



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 multiclasse

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 classificação *multiclass* ([2.MLPMulticlass_basic.pdf](#) e [iris.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) Crie um método para o treino do modelo
 - i. Defina as funções de *loss* e de otimização (*Sparse Categorical Cross Entropy* e *Stochastic Gradient Descent*, respetivamente)
 - 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

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 *Accuracy Score*

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: [5.1,3.5,1.4,0.2]
- c) Imprima a previsão

T2 – Melhore o exemplo anterior ([2.MLPMulticlass_enhanced.pdf](#) e [iris.csv](#))

T2.1 – Prepare os dados

- a) Crie uma classe para o *dataset*
 - i. Leia o *dataset*
 - ii. Crie um método para indicar o n° 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 n° de casos de treino, o n° de casos de teste, o n° de casos de treino a 0, o n° de casos de treino a 1, o n° de casos de treino a 2, o n° de casos de teste a 0, o n° de casos de teste a 1, n° de casos de teste a 2, e imprima esses dados¹
 - 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

- 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*

¹ Os casos de valor 0 correspondem à *Iris setosa*, os casos de valor 1 correspondem à *Iris versicolor*, e os casos de valor 2 correspondem à *Iris virginica*.

- ii. Defina as funções de *loss* e de otimização (*Sparse Categorical Cross Entropy* e *Stochastic Gradient Descent*, respectivamente)
- iii. Aplique a iteração das *epochs* para
 - Calcular os outputs do modelo (previsão/ *forward*)
 - Calcular a perda (*loss*)
 - Calcular a *accuracy* (*Accuracy Score*)
 - 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 *live loss* e envie, para visualizar o processo de treino
- c) Treine o modelo

T2.6 – 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* (*Accuracy Score*)
- f) Contabilize os casos em que o modelo acertou e os casos em que falhou
- g) 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: [5.1,3.5,1.4,0.2]
- c) Imprima a previsão

Links úteis:

<https://medium.com/swlh/cross-entropy-loss-in-pytorch-c010faf97bab>

<https://jbencook.com/cross-entropy-loss-in-pytorch/>

https://gombru.github.io/2018/05/23/cross_entropy_loss/

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