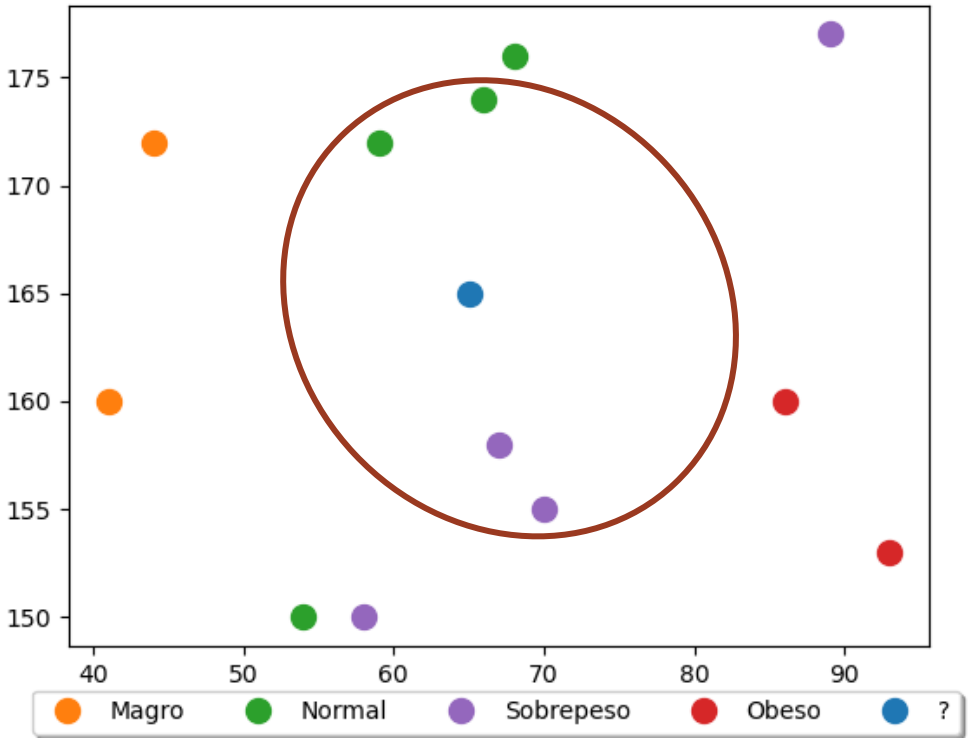


No exemplo

K=4

Peso	Altura	IMC	Distância	K
67	158	Sobrepeso	7,3	1
66	174	Normal	9,1	2
59	172	Normal	9,2	3
70	155	Sobrepeso	11,2	4
68	176	Normal	11,4	5
58	150	Sobrepeso	16,6	6
54	150	Normal	18,6	7
86	160	Obeso	21,6	8
44	172	Magro	22,1	9
41	160	Magro	24,5	10
89	177	Sobrepeso	26,8	11
93	153	Obeso	30,5	12
65	**165**	**?**		

