Qeexo Machine Learning Challenge

Basic Finger Sense Classifier

CHETAN KABRA
EMAIL ID: CHETANKABRA8@GMAIL.COM

Qeexo Machine Learning Challenge

Basic Finger Sense Classifier

Summary

The goal of the challenge is to write a program that takes input files describing information about a touch on a mobile device and determines whether the touch is from a finger pad or knuckle.

Dataset :

There are approximately 20,000 training instances (10,255 knuckle and 10,404 pad) and 10,000 test instances (5,263 knuckle and 5,265 pad).

Dataset information in details:

- Each instance represents data from a single finger tap, which contains touch.csv and audio.wav
- An audio.wav contains a sensor response represented as a one-dimensional signal
- Touch.csv: (contains information about a touch)
    - Major axis of an ellipse
    - Minor axis of an ellipse
    - X
    - Y
    - Orientation
    - Pressure
- Each touch may be from a pad or knuckle.

Implementation

1. Exploring Dataset:

Started with audio.wav:
Used scipy.io.wavfile to convert the audio file into 256 Feature vector.

- Max_value : 32767.0
- Min_value : -32767.0

Touch.csv: contains information related to touch

- Major axis of an ellipse : max_value : 11 , min_value = 2.0
- Minor axis of an ellipse: max_value : 10 , min_value = 2.0
- X : max_value : 1079 , min_value = 0.0
- Y: max_value : 1919 , min_value = 0.0
- Orientation : max_value : -1 , min_value = -1.0
- Pressure : max_value : 0.0 , min_value = 0.0

2. Feature Scaling and Feature Selection

1. For feature scaling used Z-score normalization technique:

$$Normalized\ (e_i) = \frac{e_i - \bar{E}}{std(E)}$$

2. Feature selection as I don't have much choice I have selected below feature set for my classifier:
   a. 256 feature vector of audio.wav
   b. Touch.csv (major, minor, x and y)
3. I have not used orientationa and pressure because they have zero variance data values.
4. Created a feature vectore with (size of the dataset X 260)

3. Creating K Fold Cross – Validation set:

Implemented K fold cross -validation and divided the training dataset into equal K size of training and testing or(validation dataset).

To perform various experiments and to analyze the classifier used 10 fold cross validation.

4. Experiments:

1. KNN :

Started with simple K – Nearest Neighbours classifier and tried to test how well my dataset is working with this classifier.

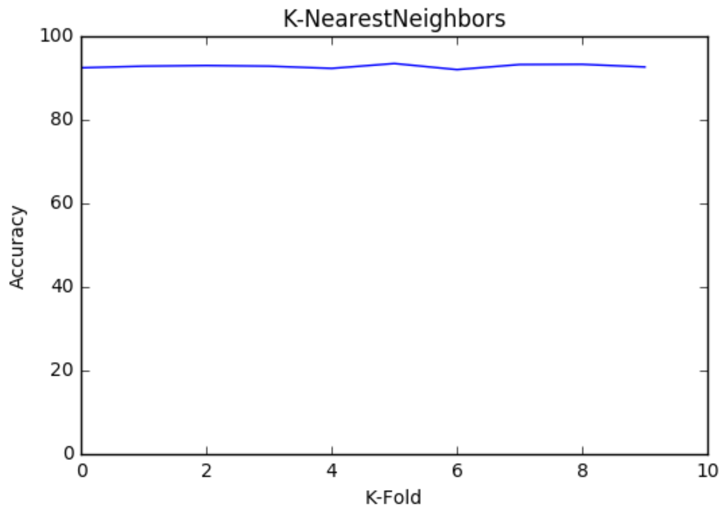
Used euclidean distance to calculate the distance between each feature and then taking the K nearest minimum distance to predict the label for that instance.

Used Grid SearchCV from sklearn to find the best K value for a given dataset

As per the result I have got best K =1 for the cross validation set.

Average Accuracy : 92.6537530266

Line Graph:



Average Confusion Matrix :

	Predicted Zero (Pad)	**Predicted One (Knuckle)**
Actual Class 0 (Pad)	1029	122
Actual Class 1 (Knuckle)	33	881

2. Logistic Regression:

Choose this classifier as this is simple one and mostly the output lies between 0 and 1.

As the hyposis function of the Logistic Regression is sigmoid function.

