

Spoken Digit Recognition

ECE 480 Final Project

Shaan Gondalia

December 13, 2021

Table of contents

1. Problem Description
2. Data Modeling
3. Maximum Likelihood Classification
4. Results
 - k-Means
 - Expectation Maximization
5. Conclusions
6. Appendix

Problem Description

Spoken utterances can be represented by a time series of cepstral coefficients. These cepstral coefficients are computed by transforming speech to the frequency domain, taking its logarithm, then transforming that into the quefrequency domain. In this case, we are observing the spoken Arabic digits of 0-9, which are represented by frames of 13 cepstral coefficients.

The goal of this project is to explore the use of Gaussian Mixture Models (GMMs) in the field of spoken digit recognition. We will construct these GMMs with k-Means clustering and with the Expectation Maximization (EM) algorithms, and then use the Maximum Likelihood Classification to classify the spoken Arabic digits 0-9.

We will explore the effects of several modeling choices. Namely, the effects of training seperate models based on gender, the effects of covariance shape, the effects of dimensionality reduction, and the effects of cluster count. In the end, we will compare the performance of these modeling choices to determine the parameters for the most accurate speech recognition model.

Motivation

Speech recognition is a pertinent problem in the modern day. Internet Of Things (IoT) devices and accessibility tools use speech recognition to perform tasks, such as turning on lights or browsing the internet. As these smart devices become more popular, it is increasingly important to have efficient and accurate speech recognition models.

GMMs themselves have broad applications that extend outside of the scope of speech recognition. One such field that this type of modeling can be useful for is Computer Vision. GMMs can be used to segment backgrounds and foregrounds, as well as identify objects of interest in images.

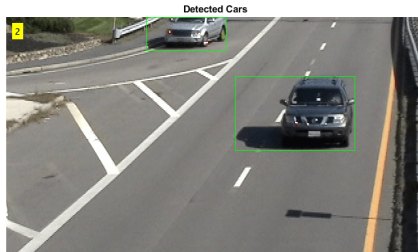


Figure 1: Car Detection using GMMs in MATLAB

Data Modeling

Dataset

The dataset contains 8800 blocks of Mel Frequency Cepstral Coefficients (MFCCs) from speakers uttering the Arabic digits 0-9. Each block is comprised of a time-series of 4-93 frames, representing a single utterance of a digit.

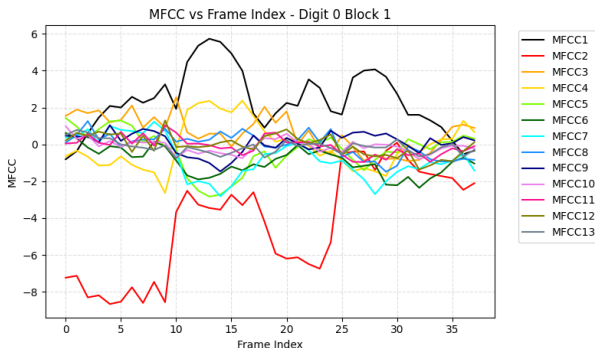


Figure 2: MFCC vs. Frame Index for Digit 0 Block 1

Modeling Choices - k-Means vs. Expectation Maximization

K-Means clustering partitions each frame of MFCCs into M clusters in which each object belongs to the cluster with the nearest mean. The process begins with an initial seed of cluster centers, and iterates until every frame belongs to the same cluster for 2 consecutive iterations. The means and covariances of the clusters can be extracted, which can be used as components for a GMM.

Expectation Maximization is also an iterative technique that maximizes the likelihood of the data given a model. The first step estimates the missing variables and the second step finds parameters that optimize the likelihood of the model.

We will observe the difference in performance in both of these modeling techniques. We will also analyze the run-time of each technique, to see if one model is less computationally heavy than the other. All of the following modeling choices will be repeated in both the k-Means and EM implementations.

Modeling Choices - Number of GMM Components

A GMM follows the formula:

$$f(x) = \sum_{m=1}^M \pi_m \mathcal{N}(\mu_m, \Sigma)$$

Where M is the number of GMM Components (the number of Gaussians included in the model). In this specific case, the intuition is that this number of GMM components should match the number of phonemes for each digit. In practice, this phoneme count is not concrete, as phonemes blend together and get muddled depending on the speaker. Also, this intuition ignores transitions between phonemes, which are not instant and do get picked up in the MFCC time series.

