Classificação de resultado final em Lógica de Programação usando Multi Layer Perceptron

Introdução

Este experimento foi desenvolvido por Vilson Rodrigues Câmara Neto. A base de dados usada foi advinda do sistema LoP da ECT-UFRN, ela é extensa e contém muitos atributos, desde as notas das provas 1 e 2 à submissões durante as 12 primeiras semanas, ela contém 4 semestres, de 2017.2 até 2019.1. O objetivo é prever o mais cedo possível o resultado final na matéria. Para resolução deste problema foi utilizada a *Neural Network Multi Layer Perceptron - MLP* da biblioteca *Keras* e o *Google Colaboratory* para inserir e executar os códigos *python*. A *MLP* tenta simular o funcionamento do cérebro, ela possui camadas de neurônios, o que a diferencia do *Perceptron* original.

Metodologia

Foi separado a base de dados em 2, uma para treinamento e outra para teste, as proporções foram 70% e 30% respectivamente. Os atributos utilizados foram:

- Nota da prova 1
- Submissões em listas de exercício na semanas 4 e 5

No experimento foi usado a de 3 camadas, cada uma com 4 neurônios, o tipo de propagação utilizada foi o “Uniform” e a função de ativação foi das 3 camadas de neurônios foi a “Relu”, e a camada de saída teve a função de ativação “Sigmoid”.

Para o treinamento da rede o valor de “batch_size” foi de 10 e o das “epochs” foi de 8000. Este modelo retornou uma acurácia de 82,9%.

Códigos

Separando conjunto de teste e treinamento:

```
# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 0)
```

Camadas da MLP:

```
# Adding the input layer and the first hidden layer
classifier.add(Dense( activation = 'relu', input_dim = 2 , units = 4, kernel_initializer = 'uniform'))

# Adding the second hidden layer
classifier.add(Dense( activation = 'relu', units = 4, kernel_initializer = 'uniform' ))

# Adding the second hidden layer
classifier.add(Dense( activation = 'relu', units = 4, kernel_initializer = 'uniform' ))


# Adding the output layer
classifier.add(Dense( activation = 'sigmoid', units = 1, kernel_initializer = 'uniform'))
```

Configuração do treinamento:

```
# Fitting the ANN to the Training set
classifier.fit(X_train, y_train, batch_size = 10, epochs = 8000)
```

Experimentos

O resultado obtido por esse modelo de MLP foi muito bom, com quase 83% de acurácia da para intervir ainda na primeira unidade na tentativa de recuperar um aluno que apresenta similaridades com outros alunos que acabaram reprovando em semestres anteriores. A quantidade de camadas de neurônios e o número de neurônios geraram esse resultado, mostrando que os parâmetros ficaram bem ajustados, assim como o número de épocas, que dizia por quanto tempo os neurônios iriam “repassar”.

Referências

***Keras: The Python Deep Learning library*.** Disponível em:<http://keras.io/>. Acesso em: 11 set. 2019.

GÉRON, Aurélien. ***Hands on: Machine Learnig with Sckit-Learn and Tensor Flow***. Ed: 1. Alta Books.