INSTITUTO
UNIVERSITÁRIO
DE LISBOA

Introdução à Aprendizagem Automática — 2025/2026
Aprendizagem Supervisionada

> Estes exercícios devem ser resolvidos utilizando Python notebooks devido à possibilidade de gerar um relatório integrado com o código. É assumido que o estudante é proficiente em programação. Todas as respostas devem ser justificadas e os resultados discutidos e comparados com os *baselines* apropriados. Não é permitido usar bibliotecas de algoritmos de AA neste exercício.
> A pontuação máxima da tarefa é de 2 pontos. Os exercícios opcionais (se existentes) ajudam a alcançar a pontuação máxima, complementando erros ou falhas.
> **Deadline:** final da aula da semana de 27 a 31 de outubro, 2025

Nos exercícios seguintes o objetivo é programar algoritmos que, dados exemplos da entrada e saída desejadas, aprendam a imitar o comportamento presente nos dados.

Exercício 1

A “rede” neuronal na Fig. 1 representa um perceptrão com duas entradas e uma saída, que também pode descrito pelas equações seguintes:

$$o = f(s), \quad s = w_0 + w_1 \cdot x_1 + w_2 \cdot x_2 \tag{1}$$

$$f(s) = \begin{cases} 1, & \text{if} \quad s > 0,5 \\ 0, & \text{if} \quad s \leq 0,5 \end{cases} \tag{2}$$

1. **Escolha uma** das seguintes operações binárias (AND ou OR) e **construa dois vectores**: um com todas as diferentes combinações de duas entradas binárias (4 vectores):

   $$\{\{0,0\},\{0,1\},\{1,0\},\{1,1\}\}$$

   onde 0 representa $FALSO$ e 1, $VERDADEIRO$; e um outro vector que contém o alvo / resposta desejada, d, para o vector de entrada correspondente, que deverá ser o re-

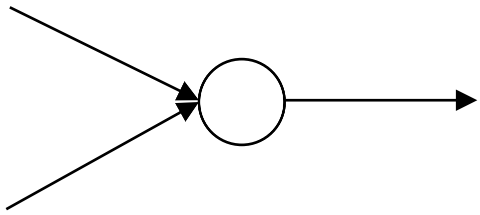

Figura 1: Perceptrão

sultado da operação desejada sobre as entreadas, nomeadamente: OR $\{0,1,1,1\}$ **ou** AND $\{0,0,0,1\}$.

2. **Inicialize** w_0**,** w_1**, e** w_2 **com pequenos valores aleatórios** e, **para cada padrão de entrada calcule a saída da rede**, guardando o resultado no vector o.

3. **Calcule** a diferença / **erro** ($e = d - o$) entre o valor desejado da resposta (d) e a saída (o), **para cada saída**.

4. Para cada erro e, adicione ao termo de atualização para w_0 (Δw_0), w_1 (Δw_1) e w_2 (Δw_2) de acordo com:

$$\Delta w_0 = \Delta w_0 + \alpha \cdot e \tag{3}$$

$$\Delta w_1 = \Delta w_1 + \alpha \cdot x_1 \cdot e \tag{4}$$

$$\Delta w_2 = \Delta w_2 + \alpha \cdot x_2 \cdot e \tag{5}$$

onde $\alpha = 10E - 4$.

5. Prepare o seu código para fazer vários ciclos sobre todo o conjunto de dados (neste caso, 4 exemplos/instâncias) várias vezes fazendo o procedimento acima descrito (**para treinar durante várias "épocas"**).

6. Após todos os exemplos serem apresentados **(no final de cada época), atualize** w_0, w_1 e w_2 de acordo com:

$$w_0 = w_0 + \Delta w_0 \tag{6}$$

$$w_1 = w_1 + \Delta w_1 \tag{7}$$

$$w_2 = w_2 + \Delta w_2 \tag{8}$$

de modo a que na próxima iteração o erro diminua. **Repita 20 épocas**.

(a) **Mostre graficamente o valor do erro em cada época**, qual a tendência do erro ao longo das épocas?

(b) **Mostre graficamente o valor de cada peso após cada atualização**. Os valores estão a convergir? Se sim, verifique se convergem para valores semelhantes em treinos diferentes (com diferentes incializações aleatórias dos pesos)?