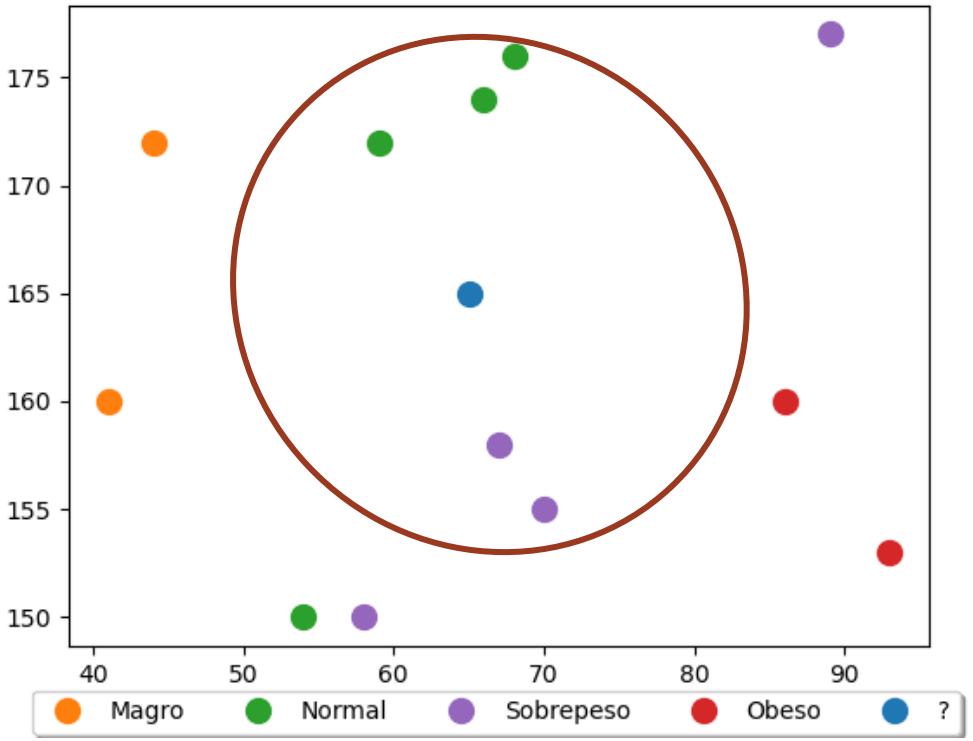
Label: Normal / Sobrepeso



No exemplo

K=5

Peso	Altura	IMC	Distância	K
67	158	Sobrepeso	7,3	1
66	174	Normal	9,1	2
59	172	Normal	9,2	3
70	155	Sobrepeso	11,2	4
68	176	Normal	11,4	5
58	150	Sobrepeso	16,6	6
54	150	Normal	18,6	7
86	160	Obeso	21,6	8
44	172	Magro	22,1	9
41	160	Magro	24,5	10
89	177	Sobrepeso	26,8	11
93	153	Obeso	30,5	12
65	**165**	**?**		



Label: Normal

Determinação do melhor K

- Não existe um método estruturado para encontrar o melhor valor para K. É necessário testar diversos valores em tentativa/erro.

- Escolher valores mais baixos para K pode trazer ruído e terá uma influência maior no resultado.
    - K = 1: sensível a outliers

- Valores altos de K terão menor variância, mas maior *bias*. É computacionalmente mais "caro".
    - K = n, *n* é o número de amostras dos dados de treino: todas as amostras novas vão ter o mesmo *label*
        - Classificação: moda
        - Regressão: média

- Usar *cross validation*: separar conjunto de treino em treino e validação e usá-lo para avaliar diferentes valores de K, escolhendo o que permite melhor desempenho (menor erro: minimização do *validation error*).

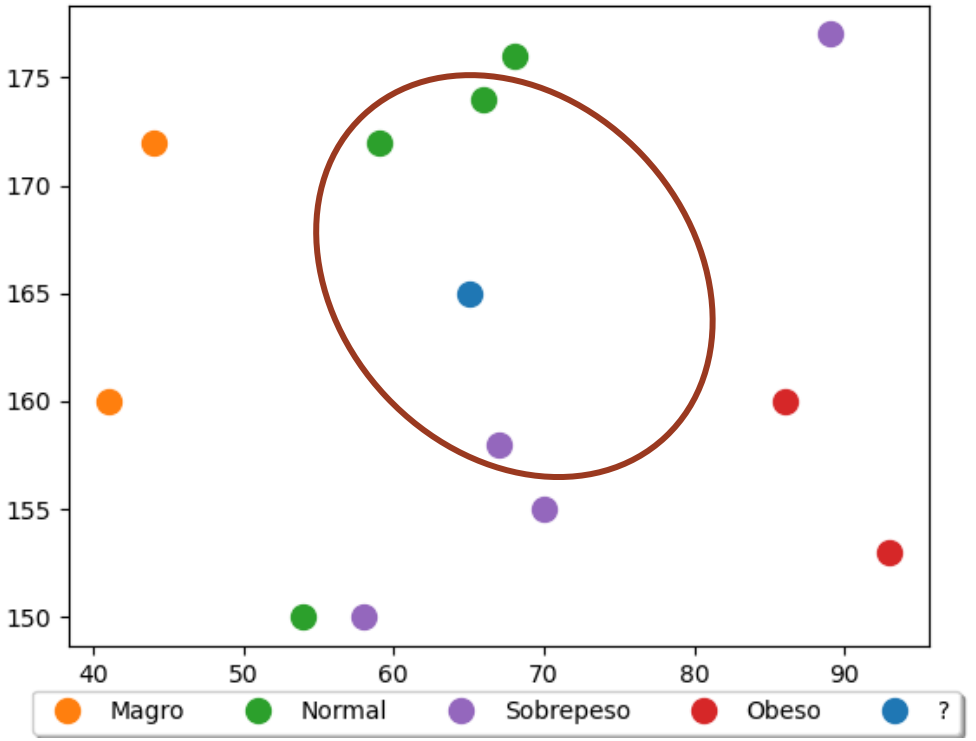
- De uma forma geral, na prática, podemos escolher $\boldsymbol{K = \sqrt{n}}$, em que *n* é o número de amostras dos dados de treino.

