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 bin
7 de abril de 2022

Tema *MLP em classificação de previsões*

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 – Instale o *PyTorch* (https://pytorch.org/get-started/locally/)

T2 – Implemente o exemplo de MLP para classificação binária (1.HelloMLP_basic.pdf e ionosphere.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 (s*anity check*)

T2.2 – 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

T2.3 – Treine o modelo

a) Crie um método para o treino do modelo
   i. Defina as funções de *loss* e de otimização
   ii. Aplique a iteração das *epochs* para
      ° Calcular os outputs do modelo (previsão/ *forward*)
      ° Calcular a perda (*loss*)
      ° Calcular os gradientes de perda (*backpropagation*)
      ° Atualizar os pesos do modelo
b) Treine o modelo

T2.4 – 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*

T2.5 – 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: [1,0,0.99539,-0.05889,0.85243,0.02306,0.83398,-0.37708,1,0.03760,0.85243,-0.17755,0.59755,-0.44945,0.60536,-0.38223,0.84356,-0.38542,0.58212,-0.32192,0.56971,-0.29674,0.36946,-0.47357,0.56811,-0.51171,0.41078,-0.46168,0.21266,-0.34090,0.42267,-0.54487,0.18641,-0.45300]
c) Imprima a previsão

T3 – Melhore o exemplo anterior (1.HelloMLP_enhanced.pdf e ionosphere.csv)

T3.1 – 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*)

T3.2 – 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*

T3.3 – 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, o nº de casos de treino a 0, o nº de casos de treino a 1, o nº de casos de teste a 0, o nº de casos de teste a 1, o rácio de g/b e b/g, e imprima esses dados[1]
    ii. Crie o gráfico

T3.4 – Defina o modelo

a) Instale o *torchinfo*
b) Faça o *import* de *summary* de *torchinfo*
c) 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*)
d) Defina a rede neuronal
e) Visualize a rede

T3.5 – Treine o modelo

a) Instale o *livelossplot*
b) Faça o *import* de *PlotLosses* de *livelossplot*

[1] Os casos de valor *0* correspondem à letra *b* e os casos de valor *1* correspondem à letra *g*

c) Crie um método para treinar o modelo usando os caos de treino
    i. Crie uma instância *liveloss*
    ii. Defina as funções de *loss* e de otimização (*BCEWithLogistsLoss* e *Adam*, respetivamente)
    iii. Aplique a iteração das *epochs* para
        ° Calcular os outputs do modelo (previsão/ *forward*)
        ° Calcular a perda (*loss*)
        ° Calcular a *accuracy*
        ° 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
d) Treine o modelo

T3.6 – Avalie o modelo

a) Crie um método para avaliar o modelo
b) Crie um método para apresentar a matriz confusão
c) Avalie o modelo usando os casos de teste
d) Calcule as previsões e os valores reais
e) Calcule a *accuracy*
f) Contabilize os casos em que o modelo acertou e os casos em que falhou
g) Imprima o relatório de classificação
h) Crie a matriz confusão
i) Apresente a matriz confusão

T3.7 – 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: [1,0,0.99539,-0.05889,0.85243,0.02306,0.83398,-0.37708,1,0.03760,0.85243,-0.17755,0.59755,-0.44945,0.60536,-0.38223,0.84356,-0.38542,0.58212,-0.32192,0.56971,-0.29674,0.36946,-0.47357,0.56811,-0.51171,0.41078,-0.46168,0.21266,-0.34090,0.42267,-0.54487,0.18641,-0.45300]
c) Imprima a previsão

T4 (OPCIONAL) – Melhore a *accuracy* obtida nos casos de teste. Tem de *descomentar* a última parte da linha da função *get_splits* na preparação de dados. Isto fará com que os resultados possam ser repetíveis. Deverá entregar o código produzido assim como um documento onde indica a *accuracy* conseguida e todas as alterações efetuadas.