(c) **Qual o efeito de aumentar ou diminuir o parâmetro α?** Consegue indicar (aproximadamente) qual o "melhor" valor para α?

(d) **Quantas épocas** (passagens por todo o conjunto de dados) levou até que **todo o conjunto fosse bem classificado?** (i.e. $\forall_i : d_i = o_i$). Repita a experiência 30 vezes, com diferentes inicializações aleatórias dos pesos e apresente a média e desvio-padrão do número de épocas até à convergência.

7. **Gere um conjunto de pontos 2D usando uma distribuição Gaussiana multivariada**.

(a) Use o código na Fig. 2 para gerar dois conjuntos, cada um com 500 pontos (pode reduzir este número para obter melhor vizualizaçao ou execuções mais rápidas se necessário).

(b) Cada conjunto deve ter diferentes centros alguma (pouca) sobreposição.

(c) Adicione uma coluna e preencha-a com 0 para o primeiro conjunto de dados e 1 para o segundo, para que se saiba qual distribuição que gerou cada ponto.

(d) Junte ambos os conjuntos e baralhe.

(e) O gráfico usando como abcissa e orfdenada as primeiras duas colunas deve ser semelhante ao apresentado na Fig. 3.

(f) Escreva o conjunto para um ficheiro.

```python
import matplotlib.pyplot as plt
import numpy as np
import random

mean = [3, 3]
cov = [[1, 0], [0, 1]]
a = np.random.multivariate_normal(mean, cov, 500).T

mean = [-3, -3]
cov = [[2, 0], [0, 5]]
b = np.random.multivariate_normal(mean, cov, 500).T

c = np.concatenate((a, b), axis = 1)
c = c.T
np.random.shuffle(c)
c = c.T

x = c[0]
y = c[1]

plt.plot(x, y, 'x')
plt.axis('equal')
plt.show()
```

Figura 2: Código para gerar duas Gaussianas multivariadas (2D)

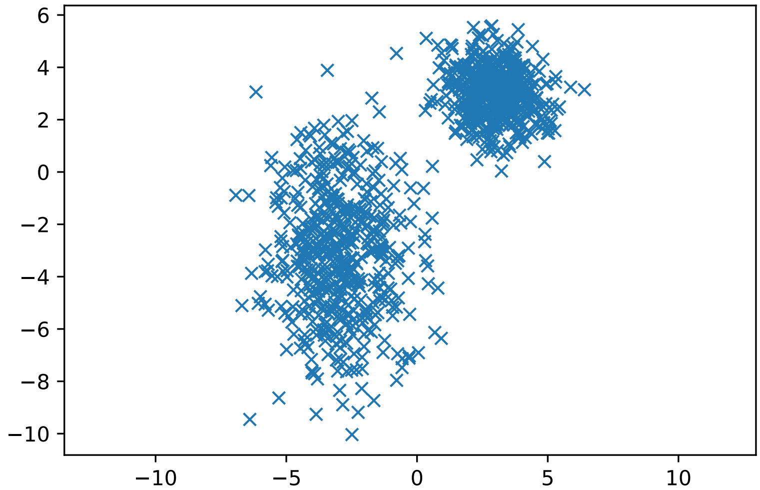

Figura 3: Pontos gerados por duas distribuições Gaussianas multivariável

8. **Use o conjuto gerado** no item anterior como **conjunto de treino** para o perceptrão e treine-o para distinguir pontos gerados por uma ou outra distribuição (ajuste o número de épocas necessário). Note que o mesmo programa aprendeu duas coisas diferentes, dependendo dos dados usados.

   Imprima com cores diferentes:

   (a) Pontos gerados pela primeira distribuição e classificados com 1 pelo perceptrão

   (b) Pontos gerados pela primeira distribuição e classificados com 0 pelo perceptrão

   (c) Pontos gerados pela segunda distribuição e classificados com 1 pelo perceptrão

   (d) Pontos gerados pela segunda distribuição e classificados com 0 pelo perceptrão

9. **Imprima (use uma matriz colorida) a matriz de confusão do teste acima**. Consegue relacionar os números na matriz com as cores da figura?

10. **Imprima as métricas** (percentagem de acerto, precisão, recall e F1) para todos os testes: **as metricas devem ser sempre a média de 30 testes** com os mesmos parâmetros e diferentes pesos iniciais.

