Natural Language Processing Task 3

Speech Phoneme Analysis and Classification

Name: Valerija Holomjova

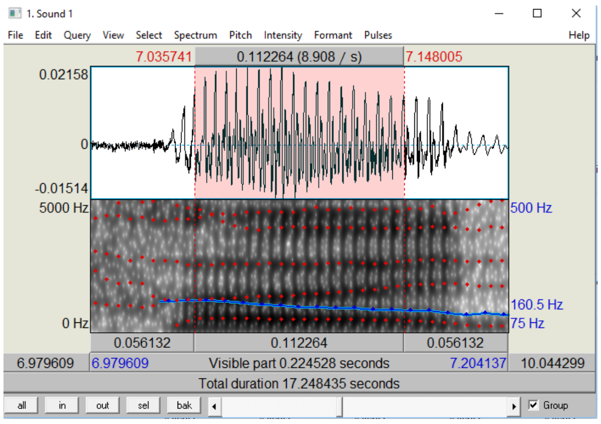
Course Code: ICS2203-SEM2-A-1819

Date: 16/04/19

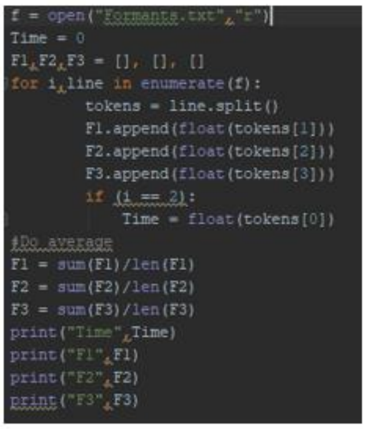
Table of Contents

Collecting the Data

I decided to choose the words “Heed”, “had” and “hood” (The first, fourth and tenth word) for my feature extraction. These had the phonemes “IY”, “AE” and “UH” respectively which were obtained using the following site[1]. I extracted the features using the Pratt[2] software and isolated the vowels of each of the words for each selected candidate and retrieve the formant listings. I would then navigate to the middle of the formant listings and extract 5 rows and calculate the average of the F1, F2 and F3 values based on these rows.

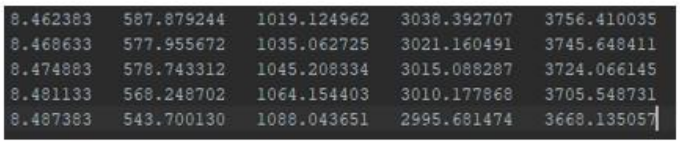


To speed up the process of calculating the average I made a short algorithm in python where I paste the five rows of formant values from Pratt into a text file and it displays the middle time and averages for me to copy and paste into my excel sheet. Below is a screenshot of the code, an example input of 5 middle rows of a vowel’s formant listing and the output of this short algorithm.



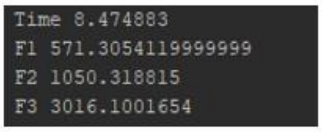
```python
f = open("Formants.txt","r")
Time = 0
F1,F2,F3 = [], [], []
for i,line in enumerate(f):
        tokens = line.split()
        F1.append(float(tokens[1]))
        F2.append(float(tokens[2]))
        F3.append(float(tokens[3]))
        if (i == 2):
            Time = float(tokens[0])
#Do average
F1 = sum(F1)/len(F1)
F2 = sum(F2)/len(F2)
F3 = sum(F3)/len(F3)
print("Time",Time)
print("F1",F1)
print("F2",F2)
print("F3",F3)
```

(A screenshot of the Code)



```
8.462383   587.879244   1019.124962   3038.392707   3756.410035
8.468633   577.955672   1035.062725   3021.160491   3745.648411
8.474883   578.743312   1045.208334   3015.088287   3724.066145
8.481133   568.248702   1064.154403   3010.177868   3705.548731
8.487383   543.700130   1088.043651   2995.681474   3668.135057
```

(A screenshot of the Input)



```
Time 8.474883
F1 571.3054119999999
F2 1050.318815
F3 3016.1001654
```

(A screenshot of the Output)

Overview of the Code

Runner() function

This is the main function which calls the other functions to perform KNN clustering. It obtains the data and labels denoted by X and y respectively from the ‘openFile()’ function and then preforms KNN for five times, each time with a different random seed generator (so that the testing and training set values are different). Please note that the value of the testing set percentage, the value of k and the distance metric used is modified here.

[1] http://www.speech.cs.cmu.edu/cgi-bin/cmudict
[2] http://www.fon.hum.uva.nl/praat/

openFile() function

The following function opens the CSV data file and goes through each row. From each row, the F1, F2, F3 values are extracted and appended as a list for the X (data) list variable. The class ID is also extracted and appended to the y (labels) list variable. These two list variables denoting the data and labels of the CSV are then returned.

doKNN() function

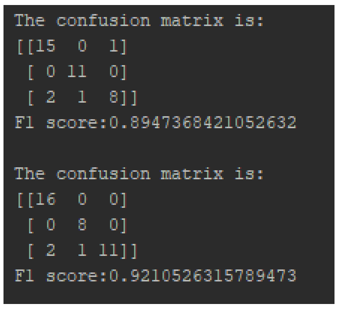
This function performs KNN clustering by implementing the library functions from the sklearn[3] library. It starts off by splitting the data and labels into the testing and training set given a percentage and random seed value. The function proceeds to build a KNN classifier given a k value and distance metric. Please note these given values are passed in from the 'runner()' function. The model is then trained using the training set and a list of predictions of the phenome class of the testing set is obtained using the trained model. At the end of the function the 'buildConfusionMatrix()' is called to evaluate the results of the KNN classifier by using the predicted classes of the testing set and display the f1 score.

buildConfusionMatrix()

This function implements the functions given by the sklearn to display the confusion matrix and f1 score for a given list of predictions and actual classes.