We will explore 3 different choices surrounding the number of GMM components to use: Constant for all digits, # of phonemes, and # of phonemes with transitions. The constant choice serves as a baseline, whereas the others aim to capture the different phonemic counts amongst digits. It is important that these component values are not too high (over-fitting leads to large variance), but it is also important that they are not too low (under-fitting leads to large bias).

The number of phonemes per digit was determined manually by observing the phonetic pronunciations of each digit as well as the MFCC graphs for each digit.

Modeling Choices - Covariance Shape

We can simplify the computational load of the models by assuming that the covariance Σ of each cluster is a certain shape. In this case, we will explore 3 different covariance shapes: Full, Diagonal, and Spherical. These shapes are defined as follows:

$$\Sigma_{FULL} = \begin{bmatrix} \sigma_{1,1}^2 & \sigma_{1,2}^2 & \dots & \sigma_{1,n}^2 \\ \sigma_{2,1}^2 & \sigma_{2,2}^2 & \dots & \sigma_{2,n}^2 \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{n,1}^2 & \sigma_{n,2}^2 & \dots & \sigma_{n,n}^2 \end{bmatrix} \quad \Sigma_{DIAG} = \begin{bmatrix} \sigma_1^2 & 0 & \dots & 0 \\ 0 & \sigma_2^2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \sigma_n^2 \end{bmatrix}$$
$$\Sigma_{SPHERICAL} = \begin{bmatrix} \sigma^2 & 0 & \dots & 0 \\ 0 & \sigma^2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \sigma^2 \end{bmatrix}$$

The covariance shape influences the model flexibility and in turn the bias-variance tradeoff. Full covariance has the largest variance and spherical covariance has the largest bias, with diagonal sitting somewhere in between.

Modeling Choices - Gender

If we create a single GMM for all possible speakers, we assume negligible effects of confounding variables such as gender and age. We can see in the graph below that there may be some general differences in the MFCCs of male and female speakers.

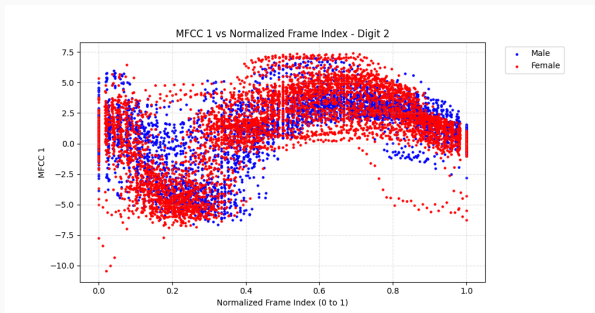


Figure 3: MFCC 1 for 100 Male and 100 Female Utterances of Digit 2

In this case, because the gender of the speakers is known, we can account for this by creating separate GMMs, trained on either the male-identifying speakers or the female-identifying speakers. This may help account for the confounding variable of gender, but adds computational complexity and halves the training data per model.

Modeling Choices - Dimensionality Reduction

We can reduce the computational load and complexity of the model by reducing the dimensionality of the training and testing data. We will do this by omitting the 6-13 MFCCs, leaving only 1-5. We could also do this using Principle Components Analysis (PCA). This comes at the cost of accuracy, as some of the data is lost. This tradeoff might be worthwhile depending on its accuracy as it cuts model execution time.

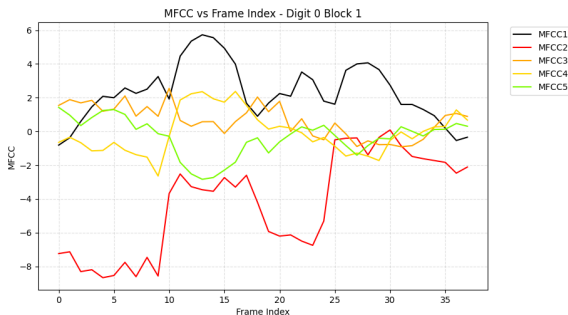


Figure 4: MFCCs 1-5 for Digit 0 Block 1

Maximum Likelihood Classification

The Maximum Likelihood Classification for a block of MFCCs X given the model parameters μ_d, Σ_d, Π_d is:

$$p(X|\mu_d, \Sigma_d, \Pi_d) = \prod_{n=1}^N \sum_{m=1}^M \pi_{m,d} p(x_n|\mu_{m,d}, \Sigma_{m,d})$$

Where N is the number of frames in the block, M is the number of GMM Components, and d is the digit.

