Project 1 Report Data Mining

Priyank Arora 1001553349
Gangadhar Viswanathan 1001535640

March 18, 2018

Abstract

The purpose of this project is to study the detailed working of 4 different data classifiers KNN, Centroid, SVM, and Linear Regression. This report includes detailed accuracy measures of the classifiers used for different Test, Train data sets and K fold cross validations on them.

The classification program includes the data handler code as well. Upon execution, the programs ask the user input for the specific task to be implemented (Task A/B/C/D) and outputs accordingly. The Project is done using **Python (2.7.12)** using numpy, sys, matplotlib, sklearn libraries.

1 Project Task A

Use the data-handler to select "A, B, C, D, E" classes from the handwritten-letter data. From this smaller data set, Generate a training and test data: for each class using the first 30 images for training and the remaining 9 images for the test. Do classification on the generated data using the four classifiers.

1.1 Solution

- Use `letter_2_digit_convert` to get class id of string "ABCDE" 1,2,3,4,5.
- SplitData2TestTrain returns `[test_Vector, test_Label, train_Vector, train_Label]` for 0-29 as training and 30-38 as test.
- Accuracy of the Label of `test_Vector`is predicted using SVM, Centroid, 5NN, and Linear regression.
- Task A outputs the classification accuracy of all the four classifiers.

1.2 Output

Command: python project1.py A

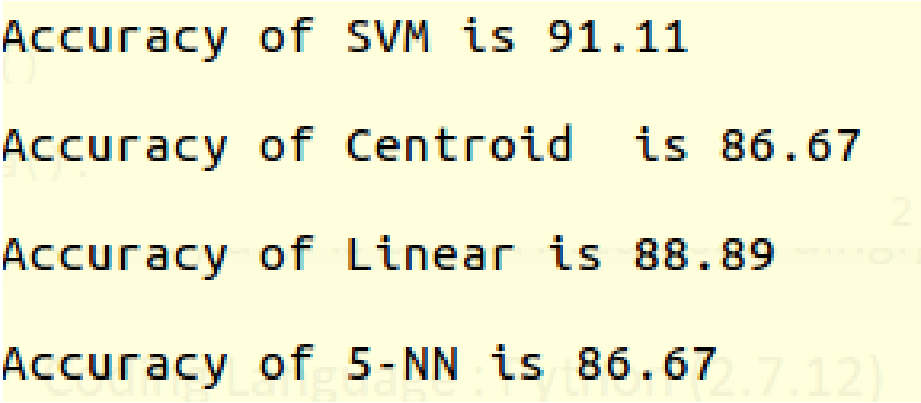

Figure 1: Task A output: showing the efficiency of accuracy in predicting class labels using classifications.

2 Project Task B

On ATNT data, run 5-fold cross-validation (CV) using each of the four classifiers: KNN, centroid, Linear Regression and SVM. If you don't know how to partition the data for CV, you can use the data-handler to do that. Report the classification accuracy of each classifier. Remember, each of the 5-fold CV gives one accuracy. You need to present all 5 accuracy numbers for each classifier. Also, the average of these 5 accuracy numbers.

2.1 Solution

- Divide complete date set into 5 sets for 5 fold cross validation.
- Use splitData2TestTrain to get these data sets in following order:
  - Fold 1 : Test(0,1) Train ({0,1,2,3,4,5,6,7,8,9} - {0,1})
  - Fold 2 : Test(2,3) Train({0,1,2,3,4,5,6,7,8,9} - {2,3})
  - Fold 3 : Test(4,5) Train({0,1,2,3,4,5,6,7,8,9} - {4,5})
  - Fold 4 : Test(6,7) Train({0,1,2,3,4,5,6,7,8,9} - {5,7})
  - Fold 5 : Test(8,9) Train ({0,1,2,3,4,5,6,7,8,9} - {8,9})
- The Accuracy of the Labels of `test_Vector`is predicted using SVM, Centroid, 5NN, and Linear regression for each fold.
- Output of the accuracy average, with individual accuracy is printed.

2.2 Output

Command: python project1.py B

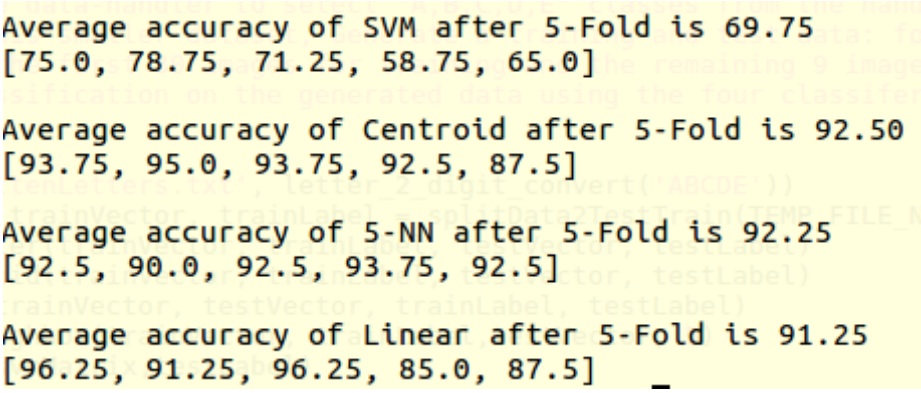

Figure 2: Task B output: showing the average efficiency of accuracy, and for each fold, in predicting class labels using classifications.

