Conhecimento e Raciocínio

Licenciatura em Engenharia Informática: 2º ano - 2º semestre 2023/2024

Ficha de Trabalho nº 5

Aprendizagem supervisionada
Métodos de Regressão
Gradiente descendente

Este tutorial ilustra como usar o Matlab para construir modelos de regressão (aprendizagem supervisionada).
Serão abordados os pontos importantes na construção destes modelos: carregar o *dataset,* visualizar e analisar o *dataset*, normalizar o *dataset*, construir o modelo de regressão, treinar e testar o modelo para prever novos valores e avaliar o seu desempenho usando várias métricas.
O objetivo será usar estes modelos treinados para fazer aprendizagem (previsão) de novos dados.

Esta ficha divide-se em quatro partes:

1. Ler, preparar, analisar e visualizar graficamente os dados do *dataset* fornecido (População da China)
2. Implementar, treinar e testar um modelo de **regressão linear simples** para o *dataset ChinaPop2*
3. Implementar, treinar e testar um modelo de **regressão não linear** (polinomial) para o *dataset ChinaPop2*
4. Implementar o algoritmo do **gradiente descendente** para encontrar os coeficientes reta do modelo de regressão linear.

Os ficheiros necessários para a resolução da ficha encontram-se no Moodle
Também é fundamental ter presentes os tópicos abordados nas aulas teóricas sobre este tema (ver slides)

Para implementar e testar cada uma das tarefas desta ficha sugere-se que criem um ficheiro ***Live Script*** acrescentando diferentes secções em cada uma das tarefas.

- em cada secção acrescentem o código pedido em cada tarefa
- corram cada secção (*Run Section*) e visualizem o resultado produzido
- se necessário, usem o **Clear output** para limpar os resultados de visualizações anteriores.

1. Dataset

O *dataset* encontra-se no ficheiro ChinaPop2.csv. Consiste num *dataset* com 72 registos, com a evolução do número de habitantes da China entre 1950 e 2022.

Crie um ficheiro Live Script e grave-o com nome **Ficha5.mlx**
Na primeira secção do ficheiro implemente o código para:

- Carregar o *dataset*: função `readmatrix`
- Criar as variáveis de entrada (**x** – anos) e de saída (**y** – nº de habitantes) que usaremos nos modelos de regressão: use a manipulação matricial para retirar a primeira coluna (variável **x**) e a segunda coluna (variável **y**)
- Guardar numa variável **m** o número de exemplos do *dataset*: use a função `length`
- Visualizar os dados do *dataset*: use a função `plot` para ver o gráfico com a relação entre as variáveis **x** e **y**
- Observa alguma correlação dos dados?
- Construa um histograma para visualizar como se encontram divididos os dados de **x**. Use 10 categorias. Use a função `histogram`

2. Normalização do dataset

- Crie uma nova secção (use *Section Break*)
- Normalize o *dataset* usando a técnica do Min-Max. Normalize ambas as colunas X e Y

$$X'_i = \frac{X_i - \min(X)}{\max(X) - \min(X)}$$

- Guarde os valores normalizados em duas variáveis **xnorm** e **ynorm**.
- Visualizar os dados do *dataset*: use a função `plot` para ver o gráfico com a relação entre as variáveis **xnorm** e **ynorm**. Comparando com o *plot* feito antes da normalização o que observa?

- Faça a *standardização* do *dataset* usando a técnica do desvio padrão. Normalize ambas as colunas X e Y. Use a função `std` (desvio padrão - σ) e `mean` (média - μ)

$$Xi' = \frac{Xi - \mu}{\sigma}$$

- Guarde os valores normalizados em duas variáveis **xnorm2** e **ynorm2**.
- Visualizar os dados do *dataset*: use a função `plot` para ver o gráfico com a relação entre as variáveis **xnorm2** e **ynorm2**. Comparando com os *plots* feitos anteriormente o que observa?

3. Regressão linear simples

As funções do Matlab usadas para os modelos de regressão são as seguintes:

```
p = polyfit(x, y, g)
```

x: vetor com os valores da variável independente (ano)
y: vetor com os valores da variável dependente (número de habitantes)
g: grau do polinómio, para a regressão linear simples use 1.
p: devolve os coeficientes (a, b) da reta **y = ax+b** que melhor se ajusta aos dados.

```
y_pred = polyval (p, x)
```

p: coeficientes obtidos pela função *polyfit*
x: vetor com valores de x
y_pred: vetor com os valores obtidos pelo modelo de regressão.

3.1. Obter a linha ax + b do modelo de regressão linear

Nesta tarefa use os valores iniciais guardados nas variáveis **x** e **y**
Crie uma nova secção. Implemente o código para:

- Obter a os parâmetros da reta de regressão linear para os dados lidos do *dataset*. Use a função `polyfit` (use grau 1):
  ```
  p = polyfit(x, y, 1)
  ```
- Obter os valores previstos para o número de habitantes usando a reta de regressão linear ajustada com os parâmetros `p` dados pela função anterior. Use a função `polyval`:
  ```
  Y_pred = polyval(p, x)
  ```
- Mostre visualmente os dados reais (**y**) e os dados previstos pelo modelo (**y_pred**): use a função `plot`

3.2. Métricas

Crie uma nova secção. Implemente o código para calcular as seguintes métricas:

- O coeficiente de Pearson (use a função `corr`). O que conclui sobre a correlação dos dados?
- As seguintes métricas

$$SSE = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \qquad MSE = \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n}$$

$$MAE = \frac{\sum_{i=1}^{n}|y_i - \hat{y}_i|}{n} \qquad RMSE = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n}}$$

3.3. Dividir o dataset para treino e teste

Crie uma nova secção. Implemente o código para executar as seguintes tarefas:

- Use os dados sem normalização
- Dividir o *dataset* em duas partes, uma vai ser usada no treino do modelo, outra parte no teste do modelo
- Percentagem para treino: variável **T = 0.7**
- Percentagem para teste (validação) será **1 – T**
- Escolher aleatoriamente os **T * m** exemplos que serão usados no *dataset* de treino. A variável **m** foi inicializada no inicio da aula com o número de exemplos do *dataset*. Use a função **ranperm** para criar um vetor de índices aleatórios de tamanho **m**
- Dividir o *dataset* em duas partes
    - Treino (índices 1 até T*m): x_train, y_train
    - Teste (índices T*m + 1 até ao fim): x_test, y_test
- Use a função *plot* para visualizar os dados de treino e os dados de teste.