This classifier is well-suited for this problem as it gives us a quantitative way to compare the likelihoods of a given block belonging to each of the digits 0-9. By maximizing this likelihood, we obtain a prediction for which digit was spoken. Finding this classifier is simple once the model parameters are found, as it is a simple product of sums of a normal distribution.

At first, there were concerns of underflow in extreme cases when finding the maximum likelihood classification. In general, most classifications hover between 1×10^{-180} and 1×10^{-200} , but every once in a while the value will surpass python's limit of 2.22×10^{-308} and underflow to 0. This happens rarely in blocks that have a large number of frames. Because it happens rarely (and for incorrect classifications) it is not a large concern, but if the data-set were expanded to contain a larger number of frames per block it would become a greater issue.

To deal with this, we could multiply the likelihoods by a constant factor during computation to make underflow even less likely.

Another issue was computation time. Originally, all of the data was stored and manipulated in python lists, making classification incredibly slow. Switching to numpy arrays speeds up the classification significantly, but runs still take 15 minutes due to the time complexity of the algorithm, which is $O(D^2 B N M)$, where D is the number of digits, B is the number of blocks, N is the number of frames per block, and M is the number of components. Each of these iterations used scipy's multivariate normal to generate a likelihood per frame, which was likely another bottleneck.

Results

k-Means Confusion Matrices - Number of Clusters

k-Means with 5 Clusters

Actual \ Predicted	0	1	2	3	4	5	6	7	8	9
0	199	1	0	6	1	0	0	12	0	1
1	8	194	0	9	2	0	0	0	6	1
2	35	0	145	9	0	11	3	3	13	1
3	13	0	2	197	3	0	0	2	3	0
4	13	2	0	0	204	0	0	0	0	1
5	6	0	0	1	3	173	0	20	1	16
6	56	0	3	3	0	0	135	1	0	22
7	12	2	0	2	51	6	0	137	5	5
8	35	0	0	3	0	0	0	0	182	0
9	0	0	1	7	0	9	9	4	0	190

k-Means with Clusters = Phonemes

Actual \ Predicted	0	1	2	3	4	5	6	7	8	9
0	196	1	0	6	1	1	0	14	0	1
1	8	195	0	8	2	0	0	0	5	2
2	42	0	140	5	0	16	2	2	10	3
3	22	0	1	180	2	0	0	4	10	1
4	13	2	0	0	202	1	0	0	0	2
5	4	0	0	1	1	164	0	22	3	25
6	58	0	0	1	0	0	136	1	0	24
7	13	2	0	1	34	5	0	150	8	7
8	44	0	0	0	0	0	0	0	176	0
9	0	0	0	2	0	9	10	3	3	193

k-Means with Clusters = Phonemes + Transitions

Actual \ Predicted	0	1	2	3	4	5	6	7	8	9
0	205	0	0	6	1	0	0	8	0	0
1	8	192	0	4	12	0	0	0	3	1
2	40	0	144	6	0	7	5	5	12	1
3	13	0	0	190	11	0	0	0	6	0
4	13	2	0	0	202	0	0	0	2	1
5	7	0	0	2	1	168	1	18	4	19
6	60	0	5	9	0	0	119	4	0	23
7	14	0	0	3	35	5	0	152	11	0
8	37	0	0	5	0	0	0	0	178	0
9	3	0	1	5	5	7	6	5	5	183

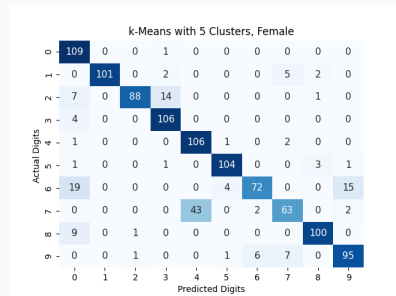
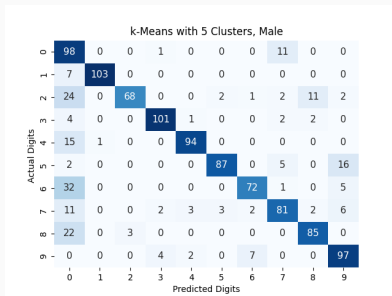
We see that the model with a constant 5 clusters per digit has the best performance, with the models that have clusters = phonemes and clusters = phonemes + transitions having slightly less accuracy.

