Classification of Spoken Digits via K-Means and Expectation Maximization Model

Evan Glas

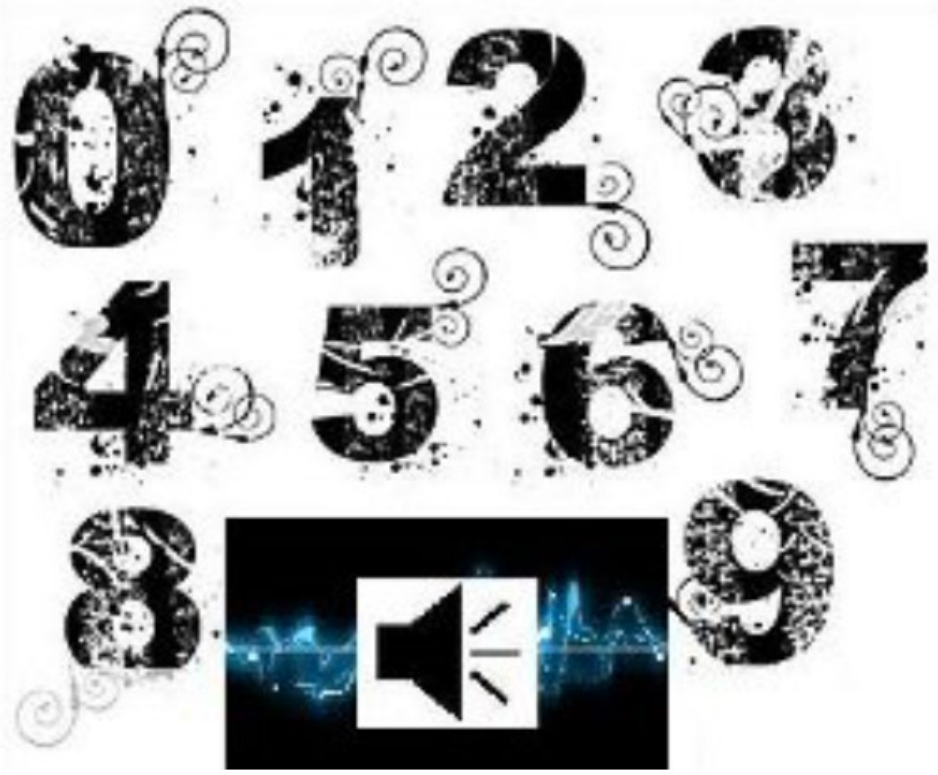
Abstract

In this project, I explored the use of various modeling choices in order to classify spoken Arabic digits according to generated Cepstral Coefficients. My first model consisted of a Gaussian Mixture Model with mixture components found through k-means on the training data. I then implement eda Gaussian Mixture Model with mixture components calculated through an expectation maximization algorithm. Next, I applied another Gaussian Mixture Model, except this time considering the time-dependent aspect of the data. I achieved a maximum of 96% accuracy across all models, with each iteration demonstrating improvement over the last. Future work may explore the application of recurrent neural networks/ reinforcement Learning given the time-dependent nature of the data.

