Universidade do Minho
Departamento de Informática
Mestrado em Engenharia Biomédica

Ramo de Informática Médica
Imagiologia
4º Ano, 2º Semestre
Ano letivo 2021/2022

Enunciado Prático MLP Regressão
7 de abril de 2022

Tema *MLP em classificação de previsões (continuação)*

Enunciado Pretende-se, com esta ficha, que seja realizado um conjunto de tarefas que permitam consolidar o conhecimento adquirido sobre o uso de uma rede neuronal *Multilayer Perceptron (MLP)* na classificação de previsão de dados, com recurso à biblioteca *Pytorch.*

Tarefas Usando um SW da sua escolha, execute as seguintes tarefas:

T1 – Implemente o exemplo de MLP para regressão (3.MLPRegression_basic.pdf e housing.csv)

T1.1 – Prepare os dados

a) Crie uma classe para o *dataset*
   i. Leia o *dataset*
   ii. Crie um método para indicar o nº de casos no *dataset*
   iii. Crie um método para retornar um caso
   iv. Crie um método para retornar índices para casos de treino e de teste
b) Prepare o *dataset*
c) Prepare os dados
d) Faça um controlo do que está a ser feito (*sanity check*)

T1.2 – Defina o modelo

a) Crie uma classe para o modelo MLP
   i. Defina os elementos do modelo
   ii. Crie um método para a sequência de propagação do *input* (*forward*)
b) Defina a rede neuronal e imprima os parâmetros do modelo

T1.3 – Treine o modelo

a) Faça o *import* de *PlotLosses* de *livelossplot*
b) Crie um método para treinar o modelo usando os casos de treino
   i. Crie uma instância *liveloss*
   ii. Defina as funções de *loss* e de otimização (*Mean Squared Error* e *Stochastic Gradient Descent*, respetivamente)
   iii. Aplique a iteração das *epochs* para
      ° Calcule os outputs do modelo (previsão/ *forward*)
      ° Calculae a perda (*loss*)
      ° Calcule a *accuracy* (*Mean Squared Error*)
      ° Calcule os gradientes de perda (*backpropagation*)
      ° Atualize os pesos do modelo
      ° Imprima para cada *epoch*, o rácio de *loss* e o rácio de *accuracy*

- Crie *logs*, atualize o *liveloss* e envie, para visualizar o processo de treino

c) Treine o modelo

→ Repare que as funções de *loss* e de otimização estão a tender para o infinito. Atualize o *learning rate* para 0.00000001 e repita as tarefas T1.1, T1.2 e T1.3.

T1.4 – Avalie o modelo

a) Crie um método para avaliar o modelo
b) Avalie o modelo usando os casos de teste
c) Calcule as previsões e os valores reais
d) Calcule a *accuracy* usando a função *Mean Squared Error*

T1.5 – Use o modelo

a) Crie um método para fazer a previsão utilizando um caso
    i. Converta o caso para tensor
    ii. Faça a previsão
    iii. Converta a previsão num *array numpy*
b) Caso: [0.00632,18.00,2.310,0,0.5380,6.5750,65.20,4.0900,1,296.0,15.30,396.90,4.98]
c) Imprima a previsão

T2 – Melhore o exemplo anterior (3.MLPRegression_enhanced.pdf e housing.csv)

T2.1 – Prepare os dados

a) Crie uma classe para o *dataset*
    i. Leia o *dataset*
    ii. Crie um método para indicar o nº de casos no *dataset*
    iii. Crie um método para retornar um caso
    iv. Crie um método para retornar índices para casos de treino e de teste
b) Prepare o *dataset*
c) Prepare os dados
d) Faça um controlo do que está a ser feito (*sanity check*)

T2.2 – Visualize os dados

a) Faça o *import* de *display* de *IPython.display*
b) Crie um método para visualizar os dados onde cria uma instância do *dataset*
c) Crie um método para visualizar o *dataset*
    i. Imprima a quantidade de casos de treino, a quantidade de casos de teste, o *batch size* dos casos de treino e o *batch size* dos casos de teste
d) Visualize os dados e o *dataset*

T2.3 – Verifique o balanceamento do dataset

a) Faça o *import* de *seaborn* e de *matplotlib.pyplot*
b) Crie um método para visualizar o balanceamento do *dataset*
    i. Conte o nº de casos de treino, o nº de casos de teste, a média dos valores de treino, a média dos valores de teste, e imprima esses dados
    ii. Crie o gráfico

T2.4 – Defina o modelo

a) Faça o *import* de *summary* de *torchinfo*
b) Crie uma classe para o modelo MLP
    i. Defina os elementos do modelo
    ii. Crie um método para a sequência de propagação do *input* (*forward*)
c) Defina a rede neuronal
d) Visualize a rede

T2.5 – Treine o modelo

d) Faça o *import* de *PlotLosses* de *livelossplot*
e) Crie um método para treinar o modelo usando os casos de treino

i. Crie uma instância *liveloss*
ii. Defina as funções de *loss* e de otimização (*Mean Squared Error* e *Adam*, respetivamente)
iii. Aplique a iteração das *epochs* para
- Calcular os outputs do modelo (previsão/ *forward*)
- Calcular a perda (*loss*)
- Calcular a *accuracy* (*Mean Squared Error*)
- Calcular os gradientes de perda (*backpropagation*)
- Atualizar os pesos do modelo
- Imprima para cada *epoch*, o rácio de *loss* e o rácio de *accuracy*
- Crie *logs*, atualize o *liveloss* e envie, para visualizar o processo de treino

f) Treine o modelo

T2.6 – Avalie o modelo

a) Crie um método para avaliar o modelo
b) Avalie o modelo usando os casos de teste
c) Calcule as previsões e os valores reais
d) Calcule a *accuracy* (*Mean Squared Error*)
e) Contabilize os casos em que o modelo acertou e os casos em que falhou
f) Imprima o relatório de classificação

T2.7 – Use o modelo

a) Crie um método para fazer a previsão utilizando um caso
   i. Converta o caso para tensor
   ii. Faça a previsão
   iii. Converta a previsão num *array numpy*
b) Caso: [0.00632,18.00,2.310,0,0.5380,6.5750,65.20,4.0900,1,296.0,15.30,396.90,4.98]
c) Imprima a previsão

Links úteis:

https://www.freecodecamp.org/news/machine-learning-mean-squared-error-regression-line-c7dde9a26b93/

https://gobiviswa.medium.com/mean-square-error-loss-functions-b8ed6c396f41

http://zerospectrum.com/2019/06/02/mae-vs-mse-vs-rmse/

https://medium.com/@Biboswan98/optim-adam-vs-optim-sgd-lets-dive-in-8dbf1890fbdc