Homework3, Due Date: Wed, Apr 10

Instructor: Dr. Mohammad Pourhomayoun

Problem1: Diagnosis of Fetal Cardio-Vascular Disease from Cardiotocography Using ANN in SciKitLearn:

a- In this problem, I selected a new medical dataset from UCI ML repository. The dataset includes 2126 fetal cardiotocograms (CTGs) collected in the Biomedical Engineering Institute in Porto, Portugal. The dataset includes 21 feature elements, and the output (NSP) has 3 classes: 1: Normal; 2: Suspect; 3: Pathologic.
For more info, you can visit: **https://archive.ics.uci.edu/ml/datasets/Cardiotocography**
You can either download the original raw data from the above link OR download the clean version of the data as a csv file from CSNS in the HW package ("CTG_clean.csv"). The goal is to build an Artificial Neural Network that can diagnose (classify) the fetal health status. The last column (NSP) in the dataset is the label: NSP - fetal state class code (1=normal; 2=suspect; 3=pathologic), and the first 21 column are the features. To see what each feature represents, take a look at the dataset info link above.

b- Use *sklearn.preprocessing.scale* function to normalize the features.

c- Design an ANN with one hidden layer with **30 neurons** to predict the fetal's health status. For your ANN, Use **random_state=1, learning_rate_init = 0.02, solver='adam', alpha=1, verbose=True, activation='logistic'**. Use 10-fold Cross-Validation to evaluate your model. Make sure to add "*verbose=True*" to see the training process. Then, Test your ANN on testing set, and calculate and report the accuracy.

d- Fix the random state for reproducibility: seed = 1, np.random.seed(seed).
Now, use **GridSearchCV** to find the best number of neurons for your 1-hidden layer network. Search in the range of (5-250) with the step size of 5 for the number of neurons (i.e. 5 neurons, 10 neurons, 15 neurons, … , 245 neurons, 250 neurons). As for other parameters, use the same arguments as part (c) for your network. What is the best accuracy, and best number of neurons?

Problem2: Face Recognition Using SVM:

a- Download the dataset "Face" from CSNS. Check out the dataset. Open some of the jpg images. This is the Oivetti database of face images from AT&T research lab. It includes 400 faces (64x64 pixels) from 40 people (10 images per person).
You have to also download the csv file that includes the labels of the images (the label is person's ID). The goal is to build a **Face Recognition** algorithm to recognize each person using PCA dimensionality reduction and a non-linear SVM.

you can use:
mpimg.imread(file_name) to load an image, and
plt.imshow(image_name, cmap=plt.cm.gray) to show an image (This is a little different from what we had in before!). Add **%matplotlib inline** at top of your code to make sure that the images will be shown inside the Jupyter explorer page.

b- Build the feature matrix and label vector: Each image is considered as a data sample with pixels as features. Thus, to build the feature table you have to convert each 64x64 image into a row of the feature matrix with 4096 columns.

c- Normalize each column of your feature matrix using preprocessing.scale (This is required!).

d- Use sklearn functions to split the normalized dataset into testing and training sets with the following parameters: **test_size=0.25, random_state=5**.

e- The dimensionality of the data samples is 4096. Use PCA (Principal Component Analysis) to reduce the dimensionality from 4096 to 50 (i.e. only 50 principal components!). You should "**fit**" your PCA on your training set only, and then use this fitted model to "**transform**" both training and testing sets (When you finish this step, the number of columns in your testing and training sets should be 50). We will cover the details of PCA in next sessions of class. But for now, you can use this format:

```
from sklearn.decomposition import PCA
k = 50  # k is the number of components (new features) after dimensionality reduction
my_pca = PCA(n_components = k)
# X_Train is feature matrix of training set before dimensionality reduction,
# X_Train_New is feature matrix of training set after dimensionality reduction:
X_Train_new = my_pca. fit_transform(X_Train)
X_Test_new = my_pca. transform(X_Test)
```

f- Design and Train a non-linear SVM classifier with "RBF Kernel" to recognize the face based on the training dataset that you built. Use **SVC(C=1, kernel='rbf', gamma=0.0005, random_state=1)**. Then, Test your SVM on testing set, and calculate and report the accuracy. Also, calculate and report the Confusion Matrix using **metrics.confusion_matrix(y_test, y_predict**).

g- Now, use **GridSearchCV** to find the best value for parameter **C** in your SVM. Search in this list: [0.1, 1, 10, 100, 1e3, 5e3, 1e4, 5e4, 1e5].
Important Note: Remember that in this part, we want to use cross-validation method (GridSearchCV) to find the best C. Thus, you don't use X_train/ X_test anymore. You have to perform another pca, this time on the entire dataset after normalization (**X_normalized_pca = my_pca.fit_transform(X_normalized)**), and then use it in GridSearchCV with 10-fold cross validation to find C.