Image Source: https://www.arabicpod101.com/blog/2019/10/24/arabic-numbers/

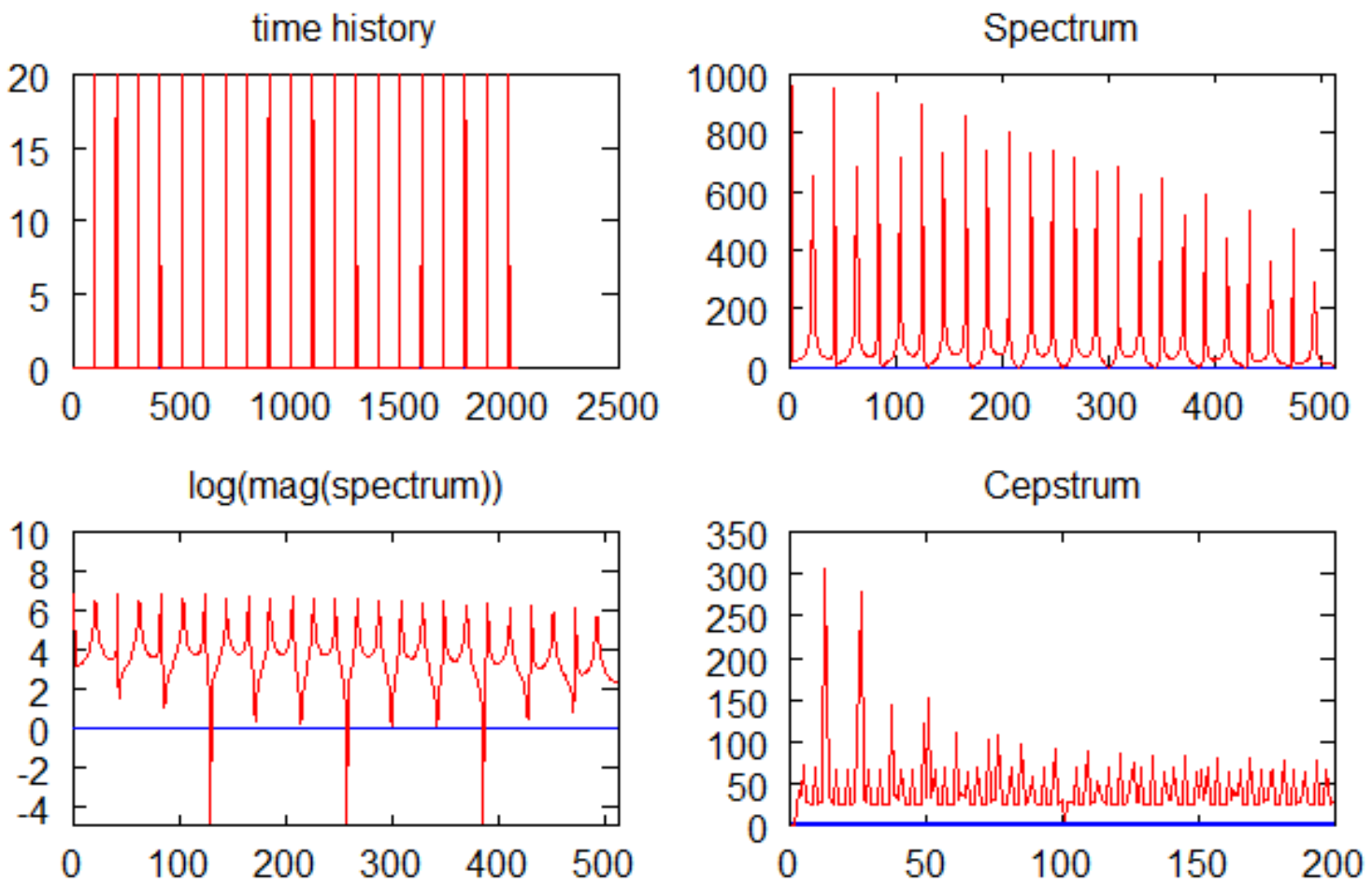
Data Set

This project used data from a dataset titled “Spoken Arabic Digits”. The dataset contains the top 13 Cepstral Coefficients for 8800 individual speech tokens of the digits 0-9 spoken in Arabic. The data was generated from 88 unique speakers (44 male, 44 female), who each recorded every digit from 0 – 9 10 times. Each speech token consists of 4-90 samples of Cepstral Coefficients, although most tokens contain about 30 individual samples. Thus, a typical token would consist of the highest 13 Cepstral Coefficients across about 30 sequential samples in time. This gives a total of about 390 individual values which represent each speech token.



Data Set/Image Source: https://archive.ics.uci.edu/ml/datasets/Spoken+Arabic+Digit

Cepstral Coefficients



Image/Equation Source: https://en.wikipedia.org/wiki/Cepstrum#cite_note-Bogert_1963-2

"Cepstral Coefficients" are a discrete time representation of the degree of harmonicity within the harmonicity for a given signal (generally audio). More formaly, the "cepstrum" is defined using the following formula, where f(t) is the original input signal.

$$C_p = \left| \mathcal{F}^{-1} \left\{ \log\left(\left| \mathcal{F}\{f(t)\} \right|^2 \right) \right\} \right|^2$$

The diagram on the left demonstrate the stages to find the cepstrum of a given signal. First, the original signal is coverted to the frequency domain. Next, one must take the log of this signal (so as to avoid returning the original signal in the next step). Finally, the cepstrum is obtained through the by applying an inverse Fourier transform.

Objectives

- I set three main objectives for my completion of this project.

1. Learn how to apply, interpret, and ultimately refine modify various probabilistic modelling techniques in the context of given data.
2. As a byproduct of the above objective, build a classifier which minimizes the error rate on the test set of the Arabic Spoken Digits Set.
3. Compute additional features/modify the data in some way to achieve better results.

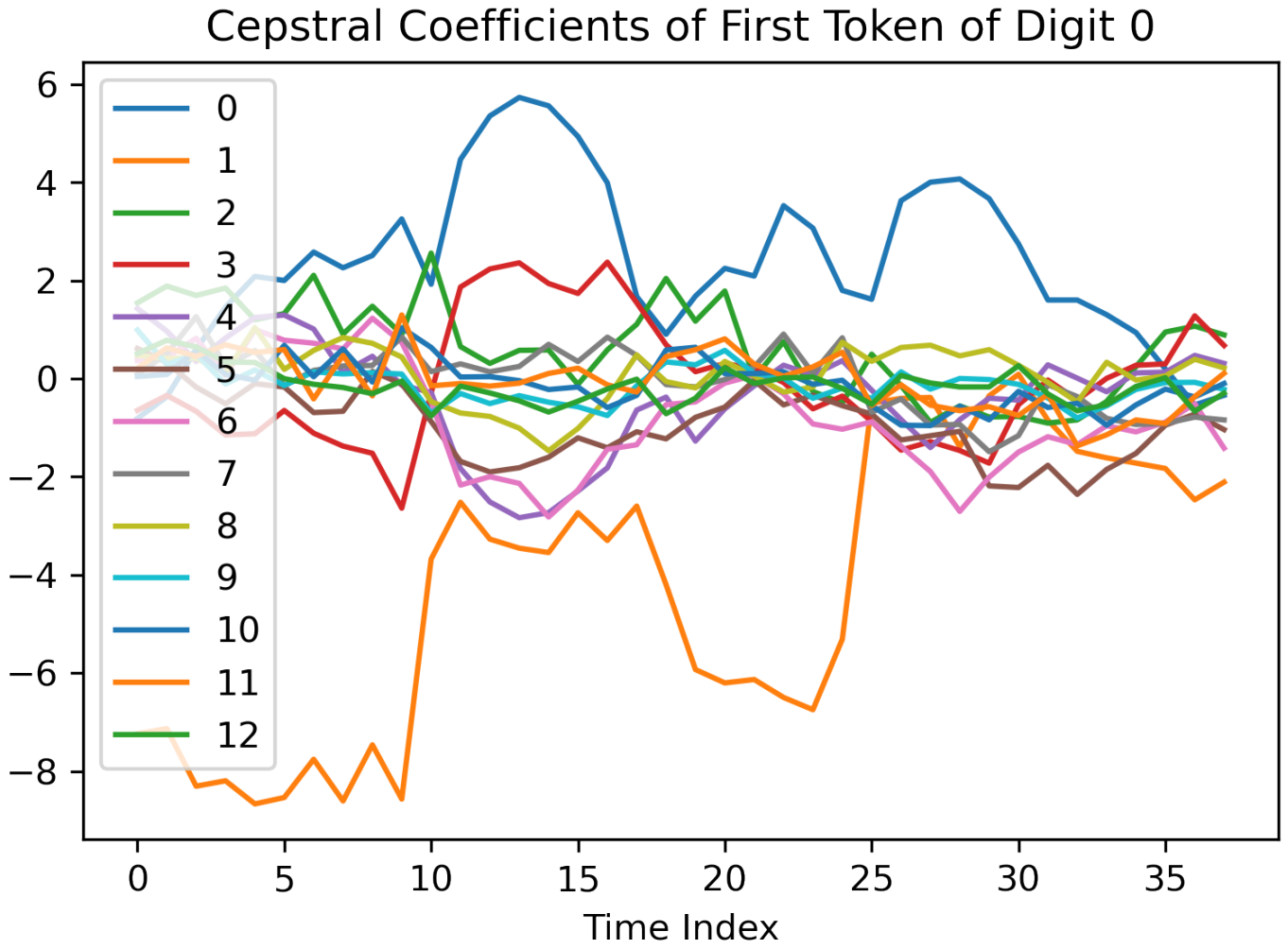
EDA

	Values for Training Data
Count of Tokens	6599
Means Number of Samples per Token	39.89
Standard Deviation of Samples per Token	8.72
Minimum Number of Samples	4
Maximum Number of Samples	93
Median Number of Samples per Token	39
75% Value of Number of Tokens	45