All 3 of the models commonly mispredict digits as 0, and also share other mispredictions. The confusion matrix heat-maps illustrate these patterns (for instance, misclassification of digit 7 as digit 4, etc.).

In general, the digits 0, 4, and 9 were the most accurate in these 3 models. The models also commonly mispredicted the digits as 0, 4, and 9, which indicates that the training data or model classifier had some bias toward these digits.

Because the 5 cluster model performed better than the other 2 (total accuracy of 0.798 vs 0.787 and 0.788), all of the following models will use 5 clusters.

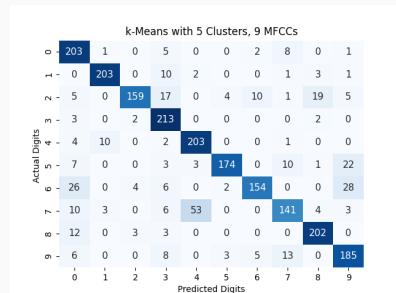
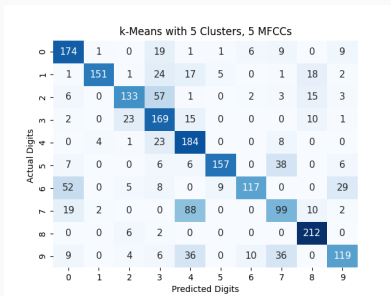
k-Means Confusion Matrices - Gender



Both the Male- and Female-only models had an accuracy larger than the overall model (0.805 and 0.858, respectively). This shows that the latent variable of gender does have an impact on trends in the MFCC time series, and should be accounted for when constructing a system.

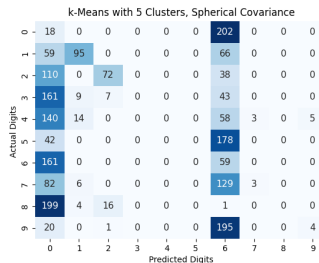
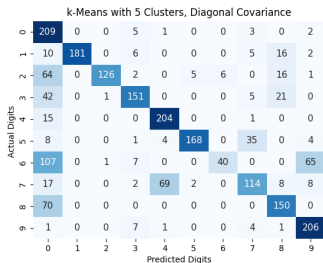
The male model follows similar patterns as the original model, with many incorrect predictions of the digit 0. This behavior is not as strong in the female model, which instead had the misclassification of digit 7 as digit 4 that was noted previously.

k-Means Confusion Matrices - MFCC Count



As expected, accuracy falls off when omitting many MFCCs. The model trained on 5 MFCCs had an accuracy of 0.689, much lower than the baseline. Unexpectedly, the model trained on 9 MFCCs resulted in a greater accuracy than the base model. This may indicate that some of the later MFCCs essentially contribute noise to the system that throws off the classification. If this pattern holds true in general, it is worth considering using only 9 MFCCs as it saves 30% in computation time.

k-Means Confusion Matrices - Covariance Shape



Finally, we observe that the diagonal and spherical covariance models lose a great deal of accuracy. The diagonal covariance model performed slightly better than the 5 MFCC model with an accuracy of 0.704, but still much worse than the baseline. While this implementation did save computation time, it is not worth the accuracy loss.

The spherical covariance model was by far the worst performing model of the 9. It had an accuracy equivalent to that of random guesses. Interestingly, the confusion matrix indicates that this model heavily favored the digits 0 and 6. This may be due to the way the spherical covariance was calculated.

