HOMEWORK 6

In this lab you will do some unsupervised learning in the domain of phonetics. You will cluster vowel tokens into classes on the basis of acoustic features using K-means.

PART I

The Peterson/Barney vowel data consist of measurements of four acoustic features, F0, F1, F2 and F3 values, for two repetitions of 10 different vowels by 76 speakers of British English (33 adult males, 28 adult females, and 15 children). There are therefore 1520 vowel tokens; each row represents a vowel, and there are 7 features (columns) for each token as follows:

column	*information*	*key*
1	Speaker type	1=male, 2=female, 3=child
2	Speaker number	a unique id for each speaker: 1-76
3	Vowel identity	1 = i, 2 = ɪ, 3 = e, 4 = æ, 5 = ʌ, 6 = ɑ, 7 = ɔ, 8 = ʊ, 9 = u, 10 = ɜ
4	F0	Fundamental frequency
5	F1	First formant
6	F2	Second formant
7	F3	Third formant

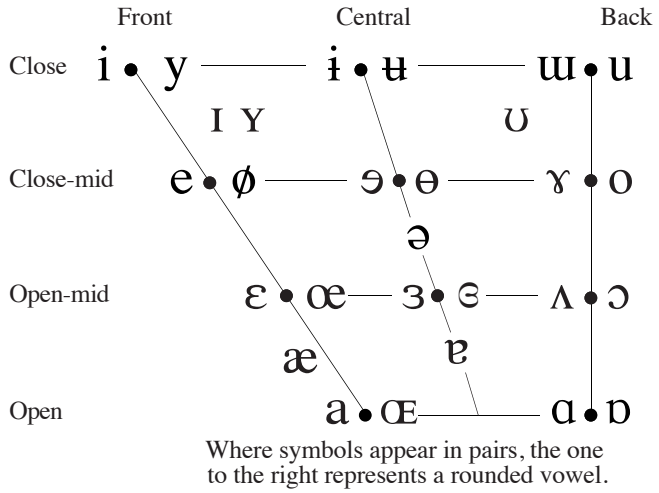

Figure 1: Articulatory Vowel Chart

Vowels of the world's languages can be categorized on the basis of various articulatory or acoustic features. One standard articulatory-based representation of the vowel space is shown Figure 1. This chart represents the relative frontness (left/right) and relative height (up/down) of the tongue. Vowels in the upper left are high and front; vowels in the lower right are low and back. This chart also represents roundness of the lips, but we won't be looking at this feature in this assignment.

The four measurements provided in the data for each vowel token are the fundamental frequency (in Hertz) and the frequencies of the first three formants. Formants are peaks in the acoustic frequency spectrum and depend on the shape of the vocal tract at the time of production. F0, or fundamental frequency represents the pitch of the speaker's voice. The first two formants,

F1 and F2, are the main acoustic features we use to disambiguate vowels because each vowel has different prototypical F1 and F2 values (though see discussion below).