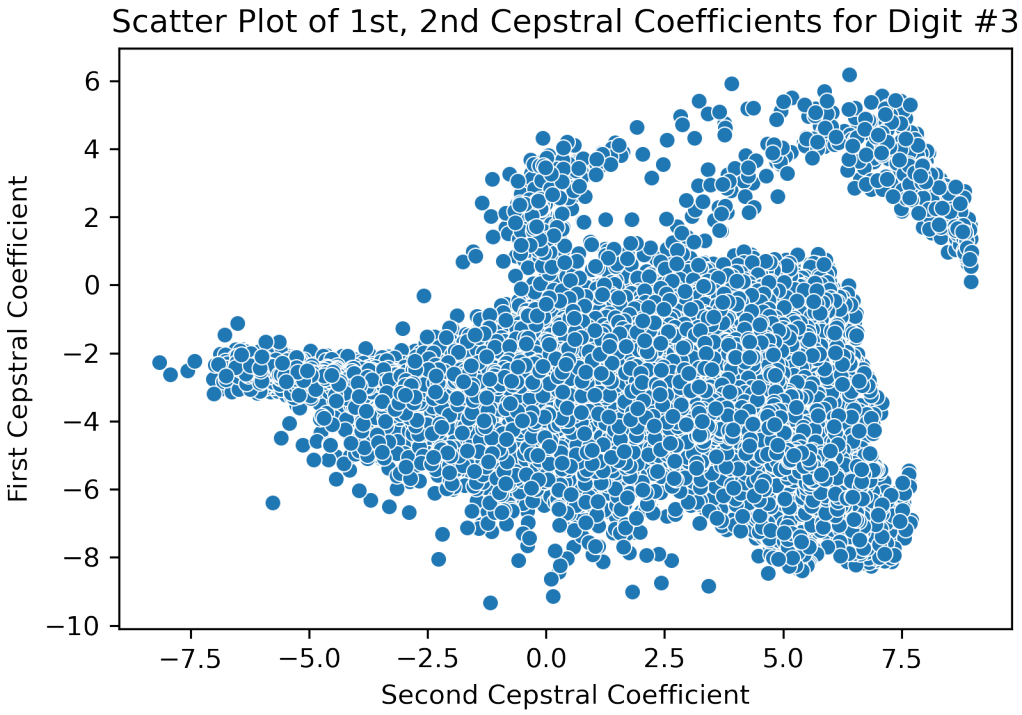
I began my analysis by considering the structure of the given data. My first inclination when presented with the data was to perhaps use correlation or to just “match” each sample with a representative mean from each digit. However, I found this would be more difficult that originally thought given the range of number of samples per token. As such, each sample exists on its own time-frame, making temporal comparisons more difficult. I chose to proceed with just spatial analysis of the data without considering the cepstral coefficient index.

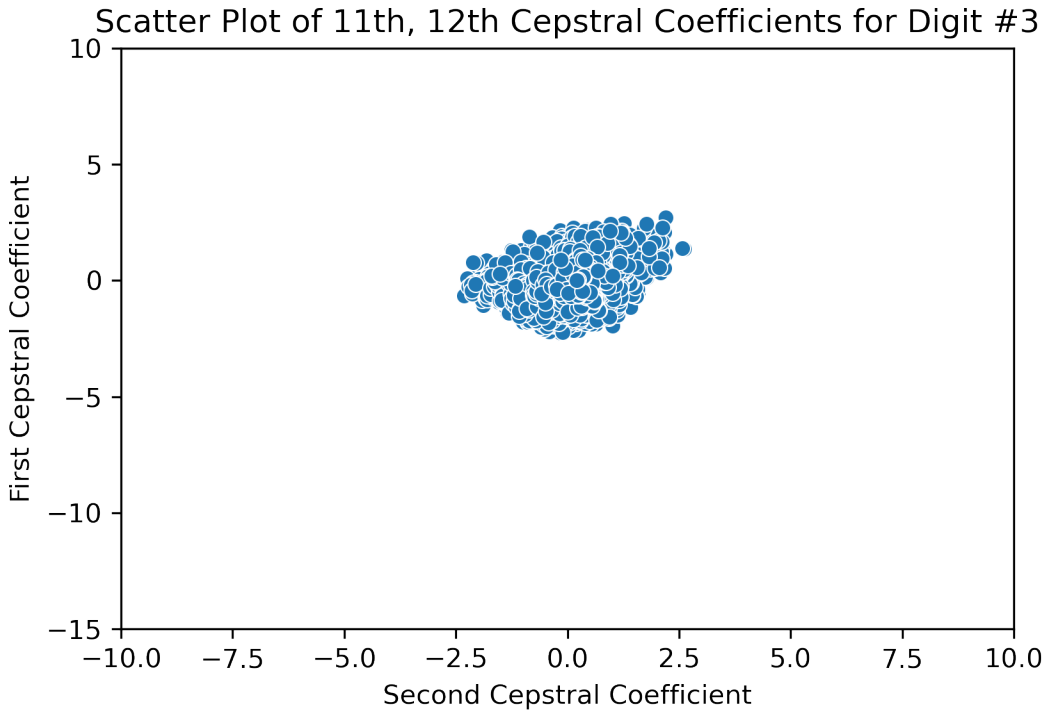
EDA

The plot to the right depicts the values of all 13 Cepstral Coefficients across all time indices for a given speech token. One may see the high degree of variation across time for Cepstral Coefficients 0-4, while the remaining coefficients remains relatively stable about the value 0 on the y-axis. As the coefficients are already sorted by relative importance, this gives further evidence that potentially only a few Cepstral Coefficients will prove valuable at classification time.