k-Means Accuracies

k-Means									
Digit	5 Clusters	Phomenes	Transitions	Male	Female	5 MFCCs	9 MFCCs	Diag	Spherical
0	0.905	0.891	0.932	0.891	0.991	0.791	0.923	0.95	0.082
1	0.882	0.886	0.873	0.936	0.918	0.686	0.923	0.823	0.432
2	0.659	0.636	0.655	0.618	0.8	0.605	0.723	0.573	0.327
3	0.895	0.818	0.864	0.918	0.964	0.768	0.968	0.686	0
4	0.927	0.918	0.918	0.855	0.964	0.836	0.923	0.927	0
5	0.786	0.745	0.764	0.791	0.945	0.714	0.791	0.764	0
6	0.614	0.618	0.541	0.655	0.655	0.532	0.7	0.182	0.268
7	0.623	0.682	0.691	0.736	0.573	0.45	0.641	0.518	0.014
8	0.827	0.8	0.809	0.773	0.909	0.964	0.918	0.682	0
9	0.864	0.877	0.832	0.882	0.864	0.541	0.841	0.936	0.018
Total:	0.798	0.787	0.788	0.805	0.858	0.689	0.835	0.704	0.114

Figure 5: k-Means Per-Digit and Total Accuracies

In general, most of the k-Means models hovered at an accuracy of 0.8. Most models favored digits 0, likely due to shared phonemes between 0 and the other digits. 0 and 4 were the most accurate digits, reinforcing the idea that the classifier and models had some bias favoring these 2 digits. The least accurate digits were 6 and 7, which were commonly misidentified as 0, 4, or 9.

EM Confusion Matrices - Number of Clusters

EM with 5 Clusters

0	204	0	0	0	0	0	5	10	0	1
1	3	209	0	0	0	1	1	6	0	0
2	1	1	185	0	0	5	14	8	5	1
3	8	2	1	189	0	0	1	18	1	0
4	11	1	0	0	199	0	0	9	0	0
5	0	0	0	0	0	198	6	14	1	1
6	11	0	0	0	0	0	207	1	0	1
7	8	0	0	0	21	9	0	180	0	2
8	27	0	7	0	0	0	0	2	184	0
9	0	0	0	1	0	8	5	9	0	197
	0	1	2	3	4	5	6	7	8	9

Actual Digits

Predicted Digits

EM with Clusters = Phonemes + Transitions

0	197	0	0	1	0	0	8	9	0	5
1	3	208	0	3	0	0	0	5	0	1
2	17	3	143	0	0	16	23	1	5	12
3	33	13	5	115	0	1	4	34	5	10
4	18	1	0	0	82	15	0	97	0	7
5	0	0	0	0	0	197	1	12	0	10
6	7	0	0	0	0	0	206	1	0	6
7	10	0	0	0	7	5	1	178	0	19
8	30	0	2	0	0	0	0	7	172	9
9	2	0	0	0	0	0	7	2	0	209
	0	1	2	3	4	5	6	7	8	9

Actual Digits

Predicted Digits

EM with Clusters = Phonemes

0	201	0	0	2	0	0	9	6	0	2
1	4	205	1	0	2	1	1	2	4	0
2	11	4	146	24	0	8	18	0	7	2
3	25	1	10	109	2	7	7	56	2	1
4	10	1	0	2	146	10	0	46	0	5
5	0	0	0	2	0	198	2	12	0	6
6	9	0	0	0	0	0	203	0	0	8
7	9	0	0	0	7	10	0	183	1	10
8	40	13	6	1	0	0	0	0	155	5
9	0	0	3	0	0	4	3	1	0	209
	0	1	2	3	4	5	6	7	8	9

Actual Digits

Predicted Digits

We see that there is a larger drop-off in performance when changing the number of clusters as compared to the k-Means models. The models that cluster on phonemes and phonemes + transitions tend to have similar misclassifications, such as the cluster of incorrect predictions of the digit 0 and 7. These also exist in the 5-cluster model, but are less severe. This indicates some bias towards choosing classifying the digits 0 and 7, potentially because they share common phonemes with other digits or some other hidden reason.

We can note that the model with clusters = phonemes and the model with clusters = phonemes + transitions performed very similarly, other than a severe drop in accuracy for the digit 4. Such a large drop in accuracy is surprising, but may indicate that the digit 4 has very similar phonemic transitions as another digit (in this case, likely the digit 7).

Because the 5 cluster model performed better than the other 2 (total accuracy of 0.887 vs 0.798 and 0.776), all of the following models will use 5 clusters.