3.4. Treinar e testar o modelo

Crie uma nova secção. Implemente o código para executar as seguintes tarefas:

- Tal como fez anteriormente use a função **polyfit** para re-treinar o modelo de regressão linear simples apenas com as variáveis **x_train** e **y_train**
- Usando a função **polyval** crie os outputs do modelo: **y_pred_train** e **y_pred_test**
- Use a função ***plot*** para visualizar os dados observados e os dados previstos (treino e teste)
- Calcule as métricas de erro nas previsões feitas no conjunto de teste. Esta métrica dá-nos o desempenho do modelo **para exemplos que nunca foram usados no treino**: capacidade de aprendizagem/generalização.
- Faça várias execuções. O que observa?
    - Devido à aleatoriedade da divisão do conjunto de treino/teste pode haver diferenças nos resultados de diferentes execuções.
- Altere o valor de T para 50%, 30%, 90%, ... qual o desempenho do modelo?

4. Regressão não linear (polinomial)

Muitas vezes, uma linha não é capaz de capturar a relação entre os dados do *dataset*. Nestes casos pode usar-se uma regressão não linear, usando por exemplo um polinómio de grau 2, 3, ...10, etc. Para calcular os parâmetros do polinómio usa-se na mesma a função **`polyfit`**, alterando apenas o terceiro argumento, para o grau do polinómio que se pretende.
Por exemplo os parâmetros do polinómio de **2º grau**: $\mathbf{y = ax^2 + bx + c}$ é calculado pela função

```
p = polyfit(x,y,2)
```

4.1. Modelo de regressão usando um polinómio

Crie uma nova secção. Implemente as seguintes tarefas:

- Comece por usar os dados não normalizados (x, y). Use a função **polyfit** para treinar o modelo de regressão com um polinómio de 2º grau.
- Usando a função **polyval** verifique os valores previstos pelo modelo.
- Use a função *plot* para visualizar os dados observados e os dados previstos.
- Calcule as métricas de forma semelhante à secção 3.2
- Aumente o grau do polinómio para 3, 4, 5, ... o que observa no desempenho do modelo?
- Os dados do *dataset* antes da normalização têm uma escala demasiado grande e não conseguem ser corretamente ajustadas pelo modelo. Repita as tarefas anteriores usando os dados normalizados **xnorm**, **ynorm**.

5. Gradiente descendente

- **Definir a função de custo**

Antes de implementar o gradiente descendente, a etapa inicial envolve definir a função de custo que requer minimização. A seleção da função de custo **depende do problema específico a ser abordado**. No cenário de regressão linear básico, a função de custo pode ser a soma dos quadrados dos erros (SSE)

- **Inicialização dos parâmetros**

A seguir, precisamos inicializar os parâmetros do nosso modelo com alguns valores iniciais. Para a regressão linear, podemos inicializar os parâmetros **a** e **b** aleatoriamente.

- **Configuração dos híper-parâmetros**

Os híper-parâmetros são parâmetros que controlam o comportamento do algoritmo de otimização. No gradiente descendente, o híper-parâmetro principal é o coeficiente de aprendizagem, que determina o tamanho do passo dado em cada iteração. Além disso, especifique o número de iterações a serem executadas.

- **Executar N iterações de descida do gradiente**

Agora, procedemos à execução das iterações de descida do gradiente. Durante cada iteração, calculamos o gradiente da função de custo em relação aos parâmetros e, posteriormente, atualizamos os valores dos parâmetros movendo-nos na direção oposta ao gradiente. Repetimos este processo até à convergência ou atingir o número máximo de iterações.

Crie uma nova secção. Implemente o algoritmo do gradiente descendente para encontrar os coeficientes da reta de regressão linear. Para testar o algoritmo use os valores normalizados (**xnorm**, **ynorm**)

- Começar com uma reta aleatória: escolher **a** e **b** aleatórios (use a função **rand()**)
- Inicializar o coeficiente de aprendizagem (LR) com 0.01
- Calcular o custo (use o SSE)
- Fazer um ciclo de 100 iterações para ajustar os valores de **a** e **b** e diminuir o valor do custo usando o gradiente (minimizar SSE):

$$a = a - LR * 2 \sum_{i=1}^{n} (\boldsymbol{a}x_i + \boldsymbol{b} - y_i).x_i$$

$$b = b - LR * 2 \sum_{i=1}^{n} (\boldsymbol{a}x_i + \boldsymbol{b} - y_i)$$

No final do ciclo imprima o valor do erro (SSE) e faça o *plot* das variáveis **xnorm**, **ynorm** e dos valores estimados pela reta final.

Altere o valor do coeficiente de aprendizagem para 0.1, 0.001, ...
Altere o número de iterações.
Execute e compare os resultados obtidos pelo gradiente descendente com os coeficientes obtidos usando a função **polyfit**