EDA

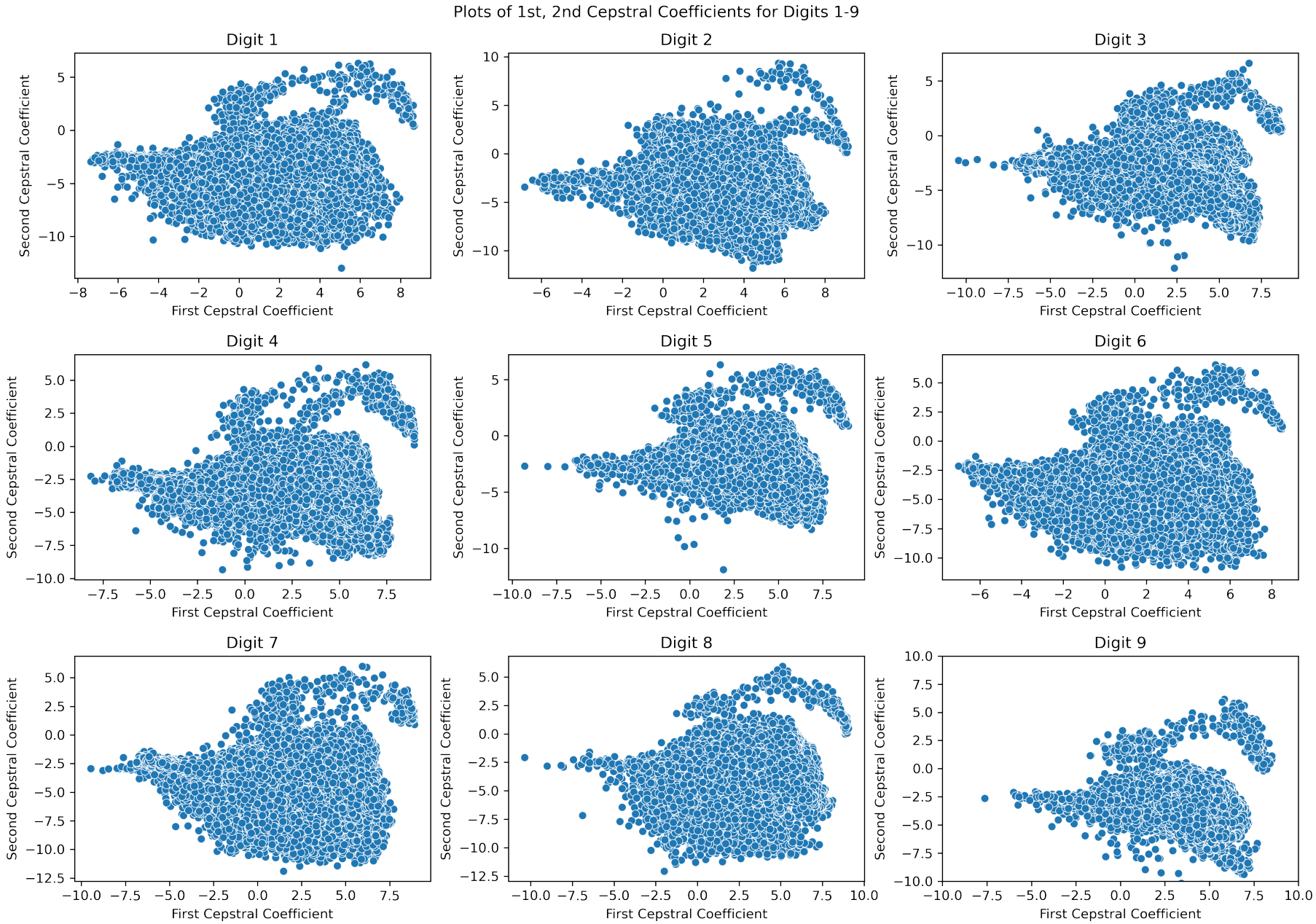




To the left is a scatter plot of the values of the two most significant Cepstral Coefficients across all tokens of a given digit in the training set. As seen, the first two cepstral coefficients certainly display a high degree of non-random variability. On the other hand, the plot on the right, which shows the 11^{th} and 12^{th} Cepstral Coefficients, reveals a more “blob-like” distribution centered around the origin. This again supports the notion that higher Cepstral Coefficients (above 4) begin to resemble noise rather than valuable data.