EM Confusion Matrices - Gender

EM with 5 Clusters, Male

Actual \ Predicted	0	1	2	3	4	5	6	7	8	9
0	106	0	0	0	0	0	3	1	0	0
1	2	102	0	0	0	0	0	6	0	0
2	8	12	64	4	0	3	4	4	8	3
3	6	2	0	86	0	0	0	15	1	0
4	9	1	0	0	100	0	0	0	0	0
5	1	0	0	1	0	95	1	5	0	7
6	11	0	0	0	0	1	96	1	0	1
7	9	0	0	1	0	4	0	96	0	0
8	20	0	6	0	0	0	0	9	75	0
9	0	0	1	1	0	0	1	0	0	107

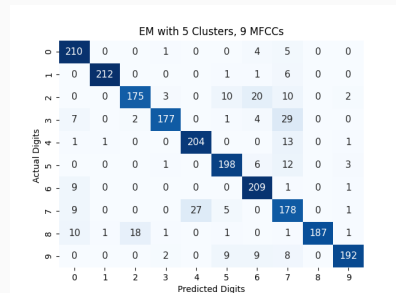
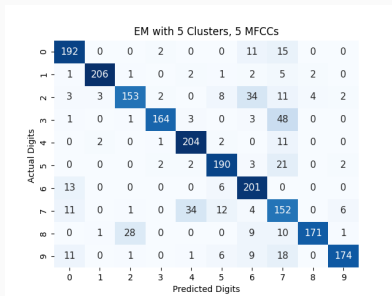
EM with 5 Clusters, Female

Actual \ Predicted	0	1	2	3	4	5	6	7	8	9
0	108	0	0	0	0	0	1	0	0	1
1	0	110	0	0	0	0	0	0	0	0
2	3	0	89	4	0	0	14	0	0	0
3	7	0	0	99	0	0	1	3	0	0
4	0	0	0	0	101	1	0	8	0	0
5	0	0	0	0	0	106	0	1	0	3
6	0	0	0	0	0	0	110	0	0	0
7	0	0	0	0	21	0	0	88	0	1
8	6	0	0	0	0	0	0	0	103	1
9	0	0	0	0	0	0	1	9	0	100

We can see that the model for Male data performs worse than the original 5 cluster model (total accuracy of 0.843 vs 0.887), but the model for Female data performs better (total accuracy of 0.922 vs. 0.887). If we consider the total accuracy between both models, we get an average of 0.8825, which is slightly worse than the original. This does not mean that speaker gender is an unimportant variable, but does indicate that there is a difference in how the classifier and model performs with male vs female data.

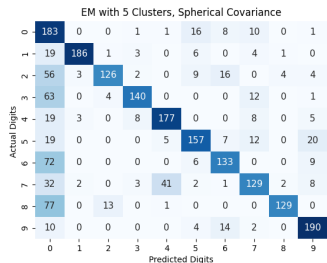
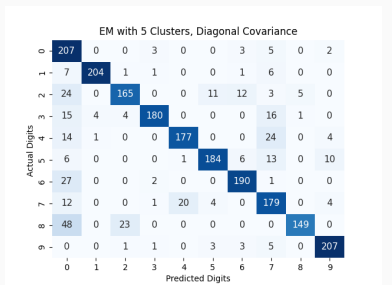
The male model follows similar patterns as the original model, with many incorrect predictions of the digit 0. Surprisingly, this behavior is not present in the female model, which had much fewer misclassifications in general.

EM Confusion Matrices - MFCC Count



The reduced MFCC models exhibit the expected behavior. The model trained on 5 MFCCs had an accuracy of 0.821, and the model trained on 9 MFCCs had an accuracy of 0.883, both being worse than the model trained on all 13. This accuracy drop-off is less severe than k-Means models. A model that only considers 9 MFCCs may be worthwhile as the 0.4% drop in accuracy may be worth the 30% time save, especially when working with larger data sets.

EM Confusion Matrices - Covariance Shape



Both the diagonal and spherical covariance models lose some accuracy. The diagonal covariance model had a decent accuracy of 0.837, which is still worse than the baseline. While this implementation did save computation time, it is not worth the accuracy loss. If time complexity is the limiting factor, it is more worthwhile to consider fewer MFCCs than changing the covariance shape.

The spherical covariance model was again the worst performing model, but not nearly as bad as its k-Means counterpart. It favored the digit 0 heavily. Otherwise, the shape of the confusion matrix matches other models, just with a larger number of incorrect guesses.