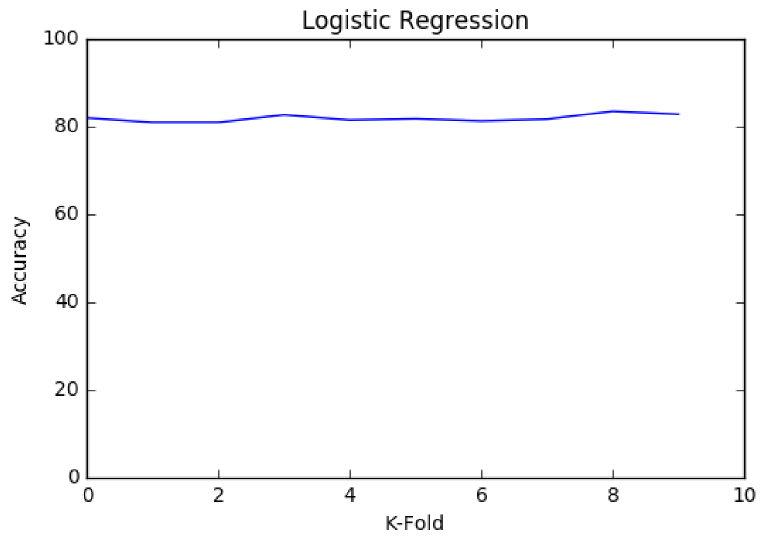
Used logisticRegression library from sklearn.

Finding:

Logistic regression was fast to compute results, but accuracy was not up to the mark.

Average Accuracy : 81.7723970944%

Line Graph:



Average Confusion Matrix :

	Predicted Zero (Pad)	**Predicted One (Knuckle)**
Actual Class 0 (Pad)	906	202
Actual Class 1 (Knuckle)	156	801

3. Support Vector Machine:

I have used my own SVM library for Linear Kernal:

Below are details how I implemented SVM using CVXOPT.

The CVXOPT function in python was used for calculating the values of Lagrange multiplier.

After getting this multipliers we calculated the hyperplane using following formulation

$$W = \sum_{i=1}^{n} \propto_i y_i X_i$$

In this we know yi is classlabel and and xi is image.

Next term to be calculated is b which can be calculated by modifying our orginal formula which was

$$y_i(W^T X_i + b) \geq 1$$

we can modify the above equation as $b = \frac{1}{y_i} - W^T X_i$

In this formulation yi and Xi has to be from positive class to get the better results. We give the value of data of positve class and calculate 'b'.

Average Accuracy : 88.15151 %

4. SVM RBF and Poly Kernel:

Used SVM RBF ad Poly libraries provided by Sklearn

Moving forward with my experiment used SVM used GridSearchcv to find the best kernel and other tunning parameter to enhance my classifier.

Note: chose class weight {1:1.2} because due to skewed classes in the training dataset

Used below tunning parameters:

tuned_parameters = [

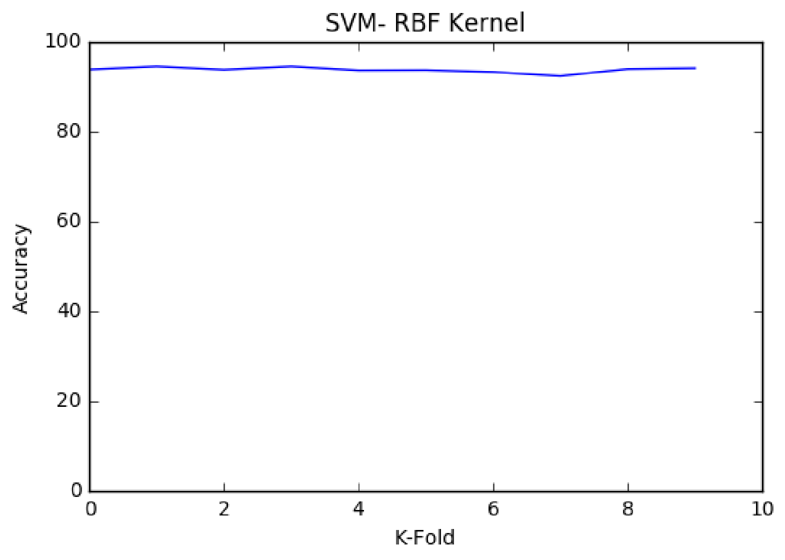
{'kernel' : ['rbf'], 'gamma': [0.1, 1e-2], 'C': [10, 100], 'class_weight': [{1:1.0145294978059483},{1:1}]},

{'kernel' : ['poly'], 'degree' : [5, 9], 'C' : [1, 10,100], 'class_weight': [{1:1.0145294978059483}],{1:1}}]

Average Accuracy : 93.6368038741 %

Average Confusion Matrix :

	Predicted Zero (Pad)	**Predicted One (Knuckle)**
Actual Class 0 (Pad)	994	99
Actual Class 1 (Knuckle)	44	928



Results and Analysis

As per my findings and above result I choose SVM with RBF kernel and with below parameters. {'kernel': 'rbf', 'C': 10, 'gamma': 0.01, 'class_weight': {1: 1.2}}.

Test Data Result analysis:

	Actual	**Predicted**
Actual Class 0 (Pad)	5265	5913
Actual Class 1 (Knuckle)	5263	4615

Predicted Accuray :

(looking at the confusion matrix and taking average false negative and false postive concluded below accuracy , though not sure I am just playing with it ☺)

It should be in the range of 89 – 92%