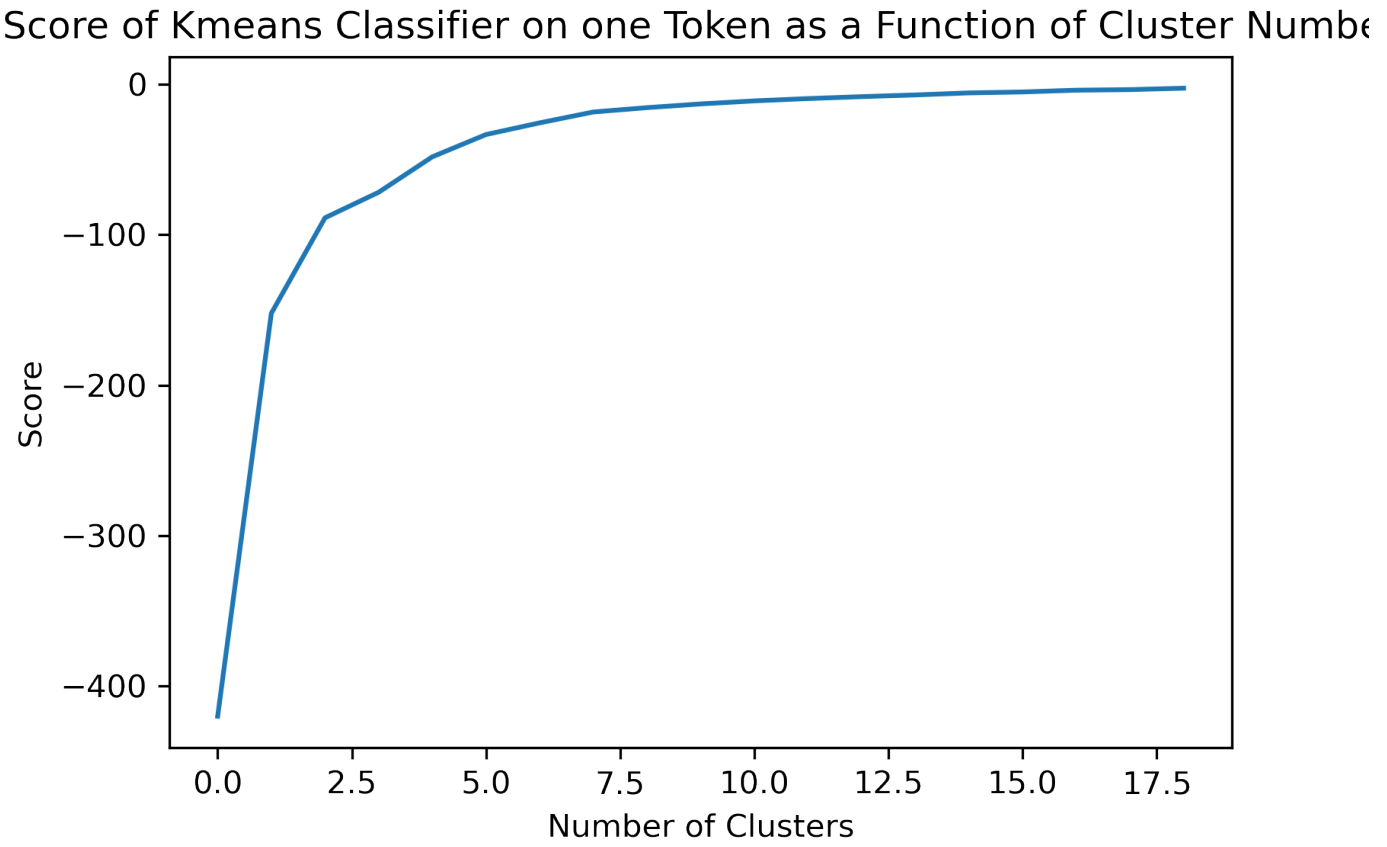
EDA

To the right are the scatter plots of the first and second cepstral coefficients the other 9 digits in the training set. As seen, all data sets appear to take on a roughly similar “anglerfish-like” shape, with notable differences such as the density of points moving to the bottom left of the plot (i.e. digit 5 appears cut off on the bottom, whereas digit 8 is more rounded).



K-Means Gaussian Mixture Model

I proceeded to fit a k-means model to the training data. Since the scatter plots of some of the more relevant Cepstral Coefficients did not demonstrate distinct “clusters” in the data, I was initially unsure how to determine the number of coefficients to use doing data-fitting. I decided to test a k-means model on just one digit by fitting a model with n-clusters then finding the score of that individual model. The graph to the right demonstrates no clear “elbow” per say where the number of coefficients clearly suits the data (does not underfit or overfit), however, I chose to use 5 Cepstral Coefficients which covers the majority of improvement of score from 1-20 Cepstral Coefficients.

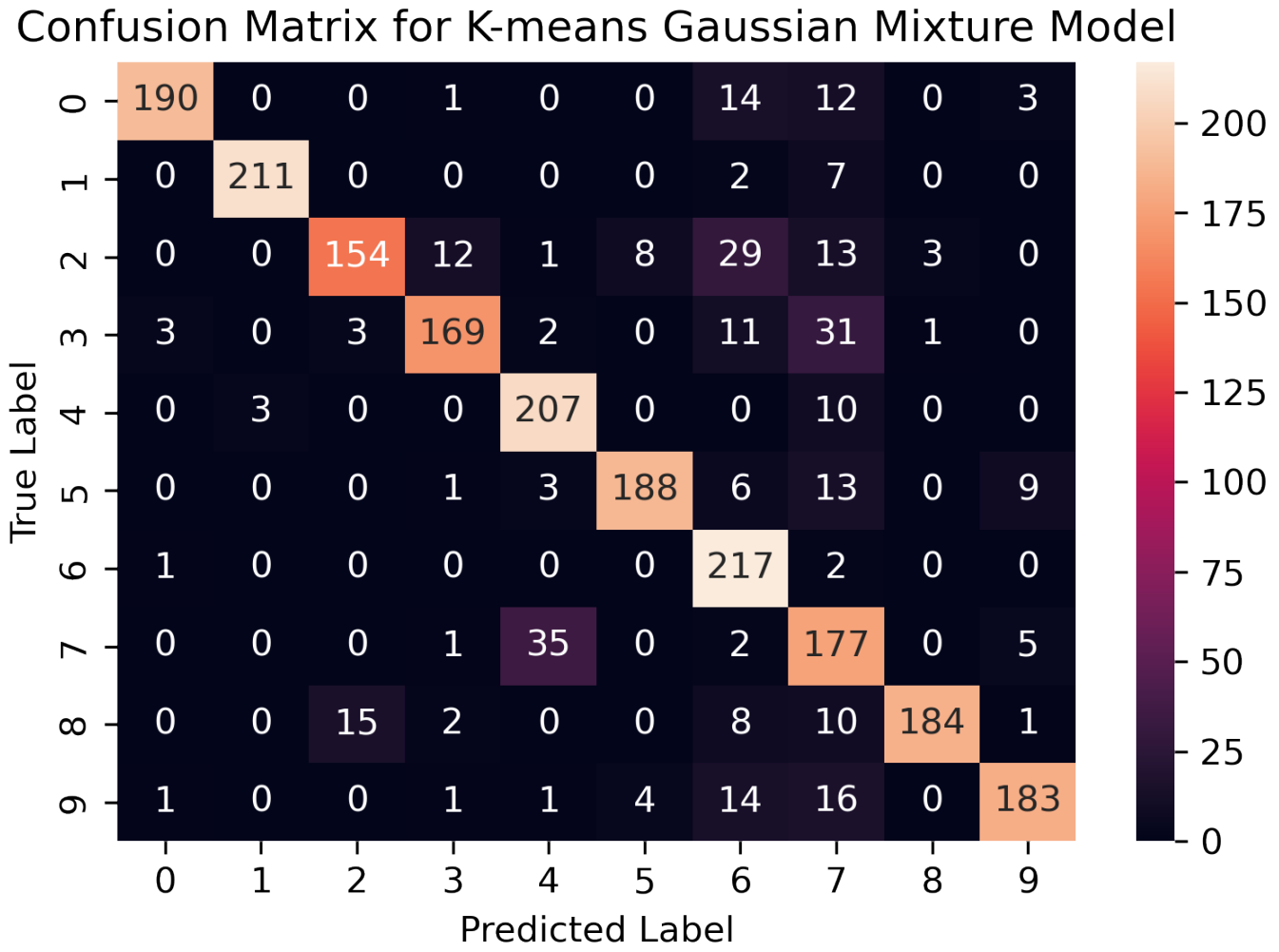


K-Means Gaussian Mixture Model

The model was generated as follows:

1. A k-means model with 5 clusters was fit to the training data (all 13 Cepstral Coefficients included).
2. I retrieved the cluster centers from the trained k-means model.
3. I computed the covariance matrices for the values assigned to each cluster in the model.
4. Assuming a Gaussian distribution for each cluster, I classified new data by determining the digit that produced the highest maximum-likelihood score for a given speech token.