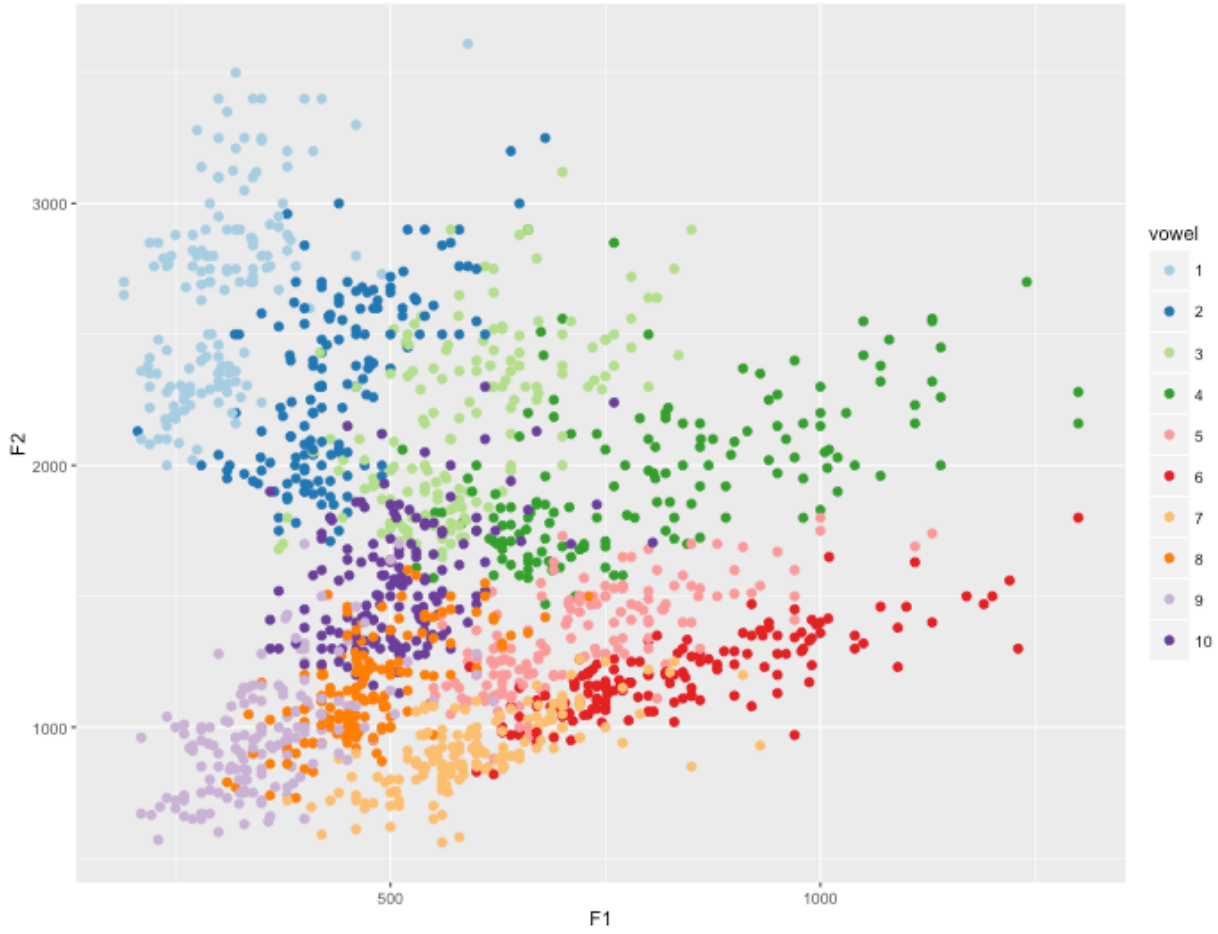

Figure 2: Scatterplot of F1 vs F2, labeled by vowel identity

The main programming portion of the assignment will be to implement k-means clustering using F1 and F2 as the features. Figure 2 shows you what the data look like, and how well you can possibly hope to do using this technique.

As you can see, the different vowels do form fairly clear clusters. For example, a token with f1=900 and f2=2000 is clearly a token of vowel 4 (æ); whereas a token with f1=200 and f2=2700 is clear a token of vowel 1 (i). In fact, by comparing the scatterplot in Figure 2 to the vowel chart in Figure 1, you can see a clear correspondence between vowel height and formant 1, on the one hand, and vowel backness and formant 2 on the other.

Nonetheless, these clusters have a non-trivial amount of overlap and are not particularly well separated. For example, if we see a token with f1=600 and f2=2000, it could be any one of 3 possible vowels. There's no way to tell definitively on the basis of this information alone which cluster the vowel should belong to. Therefore, there's no hope that clustering using these two features alone can group these ten vowels into clusters that correspond *exactly* to the true vowel labels. To see how difficult the problem we're trying to solve really is, look at the scatterplot in Figure 3. This is exactly the same as Figure 2 with the labels/colors removed. This is the

information available to the algorithm. It's supposed group all these dots into 10 clusters. How well can this technique possibly work? It is the goal of this homework to answer that question.

