Lecture 13 – Neural Network

Mr. Yousif G. Arshak
University of Zakho
Computer Science Department
yousif.arshak@uoz.edu.krd

OUTLINES

- Introduction
- Neural Network
- Problem Statement
- Evaluation Metric
- Implementation of Neural Network in Python

Introduction

- Neural Networks are used to solve a lot of challenging artificial intelligence problems. They often outperform traditional machine learning models because they have the advantages of non-linearity, variable interactions, and customizability. In this guide, we will learn how to build a neural network machine learning model using scikit-learn. But before we start, it is a good idea to have a basic understanding of a neural network.

Neural Network

- The process of creating a neural network begins with the *perceptron*. In simple terms, the perceptron receives inputs, multiplies them by some weights, and then passes them into an activation function (such as logistic, relu, tanh, identity) to produce an output.
- Neural networks are created by adding the layers of these perceptrons together, known as a multi-layer perceptron model. There are three layers of a neural network - the input, hidden, and output layers.
- The *input layer* directly receives the data, whereas the *output layer* creates the required output. The layers in between are known as *hidden layers* where the intermediate computation takes place.
- A neural network algorithm can be used for both classification and regression problems. Before we start building the model, we will gain an understanding of the problem statement and the data.

Problem Statement

- The aim of this guide is to build a classification model to detect diabetes. We will be using the diabetes dataset which contains 768 observations and 9 variables, as described below:

1.pregnancies - Number of times pregnant.
2.glucose - Plasma glucose concentration.
3.diastolic - Diastolic blood pressure (mm Hg).
4.triceps - Skinfold thickness (mm).
5.insulin - Hour serum insulin (mu U/ml).
6.bmi – Basal metabolic rate (weight in kg/height in m).
7.dpf - Diabetes pedigree function.
8.age - Age in years.
9.diabetes - “1” represents the presence of diabetes while “0” represents the absence of it. This is the target variable.

Evaluation Metric

We will evaluate the performance of the model using accuracy, which represents the percentage of cases correctly classified.
Mathematically, for a binary classifier, it's represented as accuracy = (TP+TN)/(TP+TN+FP+FN), where:

- True Positive, or TP, are cases with positive labels which have been correctly classified as positive.
- True Negative, or TN, are cases with negative labels which have been correctly classified as negative.
- False Positive, or FP, are cases with negative labels which have been incorrectly classified as positive.
- False Negative, or FN, are cases with positive labels which have been incorrectly classified as negative.

Implementation of Decision Tree in Python

Step 1 - Loading the Required Libraries and Modules

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import sklearn from sklearn.neural_network
import MLPClassifier from sklearn.neural_network
import MLPRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from math import sqrt
from sklearn.metrics import r2_score
```

Step 2 - Reading the Data and Performing Basic Data Checks

```
df = pd.read_csv('diabetes.csv')
print(df.shape)
df.describe().transpose()
```

Step 3 - Creating Arrays for the Features and the Response Variable

```
target_column = ['diabetes']
predictors = list(set(list(df.columns))-set(target_column))
df[predictors] = df[predictors]/df[predictors].max()
df.describe().transpose()
```

Step 4 - Creating the Training and Test Datasets

- ```
  X = df[predictors].values y =
  df[target_column].values X_train, X_test,
  y_train, y_test = train_test_split(X, y,
  test_size=0.30, random_state=40)
  print(X_train.shape); print(X_test.shape)
  ```
  ```

Step 5 - Building, Predicting, and Evaluating the Neural Network Model

- ```
  from sklearn.neural_network import MLPClassifier
  mlp = MLPClassifier(hidden_layer_sizes=(8,8,8),
  activation='relu', solver='adam', max_iter=500)
  mlp.fit(X_train,y_train) predict_train =
  mlp.predict(X_train) predict_test =
  mlp.predict(X_test)
  ```
- ```
  from sklearn.metrics import
  classification_report,confusion_matrix
  print(confusion_matrix(y_train,predict_train))
  print(classification_report(y_train,predict_trai
  n))
  ```

- ```
  print(confusion_matrix(y_test,predict_test))
  print(classification_report(y_test,predict_test
  ))
  ```
  ```