Results of K-Means GMM Model



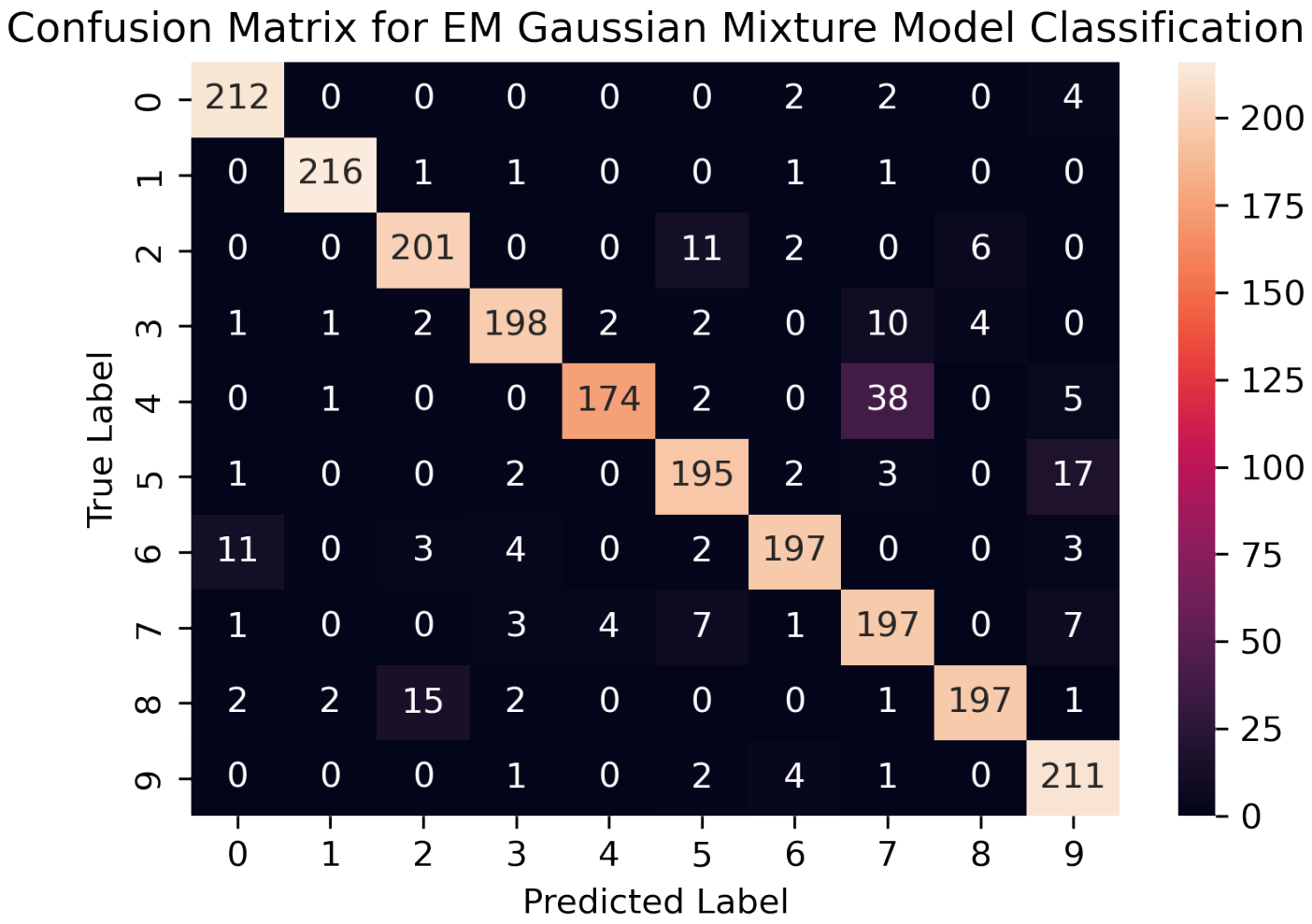
Overall Accuracy on Test Set: 84%

The K-Means GMM model performed relatively well across most digits. The confusion matrix demonstrates a high likelihood of error fort certain digits more than others. In particular, the classifier found the digit “2” to be challenging, achieving an accuracy of 154/220 sample, vs. digit 6, which was nearly perfect at 217/220. For digit 2, the model often confused “2” for “6” or “7”, likely due to the similarity between the Cepstral Coefficients/phonemes of the spoken words and the fact the model does not consider the time-position of the cepstral coefficients.

EM Gaussian Mixture Model

- Next, I implemented another Gaussian Mixture Model, except this time, the cluster centers and covariance matrices were determined using an expectation maximization algorithm.
- In this training scheme, the training algorithm will attempt to maximize the likelihood ratio that the training data derived from a given GMM.
- This training model should help better classify points which may fall on the boundary of multiple clusters, but should most likely belong to one cluster according to the relative likelihood of membership within each cluster distribution.