3 Project Task C

On handwritten letter data, fix on 10 classes. Use the data handler to generate training and test data files. Do this for seven different splits: (train=5 test=34), (train=10 test=29), (train=15 test=24) ,(train=20 test=19), (train=25 test=24) , (train=30 test=9) , (train=35 test=4). On these seven different cases, run the centroid classifier to compute average test image classification accuracy. Plot these 7 average accuracy on one curve in a figure. What trend can you observe? When do this task, the training data and test data do not need be written into files.

3.1 Solution:

- `letter_2_digit_convert`returns the class ids for letters 'ABCDEFGHIJ' $10 Alphabets$.
- splitData2TestTrain returns the `[test_Vector, test_Label, train_Vector, train_Label]`for following sets:
  - Set 1 : Test(5:38) Train (0:4)
  - Set 2 : Test(10:38) Train (0:9)
  - Set 3 : Test(15:38) Train(0:14)
  - Set 4 : Test(20:38) Train(0:19)
  - Set 5 : Test(25:38) Train (0:24)
  - Set 6 : Test(30:38) Train (0:29)
  - Set 7 : Test(35:38) Train (0:34)
- Accuracy of the Labels of `test_Vector`is predicted using Centroid method for each set.
- Output of the individual accuracy of each case is plotted on graph.

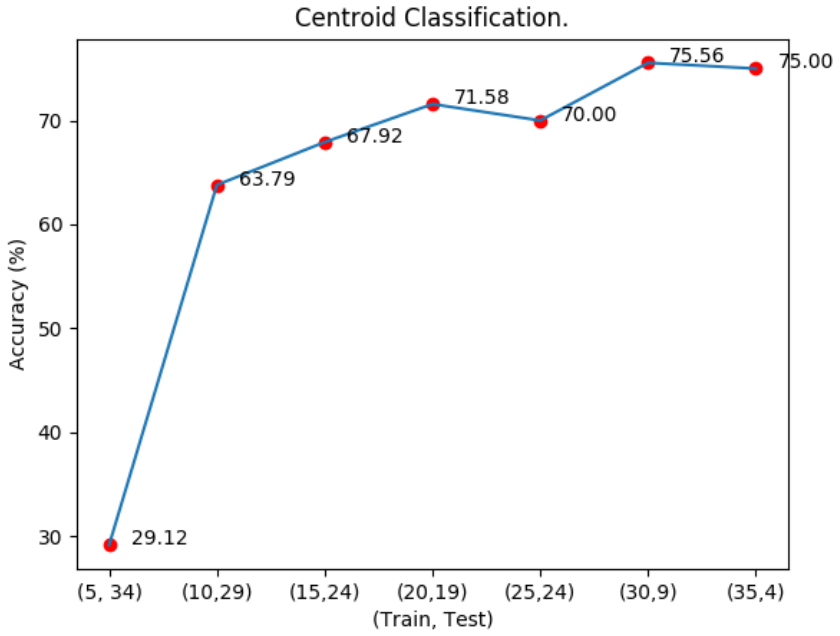

Figure 3: Task C output : The graph show the progress in the efficiency as the train instances increase. Highest Accuracy is achieved when dataset size of test and train is equal.

3.2 Output:

Command : python prohject1.py C

4 Project Task D

On handwritten letter data, fix on 10 classes. Use the data handler to generate training and test data files. Do this for seven different splits: (train=5 test=34), (train=10 test=29), (train=15 test=24) ,(train=20 test=19), (train=25 test=24) , (train=30 test=9) , (train=35 test=4). On these seven different cases, run the centroid classifier to compute average test image classification accuracy. Plot these 7 average accuracy on one curve in a figure. What trend can you observe? When do this task, the training data and test data do not need be written into files.

