Department of Computer Engineering
Faculty of Engineering, University of Peradeniya
CO542
Neural Networks and Fuzzy Systems
2021

Lab 04 - Multi Layer Perceptrons (MLP)

Objectives

- Implement and train multi layer perceptron using python libraries.
- Practice the usage of neural networks in real world applications.
- Familiarize data prepossessing functions and effect of changing hyper-parameters.

Multi Layer Perceptrons

- A multi layer perceptron (i.e. feedforward neural networks with hidden layers) contains at least two layers of functional units.
- This means that at least one layer contains hidden units, which do not communicate with the environment.
- If the number of hidden units is appropriately chosen, multi layer perceptrons are universal approximators, i.e. they can solve, at least theoretically, any association problem. In other words, they can approximate any function, given enough inputs and target outputs (e.g.: nonlinearly separable classification, nonlinear regression and prediction problems).

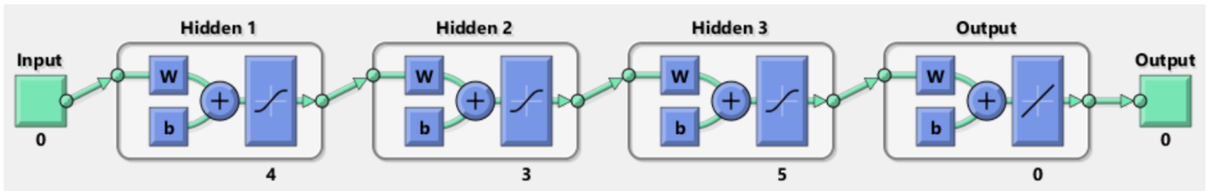

Figure 1: Multi Layer Perceptron

Multilayer Perceptrons are modeled using Scikit-learn in a 6-step process:

1. **Step 1:** Like always first we will import the modules.
   (Note: Here we have loaded the data using sklearn.datasets; however, you can also load your data using any preferred way, e.g. external urls, .csv files etc.)

```
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import train_test_split
```

```
#Importing the iris dataset
from sklearn.datasets import load_iris

#Here are some additional useful imports
import pandas as pd
from sklearn.metrics import plot_confusion_matrix
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
```

2. **Step 2:** Separate data frames "X" and "y", the values of the independent and dependent variables.

```
iris_data = load_iris()
X = pd.DataFrame(iris_data.data, columns=iris_data.feature_names)
y = iris_data.target
```

3. **Step 3:** Split the dataset into train and test dataset.
(We have reserved 20% of the dataset for checking the accuracy of the trained model. Independent train and test dataset are further scaled to make sure that the input data is standard normally distributed are centered around zero and have variance in the same order.)

```
X_train, X_test, y_train, y_test = train_test_split(X,y,
                                                 random_state=1,
                                                 test_size=0.2)
sc_X = StandardScaler()
X_trainscaled=sc_X.fit_transform(X_train)
X_testscaled=sc_X.transform(X_test)
```

4. **Step 4:** Model the MLP.
(We have used four hidden layers with different neurons in each layer. Considering the input and output layer, we have a total of 6 layers in the model. In case any optimizer is not mentioned then "Adam" is the default optimizer.)

```
model = MLPClassifier(hidden_layer_sizes=(256,128,64,32),
                    activation="relu",
                    random_state=1)
```

5. **Step 5:** Train the model and make predictions.

```
clf = model.fit(X_trainscaled, y_train)
y_pred=clf.predict(X_testscaled)
```

6. **Step 6:** Evaluate the accuracy of the classifier.

```
print(clf.score(X_testscaled, y_test))

fig=plot_confusion_matrix(clf, X_testscaled,y_test,
                         display_labels=["Setosa","Versicolor","Virginica"])
fig.figure_.suptitle("Confusion Matrix for Iris Dataset")
plt.show()
```

For more information refer to:

- sklearn.neural_network.MLPClassifier.
- sklearn.neural_network.MLPRegressor.

Task 1

Create and train a neural network that solves the XOR problem.

1. What is the number of inputs and outputs of the neurons?
2. Why is this problem NOT linearly separable?
3. What is the input training vector and target vector?
4. Create a network named 'netXOR' with 2 neurons in the input layer, 5 neurons in the hidden layer and 1 output neuron.

Note: Usually the selection of the,

- number of hidden layers
- number of neurons in each hidden layer
- type of activation functions
- the training algorithm

are done by trial and error technique and no general guidelines are assigned to choose the network parameters.

Task 2

In this exercise we will look at a real world scenario where neural networks are used: function fitting.

1. Generate data (inputs and outputs) for the following function,

$$y = -x^4 + x^3 + 23x^2 - 21x + 32 \tag{1}$$

   x should be between -100 and 100; use an interval of 0.001 between the values.
2. What is the number of inputs and outputs of the network?
3. Model the MLP using MLPRegressor instead of MLPClassifier. (They are almost the same; except for very subtle differences. Try to find their differences).
4. Generate a test set (of your choice) and test it with the network.
5. Plot the train data and model predictions on a same plot and observe up to what extend the predicted values are fitting with the original data set.

Task 3

- In this exercise you will train an MLP to classify an iris dataset into two classes.
- Please use the provided data set in FEeLS to complete the task.

1. Load the dataset as a Panda's dataframe and drop any unnecessary columns.
2. Visualize the relationships between dataset features using seaborn.pairplot

3. Separate the dataset into features and labels.
4. Convert categorical data in your dependent variable into numerical values using Sklearn Label Encoder.
5. Split the data set as train and test data accordingly (E.g.: 80% training data, 20% test data).
6. Scale the independent train and test datasets using Sklearn Standard Scaler.
7. Model the MLP using MLPClassifier.
8. Make predictions using previously reserved test data set.
9. Observe the accuracy of the model by plotting the confusion metrix..
10. Generate the classification report using Sklearn Classification Report and comment on each of the value you obtained in that report.
11. Repeat the task with 100,300,500 iterations and comment on your results.
12. Repeat the task with varying test and train data set sizes and comment on your observations.
13. Repeat the task with different learning rates: 0.002, 0.5, 1.00 and comment on the results obtained.

Submission

You must submit 3 folders (for each exercise) containing,

1. All required .py files (or .ipynb files).
2. A pdf containing all required outputs (images of output curves, network training window, confusion metrics, etc...) and explanations before the deadline (announced in the course page)

Note: Please submit all the files (pdfs, code files etc.) in a single zipped file renamed as lab_4_E_XXX.zip where XXX stands for your index no.