Results of EM GMM Model



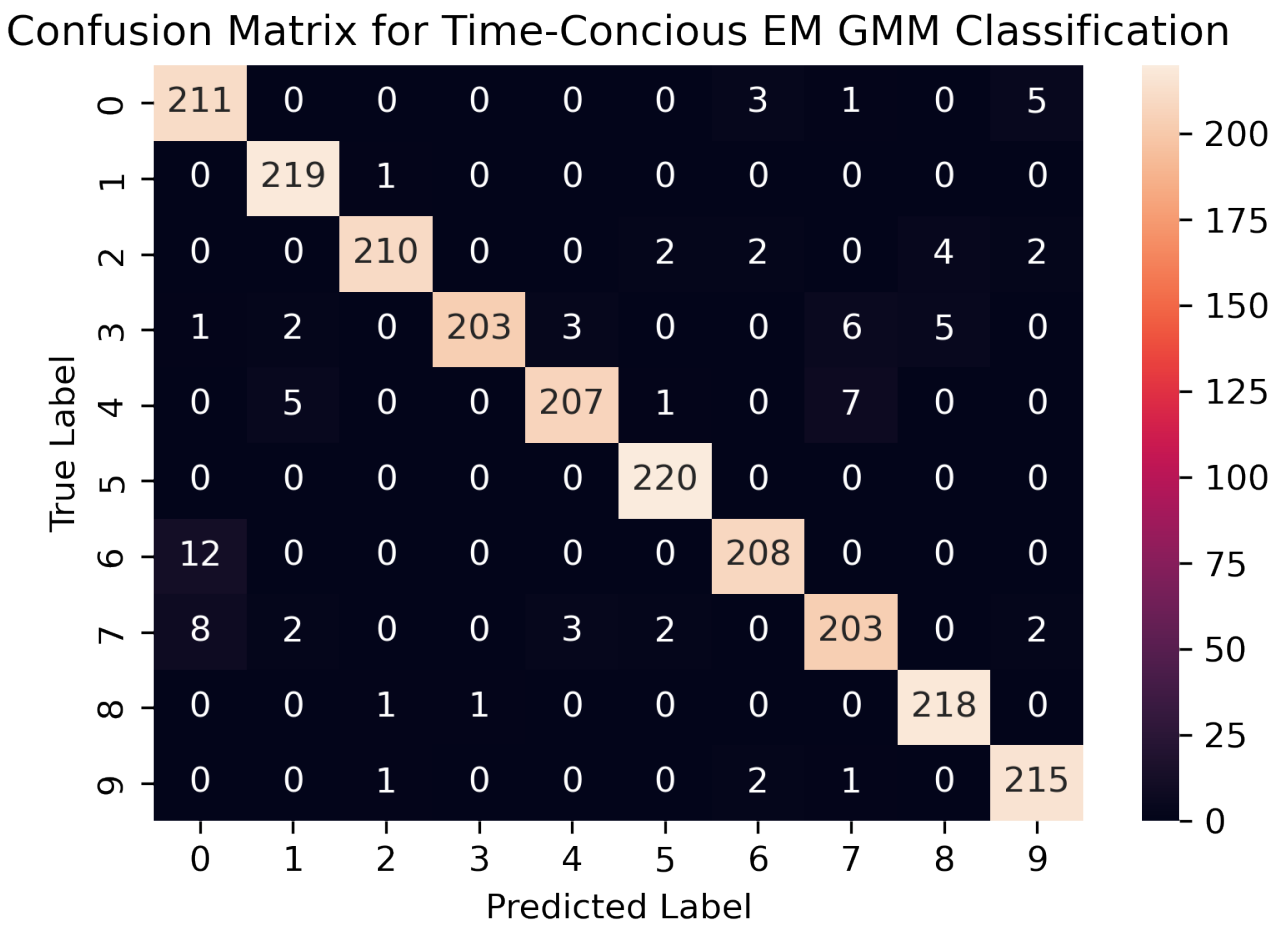
Overall Accuracy on Test Set: 91%

The EM Model Demonstrates a clear improvement over the K-Means Model, although the model still displays a low accuracy for certain digits. For instance, the model commonly confused true “4”’s for the digit 7.

Time Conscious EM GMM Model

- In order to further improve upon the last model, I decided to add a 13^{th} column to the training/test dataframes where the ith row equals i divided by the number of samples in each token. Thus, the value of the 13^{th} column on the first row would be zero, and on the last row it would be 1. This should provide a way to make a new model “time conscious”, while avoiding difficulties caused by time tokens of different sizes.
- I hope this will help eliminate some of the confusion between two numbers that contain similar phonemes, yet in different orders.