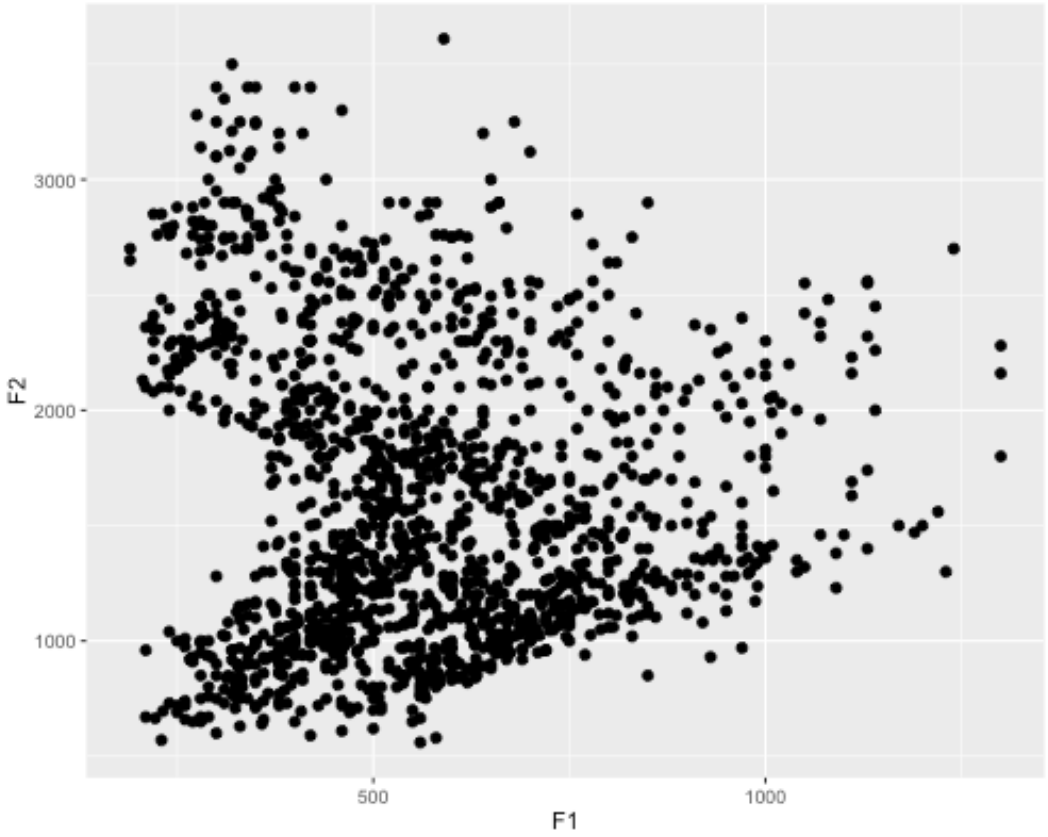

Figure 3: Vowel space - f1 vs f2

Why aren't the clusters perfectly separated? Variation. Each person speaks a bit differently; even the same person will pronounce things slightly differently on different speaking occasions. One major factor is the fundamental frequency (F0) of a person's speaking voice. As we can see from the scatterplot in Figure 4, male speakers have lower pitch than female speakers, who have lower pitch overall than children. From Figure 4, we can also see that there is a relationship between fundamental frequency and values of the other formants, in this case F1. In general, the higher the F0, the more stretched and shifted upward the range of F1. In other words, the identity of vowels is relative: different speakers have different vowel spaces, with fundamental frequency being a major factor. That's one reason there is so much overlap and noise in the scatterplot in Figure 2.

Figure 4: Scatterplot of F0 vs F1, labeled by speaker type

Ideally, we'd like to incorporate our knowledge about the relationship between fundamental frequency and the other formants to help us cluster vowels. But as a starting point, we will focus just on the information contained in the F1 and F2 values. You will then be asked to think about how the fundamental frequency information could be used to improve performance and try some of these extensions.

There is nothing to turn in for this part; just make sure you understand the task and potential issues before continuing.

PART II

Implement k-means clustering. Your program **(kmeans.py)** should read in the 'voweldata.txt' file and a number (k: how many clusters to find) and an optional initialization file (in a format like 'good_init.txt'). If the third argument is missing, your program should choose k random initial means. If the third argument is there, your program should use the means from the file as its initial means.

