COMP809 – Data Mining and Machine Learning

Lab 5 – Neural Network/ MLP

This lab covers implementations related to Multi_Layer Perception (MLP) classifier using sklearn module. In addition to the implementation of the classifiers you will learn how to display important curves/graphs using loss/accuracy.

For this excersice we use pima-indians-diabetes data set which uses eight numeric attributes to identify whether a patient has diabetes or not.

```
#Importing libraries
import pandas
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.neural_network import MLPClassifier
import matplotlib.pyplot as plt
```

```
#Load Data
url =
"https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima
-indians-diabetes.data.csv"
names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi',
'age', 'class']
rawdata = pandas.read_csv(url, names=names)
array = rawdata.values
nrow, ncol = rawdata.shape
predictors = array[:, 0:8]
target = array[:, 8]
```

#A function to see some of the attributes of NN

```
def NN_properties(model):
    loss_values = model.loss_
    print("Loss", loss_values)
    iterations = model.n_iter_
    print("iterations",iterations)
    classes_assigned= model.classes_
    print("Assigned classes", classes_assigned)
```

#Displaying loss curve using loss_curve method.Note that this only works with the MLP default solver "adam"

```
def make_plots_default(model):
    plt.plot(model.loss_curve_)
    plt.title('Loss Curve')
    plt.xlabel('Epochs')
    plt.ylabel('Loss')
    plt.show()
```

#A generic function to display training loss and testing accuracy of MLPClassifier

```
def make_plots_all(mlp, target_train, target_test,
predictors_test,predictors_train):
    max_iter = 100
    accuracy = []
    losses = []
    for i in range(max_iter):
        mlp.fit(predictors_train, target_train)
        iter_acc = mlp.score(predictors_test, target_test)
        accuracy.append(iter_acc)
        losses.append(mlp.loss_)
    plt.plot(accuracy, label='Test accuracy')
```

```
    plt.plot(losses, label='Loss')
    plt.title("Accuracy and Loss over Interations", fontsize=14)
    plt.xlabel('Iterations')
    plt.legend(loc='upper right')
    plt.show()


#A function for model building and calculating accuracy
def get_accuracy(target_train, target_test,
predictors_test,predictors_train):
    # Two hidden layers with 10 and 5 neurons  - NN
    clf = MLPClassifier(activation='logistic', solver='adam',
learning_rate_init=0.01, hidden_layer_sizes=(10, 5),
random_state=1, max_iter=200, warm_start=True)


    #Calling the make_plots_allfunction with unfitted model
    make_plots_all(clf, target_train, target_test,
predictors_test, predictors_train)


    clf.fit(predictors_train, np.ravel(target_train, order='C'))
    predictions = clf.predict(predictors_test)
    NN_properties(clf)  ##Calling NN_properties to see the model
attributes
    make_plots_default(clf)  ##Calling make_plots function to see
the error plots
    return accuracy_score(target_test, predictions)


#train-test split
pred_train, pred_test, tar_train, tar_test =
train_test_split(predictors, target, test_size=.3,
random_state=4)
#Calling get_accuracy function which also invoke other
functions NN_properties, make_plots, make_plots_all
print("Accuracy score: %.2f" % get_accuracy(tar_train,tar_test,
pred_test, pred_train))
```