Results of Time-Conscious EM GMM Model

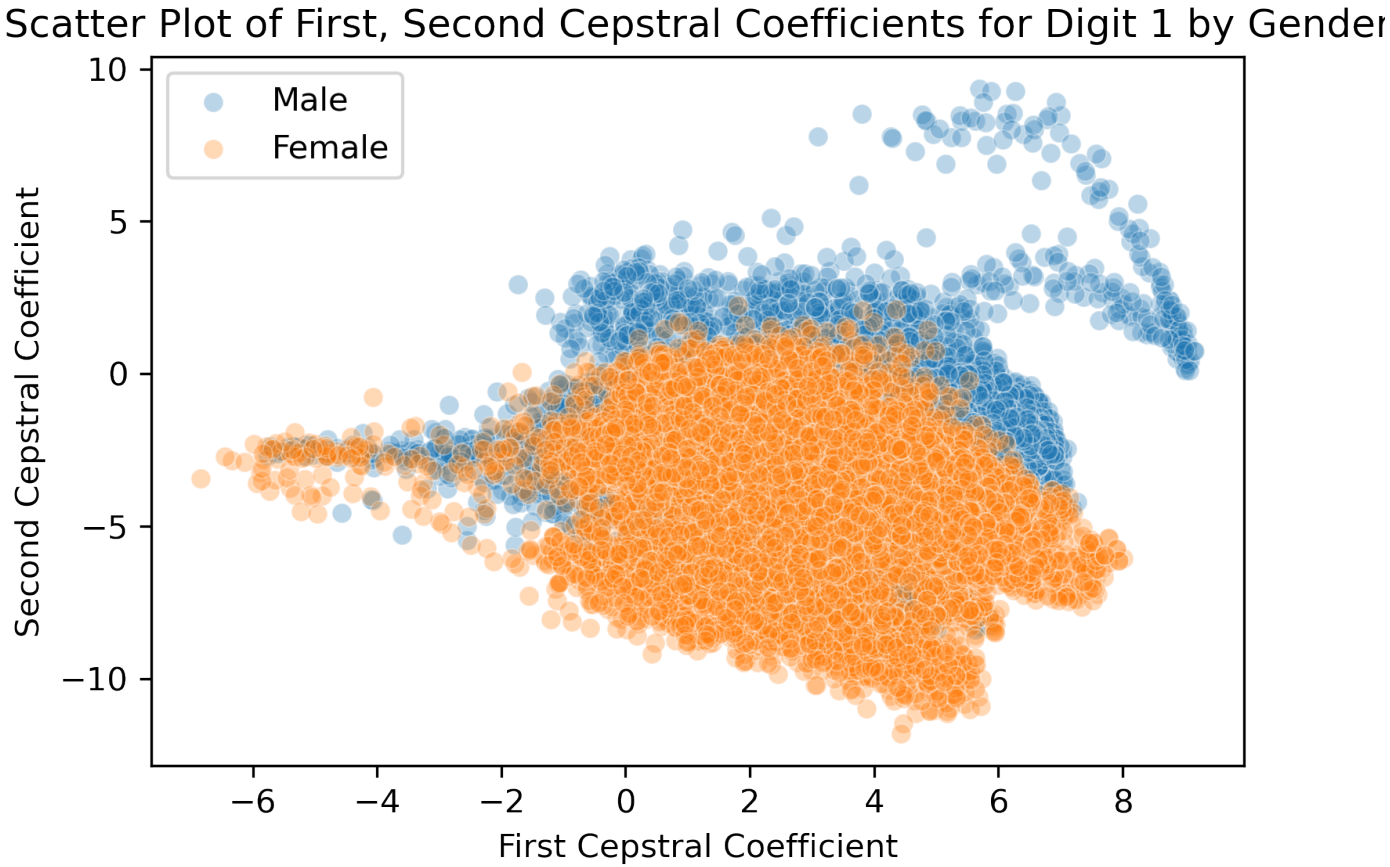


Overall Accuracy on Test Set: 96%

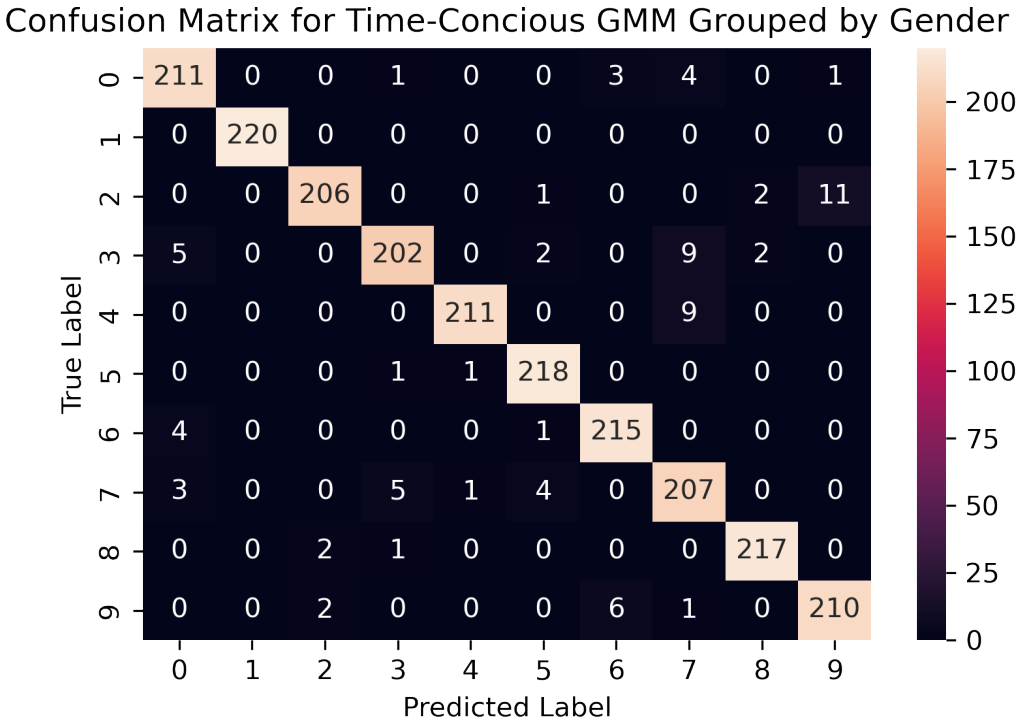
This model performed the best of the three classifiers. By adding another column to make the data “time-conscious”, this model appears to clear up most of the trouble-spots of the last two classifiers, namely (2,7), (3,7), and (4,7), where the first number is the true label. Despite perfect performance all 220 samples of two different digits, this model still displayed a lowest accuracy of 196/220 = 89% for test samples with true label 7. This suggests the model may still be underfitting, and a different approach may be needed.

Considering Gender of Digit Speaker

As mentioned earlier, the data may be separated by the gender of the speaker. Since males and females tend to display different vocal characteristics, it is reasonable to believe each gender displays a different distribution of cepstral coefficients for each cepstral coefficient. The plot to the right verifies this hypothesis, demonstrating a fundamental difference in distributions for digit 1, as an example. I hoped to leverage this difference by generating mixture components for each gender's data separately, then classifying the test samples using only the matching gender's mixture components.

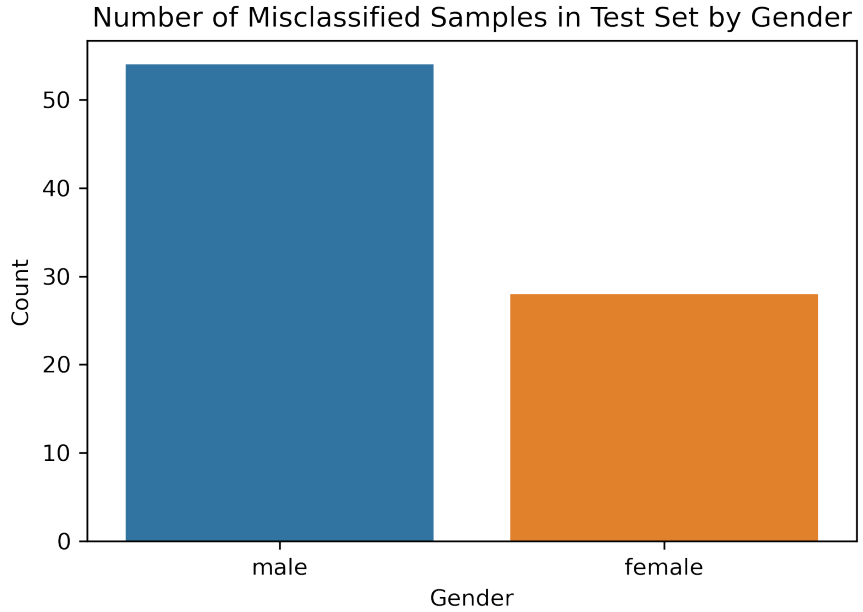


Results of Time-Conscious EM GMM Model



Overall Accuracy on Training Set: 98.1%

Overall Accuracy on Test Set: 97.5%



Accuracy on Male Samples Test Set: 95.1%

Overall Accuracy on Test Set: 96.3%

Training GMM's for spoken digits by each gender separately produced similar results as the previous approach. Although the pattern of Cepstral Coefficients for male and female speakers are visibly different, the consideration of gender does not significantly improve the accuracy of the overall model. However, the model produced slightly different accuracies for each gender, as shown above

Conclusion

The modelling choices I found to be most important throughout this project was the inclusion of more Cepstral Coefficients than I initially thought necessary. At first, I ran the k-means model with just 3 Cepstral Coefficients, which produced poor results despite the apparent “noise-like” nature of the remaining coefficients. In fact, the final test accuracy of my model only improved as I included more Cepstral Coefficients, leading me to make the deiciosn to include all 13 in my final models. I found a less important modelling choice to be constraints on the covariance matrices. In large part, I felt the scatter plots of the spoken digits data were generally convex, whole, and round. I did not see clear “clusters” of phonemes for the samples of any digit, meaning I believe it would be unlikely for a diagonal matrix to “underfit” the data. On the other hand, given the large quantity of data provided, I do not believe a full covariance matrix would cause significant overfitting, as even a full matrix contains just (12 choose 2) = 66 parameters, whereas the number individual data points in the training set is over 30 samples/token * 6600 tokens = 198000 data points.

My systems generally perform well on the test set, however, in their inability to distinguish certain numbers, such as the (4,7) pair. For the future, I would apply a more adaptive, higher variance, model that considers this time-dependent nature of the data, such as a recurrent neural network.