Final Output of Code

5 different results of the performance of KNN classifier are produced with a different training and testing set in each result. Each result displays the confusion matrix and the F1 score. The snippet of the right displays two results from the classifier using a k value of 3, the Euclidean distance metric and a random seed value of 0 and 1.



```
The confusion matrix is:
[[15  0  1]
 [ 0 11  0]
 [ 2  1  8]]
F1 score:0.8947368421052632

The confusion matrix is:
[[16  0  0]
 [ 0  8  0]
 [ 2  1 11]]
F1 score:0.9210526315789473
```

Questions and Evaluation

1. How does the performance change with different values of K?

I tested out the algorithm with different values of k using the Euclidean distance metric and random seeds value ranging to 0-4. Each time I ran the algorithm, 5 results are produced so I wrote down their F1 scores and calculated their average (as shown in the figure below) . I started with a k value of 2 which seemed to have yielded the lowest results. As I increased the k-value, the F1 score become greater until I reached a k-value of 6 to which it started to decrease. Hence it seems it performs the best with a k-value of 5.

K value	Result 1	Result 2	Result 3	Result 4	Result 5	Average
2	0.8947	0.8684	0.8684	0.8421	0.7895	0.853
3	0.8947	0.9210	0.9737	0.8947	0.8421	0.905
4	0.9210	0.8947	0.9474	0.8684	0.8684	0.9
5	0.8947	0.8684	0.9474	0.9474	0.8947	0.911
6	0.8947	0.8421	0.9737	0.8947	0.8684	0.895

[3] https://scikit-learn.org/stable/

2. What distance metric did you use? Tried any others?

Initially, I tested out the algorithm using the Euclidean distance metric as it is the most familiar for me. I proceeded to test out the algorithm with some different distance metrics that I obtained from the following page[4] using a k-value of 5 and listed the results below to compare. From the results obtained below, it shows that the Manhattan (City block) distance metric provided the most accurate results. In fact, a paper[5] I found comparing distance metrics for Phoneme Classification yielded the same concluding results – that the city block distance provided "good phoneme classification accuracy" whilst being "computationally efficient".

Metric	Result 1	Result 2	Result 3	Result 4	Result 5	Average
Euclidean	0.8947	0.8684	0.9474	0.9474	0.8947	0.911
Manhattan	0.9474	0.9211	0.9737	0.8947	0.8947	0.926
Chebyshev	0.9474	0.8684	0.9474	0.9211	0.8947	0.916

3. How does the performance change when classification is done on data for a single gender alone or when data from both genders are put together?

I modified the 'openFile()' function so that I could compare the results of the algorithm by extracting only the male, female or both gender entries from the CSV file and listed their F1 scores below. Please note that a k-value of 5 and the Manhattan distance metric was used as they yielded the highest accuracy results in the previous questions. I also kept the testing set percentage as 25%. From the results obtained it seems like using only female entries yielded the highest accuracy results while using only male entries yielded the lowest accuracy results. This could be due to the pitch difference between the male and female voice perhaps.

Data Type	Result 1	Result 2	Result 3	Result 4	Result 5	Average
Female	1.0	0.9474	0.9474	0.9474	1.0	0.968
Male	0.8421	0.8947	0.9474	0.8421	0.8947	0.884
Both	0.9474	0.9211	0.9737	0.8947	0.8947	0.926

4. What are the vowel-based phonemes that produce the most confusion?

Using a k-value of 5, a Manhattan distance metric, data from both genders and random seed values from 0 to 4, the following confusion matrix and F1 Scores were produced. The tally counts the frequency of false positives and false negatives for each phoneme based on the given confusion matrix. The 'AE' phoneme from the word "had" seemed to have the highest tally, hence the most confusion, whilst the 'UH' phoneme from the word "hood" seems to have the lowest tally and hence the least confusion. This could be to the trap-bath split (a vowel split that occurs mainly in mainstream and south-eastern accents in England). As a result, the "AE" phoneme is lengthened or changed to the sound of "A:" in some accents which could be the cause for such high confusion.

[4] https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.DistanceMetric.html

[5] https://pdfs.semanticscholar.org/c83b/e3e5b10871f5fb64367f8d87114912ad3ddc.pdf

Result Type	Result 1	Result 2	Result 3	Result 4	Result 5
Confusion Matrix	[[16 0 0] [0 11 0] [1 1 9]]	[[15 0 1] [0 8 0] [1 1 12]]	[[17 0 0] [0 7 0] [0 1 13]]	[[10 0 1] [0 12 0] [0 3 12]]	[[11 0 1] [0 8 0] [2 1 15]]
F1 score	0.9474	0.9211	0.9737	0.8947	0.8947
Tally	IY – 1 AE – 1 UH – 0	IY – 1 AE – 1 UH – 1	IY – 0 AE – 1 UH – 0	IY – 0 AE – 3 UH – 1	IY – 2 AE – 1 UH – 1