- Tentar manter o valor de K ímpar para evitar "empates" entre duas classes

No exemplo

$$\sqrt{12} = 3,46 \approx 3$$

Peso	Altura	IMC	Distância	K
67	158	Sobrepeso	7,3	1
66	174	Normal	9,1	2
59	172	Normal	9,2	3
70	155	Sobrepeso	11,2	4
68	176	Normal	11,4	5
58	150	Sobrepeso	16,6	6
54	150	Normal	18,6	7
86	160	Obeso	21,6	8
44	172	Magro	22,1	9
41	160	Magro	24,5	10
89	177	Sobrepeso	26,8	11
93	153	Obeso	30,5	12
65	**165**	**?**		



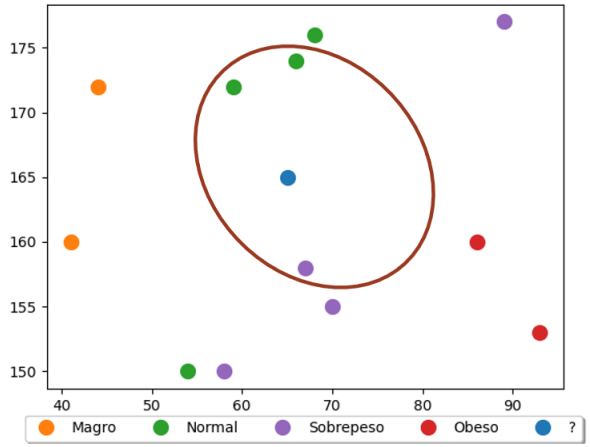
Label: Normal

KNN em Python

K=3

Peso	Altura	IMC	Distância	K
67	158	Sobrepeso	7,3	1
66	174	Normal	9,1	2
59	172	Normal	9,2	3
70	155	Sobrepeso	11,2	4
68	176	Normal	11,4	5
58	150	Sobrepeso	16,6	6
54	150	Normal	18,6	7
86	160	Obeso	21,6	8
44	172	Magro	22,1	9
41	160	Magro	24,5	10
89	177	Sobrepeso	26,8	11
93	153	Obeso	30,5	12
65	**165**	**?**		

Label: Normal



Código:

```
import pandas as pd
data = pd.read_csv("imc.csv")

x=data.iloc[:,0:2]
y=data.iloc[:,2]

from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=3, weights='distance', metric="euclidean")
knn.fit(x, y)

print("previsao (65,165): ", knn.predict([[65, 165]]))
```