Exercício 2

Implemente um classificador k-NN para classificar o conjunto que pode encontrar em `https://archive.ics.uci.edu/ml/datasets/iris`.

Dado o conjunto de dados com exemplos e a sua classe respetiva (o *conjunto de treino*) um novo exemplo (extraído do *conjunto de teste*), o classificador deve **calcular a distância**

euclideana do exemplo a cada elemento do cconjunto de treino, **seleccionar os k mais próximos** e dar como resultado **a classe da maioria dos k exemplos mais próximos** (os k-*Nearest Neighbors*).

1. **Divida o conjunto** em dois sub-conjuntos (70% / 30%). Use o maior como *conjunto de treino* e o menos como *conjunto de teste*. Para todos os exemplos de teste calcule a classificação obtida pelo método de *k-Nearest Neighbors*. Compare as métricas usando $k = 3$, 7 e 11. Para cada caso repita 30 vezes para cada valor de k, com diferentes partições (diferentes elementos no conjunto de treino e teste, mas sempre com a proporção 70/30). Use um gráfico do tipo *boxplot with whiskers* para facilitar a comparação dos valores para diferentes k.

2. **Imprima a matriz de confusão de um dos testes para cada valor de** k.

3. Considerando o conjunto de dados na Fig. 3, porque deve o k ser sempre impar?

Exercício 3

Usando o conjunto de dados do exercício anterior, **implemente um classificador Naive Bayes**.

1. **Transforme, discretizando, todas os valores nas colunas em categorias com três valores possíveis (*low/medium/high*)**. Use uma partição que faça sentido para cada coluna. Tal como no exercício anterior deve partir o conjunto em dois subconjuntos (70% / 30%). Repita o processo usado acima para obter métricas e um exemplo da matriz de confusão (neste caso não tem um parâmetro para varia por isso basta repetir uma vez com 30 partições diferentes).

2. **Como compara este classificadro com o k-NN**?

   Relembra-se quer a probabilidade de um evento/exemplo ser de uma classe $P(Class|X)$ é dada por:

   $$\mathrm{P}(Class|X) = \frac{\mathrm{P}(X|Class)\,\mathrm{P}(Class)}{\mathrm{P}(X)} \tag{9}$$

   e a decisão entre classes para um exemplo X faz-se comparando o valor para cada classe de:

   $$\mathrm{P}(Class)\,\mathrm{P}(X|Class) \tag{10}$$

   que é aproximado (assumindo a independência das *features*) por:

   $$\mathrm{P}(Class)\prod_i \mathrm{P}(X_i|Class). \tag{11}$$

Exercício 4

Use o conjunto do exercício anterior. Use a classe *Iris-setosa* como alvo ($p+$ são todos os exemplos classificados como *Iris-setosa* e $p-$ todos os restantes) e crie três subconjuntos pondo em cada um deles os exemplos que têm o mesmo valor na primeira coluna, i.e. Low DataSet tem todos os elementos que na primeira coluna têm o valor low, Medium Dataset, todos os exemplos que têm o valor $medium$ na primeira coluna e High Dataset todos os que têm $high$.

1. **Calcule a entropia dos 4 conjuntos** (o conjunto completo e os três subconjuntos).

   Relembra-se que a entropia de um conjunto (S) é calculada por:

   $$\text{entropy}(S) = -(p+) \times \log_2(p+) - (p-) \times \log_2(p-) \tag{12}$$

   O cálculo do ganho da partição de S pela *feature* a:

   $$\text{gain}(S, a) = \text{entropy}(S) - \frac{\sum_v (|S_v| \times \text{entropy}(S_v))}{|S|} \tag{13}$$

   onde $|S|$ é o número de elementos de S e S_v representa cada um dos subconjuntos de S quando particionado pelos valores v da coluna a.

2. **Qual o valor de $\text{gain}(S, a)$ para a partição acima?** O que quer isso dizer em relação à capacidade de classificar um novo exemplo antes e depois da partição de S?

3. Faça o mesmo para todas as *features*. **Qual a que tem o maior ganho?** O que quer isso dizer em termos de potencial de classificação de novos exemplos?

4. Explique como construiruia uma árvores de decisão com esta estratégia.