Your program should print out the values of the k means it settles on. That is, it should print out the k centroids it has learned. Your program should only look at columns 5 and 6 from the data – it does not know anything about the true identity of the tokens at this point!

Some details:

- k does not mean 10
    - Make sure your program is flexible enough to work with any integer k, not just 10.

- How to initialize?
  - You should begin with k random means. How random? They should be means that fall in the overall ranges of each of the formants. The first value should be somewhere between the min and max of the F1 values in the data, and the second value should be somewhere between the min and max of the F2 values in the data.
- How many times to iterate?
  - Iterate until convergence. That is, iterate until the clusters stop changing. You can put a simple check into your loop to see whether any clusters were redefined.
- How to resolve ties?
  - What do you do if some token is exactly equidistant to two means? Make an arbitrary or random choice.
- What if a mean gets no points?
  - Make sure to handle the case when a mean has no points assigned to it. You should pick new random values for this mean.

To test your code: if you run your program on "voweldata.txt" with the initial means in "good_init.txt", your program should find the following 10 means in the following order:

```
407.651162790698        2888.02325581395
420.379120879121        2365.75274725275
605.644628099174        919.190082644628
857.561538461538        1542.6
744.278350515464        1176.4587628866
881.381443298969        2218.71134020619
513.816037735849        1864.64150943396
515.994011976048        1425.46706586826
433.227272727273        1081.17045454545
367.151785714286        777.508928571429
```

In other words, the initial mean listed on the first line of "good_init.txt" should readjust into the first mean above, the second line into the second line above, and so on.

PART III

Now you will extend your program to evaluate its performance. One way to evaluate the goodness of a clustering is by comparing its same vs. different judgments to those of the target. That is, for any pair of distinct vowels v_1 and v_2, check whether or not the clustering puts them in the same cluster, and whether or not this is correct (by looking at the true vowel identities). This will give us four kinds of cases:

- *true positives*: v_1 and v_2 are in the same cluster and are the same vowel
- *false positives*: v_1 and v_2 are in the same cluster but are not the same vowel
- *true negatives*: v_1 and v_2 are in different clusters and are not the same vowel
- *false negatives*: v_1 and v_2 are in different clusters but are the same vowel

By calculating how many pairs of vowels fall into each of these cases, we can compute a trio of measures that are commonly used to evaluate performance on classification tasks:

- precision = (# true positives) / (# true positives + # false positives)
- recall = (# true positives) / (# true positives + #false negatives)
- f-score = (2*precision*recall) / (precision+recall)

Extend your program to calculate and print the three measures above using the final clusters learned by your program. To do so, you will need to compare the result of clustering to the true identities of each vowel stored in the third column.

If you correctly implement precision, recall, and f-score, you should get the following values when you initialize with good_init.txt:

- Precision: .38491019146
- Recall: .407825026142
- Fscore: .396036420255

Q1

Run your code 10 or more times with random initializations. What range do you observe for each of precision, recall, and f-score? What is the f-score of the best clustering you found?

Q2

Run your program on the child, female, and male data separately. You can do this by either having your program temporarily ignore certain lines of input, or by separating out the vowel data into three input files. What are the best precision, recall, and f-score values you get for each of these three inputs? Why are the results different from running the program on all the data at once?

Q3

Modify your code so that F1 and F2 are represented in 'mels' instead of hertz. See the following for a definition of mels:

http://en.wikipedia.org/wiki/Mel_scale

What effect does this have? Why do you think this happens and what is the cognitive significance of this (hint: think about the shape of the clusters in both spaces)?

Q4

Here's your chance to be creative! Extend your program in some way to take advantage of the information contained in the first, second, or fourth columns of the data. *Explain clearly what changes you made, why, and what effect they had on the accuracy of your program*. *Submit the new version of your program* with a new name.

What to turn in

- Answers to questions Q1-Q4 in Part III in PDF format
- Your final code (kmeans.py) after making the modification in Q3
- Your modified code (kmeansQ4.py) for question Q4.