Output:

```
previsao (65,165):  ['Normal']
```

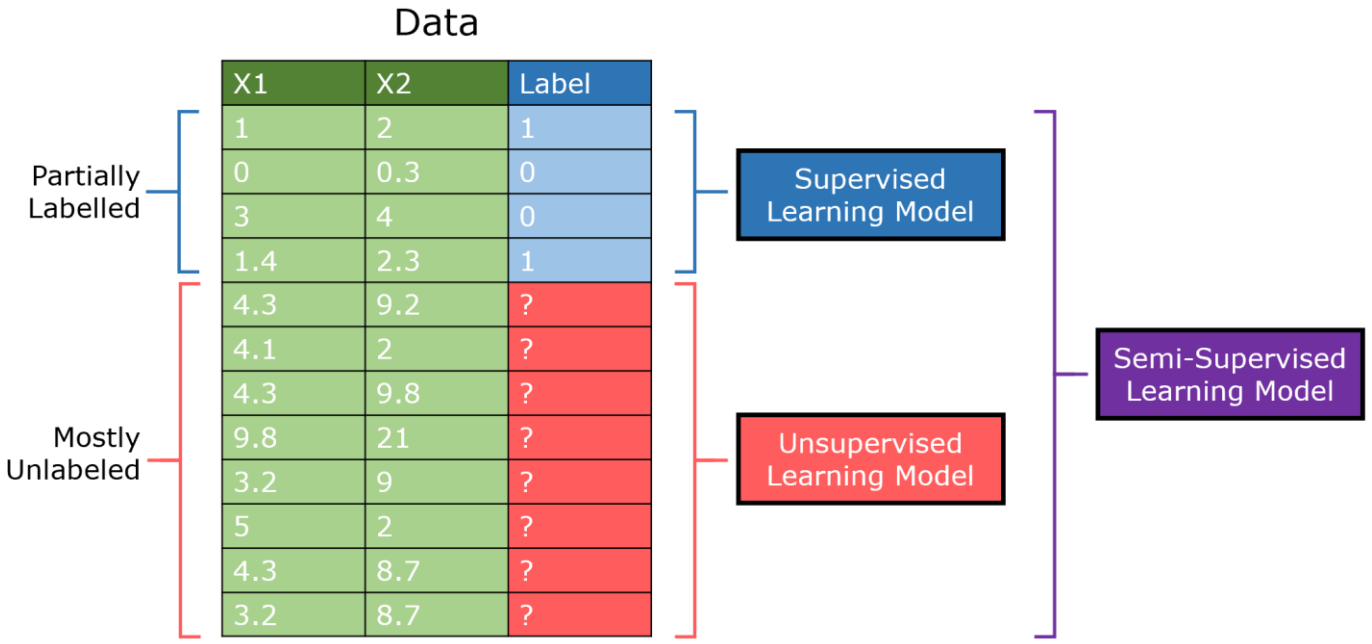
Vantagens e desvantagens de usar K Nearest Neighbors

Vantagens	Desvantagens
• Simples de implementar • Flexível • Lida naturalmente com *multiclass* • Pode ter um bom desempenho na prática com dados suficientes	• É necessário determinar o valor de K • O custo de computação é muito alto porque precisamos calcular a distância de cada instância nova a todas as instâncias de treino • Armazenamento de dados • Devemos garantir que usamos uma função de distância (semelhança) significativa.

Aprendizagem semi-supervisionada

Semisupervised learning

- *Labels* são “caros”
- Podemos ter poucos exemplos com *label* e muitos sem *label*
    - Ex: Atividades fraudulentas



Framework

1. O algoritmo utiliza para treino uma porção limitada do *dataset* contendo exemplos *labeled*
    1. Resultado: modelo “parcialmente treinado”

2. Esse modelo irá classificar a porção do *dataset* que não contém *labels*
    1. Resultado: “pseudo-classificados”

3. Junta-se as duas porções de exemplos
    1. Permite combinar aspetos de aprendizagem descritiva e preditiva

Teoricamente, pode usar-se qualquer algoritmo que seja utilizado para *supervised learning*

Aprendizagem por transferência
Transfer Learning

Transfer learning

Ver documento anexo “transfer_learning.pdf”

UNIVERSIDADE
PORTUCALENSE

Do conhecimento à prática.