4.1 Solution:

- `letter_2_digit_convert`returns the class ids for letters 'KLMNOPQRST' 10*Alphabets*.
- splitData2TestTrain returns the `[test_Vector, test_Label, train_Vector, train_Label]`for following sets:

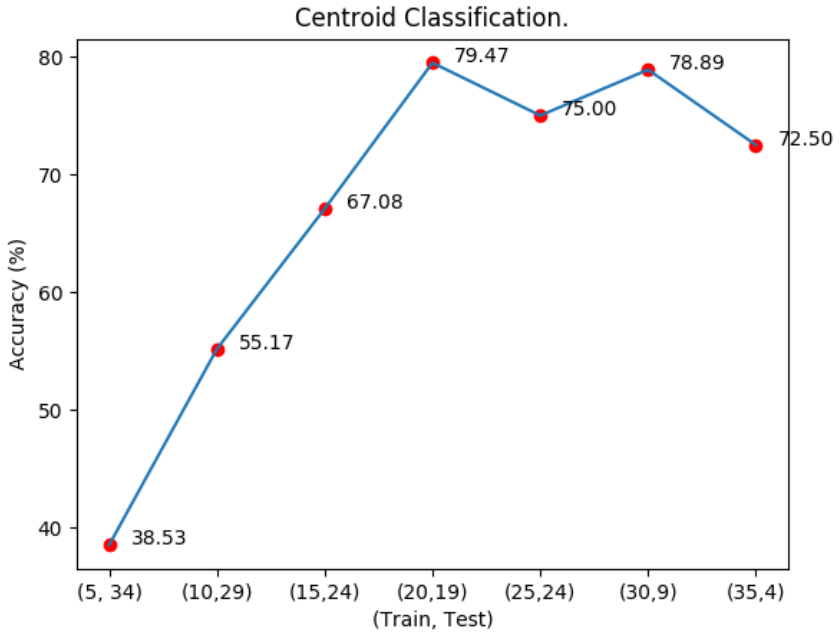

Figure 4: The graph show the progress in the efficiency as the train instances increase. Highest Accuracy is achieved when dataset size of test and train is equal.

- Set 1 : Test(5:38) Train (0:4)
- Set 2 : Test(10:38) Train (0:9)
- Set 3 : Test(15:38) Train(0:14)
- Set 4 : Test(20:38) Train(0:19)
- Set 5 : Test(25:38) Train (0:24)
- Set 6 : Test(30:38) Train (0:29)
- Set 7 : Test(35:38) Train (0:34)

- Accuracy of the Labels of `test_Vector`is predicted using Centroid method for each set.
- Output of the individual accuracy of each case is plotted on graph.

4.2 Output:

Command: python prohject1.py D

5 Project Task E

Write data handlers.

5.1 pickDataClass(filename,class_ids)

This Data handler will help you to "extract a small part of a input data" and generate training data and test data as input to feed into a classifier.
Code :

```
def pickDataClass(filename, class_ids):
    data = np.genfromtxt(filename, delimiter=',')
    listOfClassifierColumn = []
    for i in class_ids:
        a = np.where(data[0] == i)
# returns index locations of the perticular class
        listOfClassifierColumn.extend(np.array(a).tolist
            ())
# appending columns into a string
    listOfClassifierColumn = [item for sublist in
        listOfClassifierColumn for item in sublist]
# forming a array
    np.savetxt(TEMP_FILE_NAME, data[:,
        listOfClassifierColumn], fmt="%i", delimiter=',')
```

5.2 splitData2TestTrain(filename, number_per_class, test_instances)

filename: char_string specifying the data file to read. This can also be an array containing input data.
number_per_class: number of data instances in each class (we assume every class has the same number of data instances)
test_instances: the data instances in each class to be used as test data. We assume that the remaining data instances in each class (after the test data instances are taken out) will be training_instances
Return/output: Training_attributeVector(trainX), Training_labels(trainY), Test_attributeVectors(testX), Test_labels(testY)

```
def splitData2TestTrain(filename, number_per_class,
    test_instances):
    start, end = test_instances.split(":")
    listTest  = list(range(int(start), int(end)+1))
    listTrain = list((set(list(range(0,number_per_class))
        )-set(listTest)))
    Training = []
    Test = []
    data = np.genfromtxt(filename, delimiter=',')
    for i in xrange(0, data[0].size, number_per_class):
        templistTest=[x+i for x in listTest]
        templistTrain=[x+i for x in listTrain]
        templistTest.sort()
        templistTrain.sort()
        if len(Test) == 0:
            Test = data[:,templistTest]
        else:
```

```
            Test= np.concatenate((Test  ,data[:,templistTest]), axis=1)
        if len(Training) == 0:
            Training = data[:,templistTrain]
        else:
            Training= np.concatenate((Training , data[:,templistTrain]), axis=1)

    return Test[1:,], Test[0], Training[1:,], Training[0]
```

5.3 store(vector, label, fileName)

This routine will store (trainX,trainY) into a training data file, and store (testX,testY) into a test data file.

```
def store(trainX, trainY, fileName):
    np.savetxt(fileName, np.vstack((trainY, trainX)), fmt="%i", delimiter=',')
```

5.4 letter_2_digit_convert(str)

Handler converts a character string to an integer array.

```
def letter_2_digit_convert(mystr):
    mylist = []
    mystr = mystr.upper()
    for i in mystr:
        if i.isalpha():
            mylist.append(ord(i)-64)
    return mylist
```