Expectation Maximization									
Digit	5 Clusters	Phomenes	Transitions	Male	Female	5 MFCCs	9 MFCCs	Diag Cov	Spherical
0	0.927	0.914	0.895	0.964	0.982	0.873	0.955	0.941	0.832
1	0.95	0.932	0.945	0.927	1	0.936	0.964	0.927	0.845
2	0.841	0.664	0.65	0.582	0.809	0.695	0.795	0.75	0.573
3	0.859	0.495	0.523	0.782	0.9	0.745	0.805	0.818	0.636
4	0.905	0.664	0.373	0.909	0.918	0.927	0.927	0.805	0.805
5	0.9	0.9	0.895	0.864	0.964	0.864	0.9	0.836	0.714
6	0.941	0.923	0.936	0.873	1	0.914	0.95	0.864	0.605
7	0.818	0.832	0.809	0.873	0.8	0.691	0.809	0.814	0.586
8	0.836	0.705	0.782	0.682	0.936	0.777	0.85	0.677	0.586
9	0.895	0.95	0.95	0.973	0.909	0.791	0.873	0.941	0.864
Total:	0.887	0.798	0.776	0.843	0.922	0.821	0.883	0.837	0.705

Figure 6: EM Per-Digit and Total Accuracies

In general, the EM models performed better than the k-Means models, with an average accuracy of 0.83. The simple 5 cluster model proved to be the most accurate. The most accurate digits were 0, 1, and 9. The least accurate digits were 2 and 3.

Modeling Choice	Run Time (s)	
	k-Means	EM
5 Clusters	1077.4	1098.0
Phonemes	870.0	870.8
Phonemes + Transitions	1946.7	1978.2
Male	551.8	559.9
Female	527.0	535.8
5 MFCCs	334.9	338.8
9 MFCCs	667.2	675.8
Diag Covariance	336.3	352.2
Spherical Covariance	335.2	353.3

Figure 7: Total Run Time for All Models (Training and Testing)

The run time for the k-Means and

Expectation Maximization classification models are very similar, with the EM models taking around 30 seconds longer. This is a result of differences in the training time of the models, as the classification step is the same in both the EM and k-Means implementations. We see how increasing the number of clusters, number of testing blocks, number of MFCCs, and covariance fidelity increase the run-time of the models (and vice-versa). We can see that the run time follows a complexity of $O(D^2BNM)$.

Conclusions

Conclusions - Important Modeling Choices

All of the explored modeling choices had some impact on spoken digit classification performance. The most important of these are the covariance shape and number of MFCCs chosen. These modeling choices have implications on both the running time and the accuracy of the model, so finding a trade-off between the two is an important decision.

The covariance shape is very important as it defines the relationships between MFCCs. In this case, the information that is lost by switching to a diagonal or spherical covariance has a large negative impact on the performance of the model (one that is not worth the computational benefits).

The number of MFCCs presents another interesting trade-off, as it scales down the run-time linearly. In both models that used 9 MFCCs, performance was equal or better than an equivalent model trained on 13 MFCCs, making the trade-off worthwhile. This concept relates directly to the idea of the bias-variance tradeoff, with a 13 MFCC model having less flexibility and higher variance and a 1 MFCC model having the most flexibility and highest bias. Striking a balance between the two is important (in this case, the balance was 9 MFCCs).

Conclusions - Less Important Modeling Choices

Gender and number of clusters were less important. While changing the number of clusters did have an effect on the performance, there was never a number of clusters that produced a better result than a constant 5 per digit. This may indicate some flaw in the Maximum Likelihood Classification (MLC) that adds bias to the system when the number of clusters per digit is not constant. If we observe the MLC formula:

$$p(X|\mu_d, \Sigma_d, \Pi_d) = \prod_{n=1}^N \sum_{m=1}^M \pi_{m,d} p(x_n | \mu_{m,d}, \Sigma_{m,d})$$

We notice that the inner sum will be larger for digits with a larger number of clusters. This may only manifest in extreme cases (in this case the number of phonemes varied by at most 2), but nonetheless is something to consider.

While the results show that separating the models in terms of gender has some effect on total accuracy, it is important to recognize that these models were solely tested on their respective gender's testing data (the female model only used female testing data, as did the male model). If the models are run against all of the testing data instead, their performance would be much worse than the baseline. In a real-world scenario when the gender of the speaker is potentially unknown, the base model that does not consider gender might have a better performance.

After viewing the results, the single system that performs with the best balance between accuracy and runtime is the Expectation Maximization Model with 5 clusters per digit and 9 MFCCs. If the gender of the speakers on the testing data is known, then we would separate into one male model and one female model, but if not then it would be a single model that handles both Male and Female Data. The classifier will still be the Maximum Likelihood Classification that was described previously.

This system provides the second best total accuracy of 0.883, only 0.004 less than its 13 MFCC counterpart, while saving 30% in run time.

To improve the system, we should further explore the use of a non-constant number of clusters per digit. Although all of the tests that were run did not find parameters that were better than 5 clusters per digit, a better combination may exist. This may entail modifying the classifier to eliminate the issues of underflow and bias toward digits with larger numbers of clusters (if this determined to be an issue). Also, if running time is still a limiting factor the classifier should be modified (either its formula or its implementation in code). A dataset with more fine blocks (100s of frames per block instead of <100) may also help accuracy, although it would increase runtime and potential underflow concerns.







Conclusions - Lessons Learned

A better way to aggregate results could be a future improvement. Currently, results are printed to the terminal but have to be stored in files manually, which makes searching for results a pain. Next time I would focus on having sanity checks that make sure all of the code is functional before starting to train the model and classify digits. Waiting 15 minutes for results only to realize a parameter was set incorrectly was a large time-sink.

Perhaps one way to save time while testing model parameters would be to only test on a subset of the testing data. Instead of finding the likelihoods for all 2200 blocks, we could find the likelihood for 1000 blocks to determine if the modeling choice is worthwhile, then run it on all 2200 blocks if the results are promising. This would have let me test many more modeling choices. I would also implement some rudimentary form of threading in the future, as training and testing several models at once would cut down execution time drastically.

Making sure the data was formatted in a way that made sense early was very important and saved many headaches that would have come further down the line. Separating the data into 3-d numpy arrays made it very easy to work with. Making the early switch from python lists/dictionaries to numpy arrays also sped up the entire workflow. Also, designing the software to be modular in the first place simplified the process of adding modeling parameters in the end.

Appendix

-  A. V. Oppenheim and R. W. Schafer, "Dsp history - from frequency to quefrequency: a history of the cepstrum," *IEEE Signal Processing Magazine*, vol. 21, no. 5, p. 95–106, 2004.
-  N. Hammami and M. Sellam, "Tree distribution classifier for automatic spoken arabic digit recognition," *2009 International Conference for Internet Technology and Secured Transactions, (ICITST)*, 2009.
-  J. MacQueen, "Some methods for classification and analysis of multivariate observations," *Fifth Berkeley Symposium*, 1967. [Online]. Available: <http://mlsp.cs.cmu.edu/courses/fall2010/class14/macqueen.pdf>
-  A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *Journal of the Royal Statistical Society. Series B*, 1977. [Online]. Available: <https://www.jstor.org/stable/2984875>
-  "Spoken arabic digit data set." [Online]. Available: <https://archive.ics.uci.edu/ml/datasets/SpokenArabicDigit>
-  S. Gondalia, "Spoken digit recognition source code." [Online]. Available: <https://github.com/ShaanGondalia/DigitRecognition>



“Scikit api reference.” [Online]. Available:
<https://scikit-learn.org/stable/modules/classes.html>



“Numpy api reference.” [Online]. Available: <https://numpy.org/doc/>



“Matplotlib 3.5.0 documentation.” [Online]. Available:
<https://matplotlib.org/stable/index.html>



“Scipy documentation.” [Online]. Available:
<https://docs.scipy.org/doc/scipy/reference/>



“Seaborn statistical data visualization.” [Online]. Available:
<https://seaborn.pydata.org/>



“Detect cars using gaussian mixture models.” [Online]. Available:
<https://www.mathworks.com/help/vision/ug/detecting-cars-using-gaussian-mixture-models.html>

Eric Qi, Tyler Feldman, and I bounced ideas off of each other while completing this project. We didn't explicitly share code, but did share the packages and libraries that we used (Sklearn's Guassian Mixture Model and k-Means).