# Spoken Arabic Digit Classification

---

**Arnav Nayak**
ECE 480: Applied Probability for Statistical Learning

**Dr. Stacy Tantum**

# Overview

# Project Description

# Background

## Speech Recognition - Why it Matters?

The ability for a computer to recognize human speech has fundamentally altered how people interact with technology. In an increasingly digital world, accurately converting speech into text or commands is critical to breaking down barriers in **human computer interaction** and enabling seamless **user experiences** that empower people. The applications of speech recognition are vast, ranging from language translation tools to AI-powered virtual assistants[1] that power our devices and homes. Most importantly, speech recognition enables people to build software that has the potential to **dismantle communication barriers** and **democratize access to technology**.

[1]S. P. Yadav, A. Gupta, C. Dos Santos Nascimento, V. Hugo C. de Albuquerque, M. S. Naruka and S. Singh Chauhan, "Voice-Based Virtual-Controlled Intelligent Personal Assistants," 2023 International Conference on Computational Intelligence, Communication Technology and Networking (CICTN), Ghaziabad, India, 2023, pp. 563-568, doi: 10.1109/CICTN57981.2023.10141447.

# Background

## Mel-Frequency Cepstral Coefficients (MFCC)

The success of any system or model can be hurt with increased complexity. When developing models, especially predictive models based on large datasets, it is important we maintain, or even reduce, the dimensionality of our data. Doing so reduces storage space, increases computation speed, removes redundancies and prevents overfitting to noise. We can achieve **dimensionality reduction** through methods like **feature extraction**.

In order to tackle speech recognition, we need an efficient and simple representation of our input data. Using the raw audio signal is too complex and contains noise we want to remove. Instead, we can convert the audio signal to a set of coefficients that represent the parts of our signal that matter most for human perception. These are the **Mel-Frequency Cepstrum Coefficients**.[2] They are calculated by finding the frequencies at which the spectrum (found through the Discrete Fourier Transform) of the audio signal peaks. But for the purposes of this project, they are a set of numbers that best **characterize an audio signal**, and will be used as our **input features** for our models.

[2]Z. K. Abdul and A. K. Al-Talabani, "Mel Frequency Cepstral Coefficient and its Applications: A Review," in IEEE Access, vol. 10, pp. 122136-122158, 2022, doi: 10.1109/ACCESS.2022.3223444.

# Background

## Spoken Arabic Digit Dataset - UC Irvine ML Repository[3]

The dataset we're using in this project has **8,800** samples, each representing a single utterance of a digit '0' through '9' in Arabic. Each sample is organized as a time series of 35-40 analysis frames, where each frame is a set of **13 MFCC values** that represent the spectrum of the audio signal at that instant. For each digit, there are 88 speakers (44 males and 44 females) that repeat the digit 10 times (880 utterances/digit x 10 digits = 8,800 samples). The sampling rate of each sample is 11025 Hz.

The dataset is split up into a **training split** with 6,600 samples and a **testing split** with 2,200 samples. Our model parameters will be trained on the training data and its accuracy will be measured with the testing data.

[3]Bedda,Mouldi and Hammami,Nacereddine. (2010). Spoken Arabic Digit. UCI Machine Learning Repository. https://doi.org/10.24432/C52C9Q.

# Project Goals

## Spoken Arabic Digit Classification - Objectives

The goal of this project is to delve into speech recognition by focusing on the identification of spoken digits in Arabic. Specifically, we want to

- Model the **distribution** of Cepstral coefficients using **probabilistic mixture models**
- Determine the **parameters** of each model through two different **unsupervised clustering** methods and analyze the impact of both on classification accuracy
- Determine the impact of **constraining** the number of MFCC values we use in our models
- Determine the impact of restricting characterization choices for the distribution by constraining our **covariance** estimates for the mixture models
- Determine the effect of incorporating the gender of the speaker in our models - a **latent variable**

# Project Goals

## Spoken Arabic Digit Classification - How are we achieving this?

The pronunciation of each digit can be broken down into **distinct phonemes** - the smallest unit of speech distinguishing one word element from another. Over time, the MFCC values for a single digit tend to be grouped together for a particular slice of time. For example, the graph on the right shows the 13 MFFC values for a single digit (7) sample over time (frame number). The MFCC values remain relatively the same from frame number 3 to 11. They transition to another set of relatively constant values from frame 11 to 15. The same is true for frame 15 to 27. These groupings represent the distinct phonemes, or transition between phonemes, within the digit.

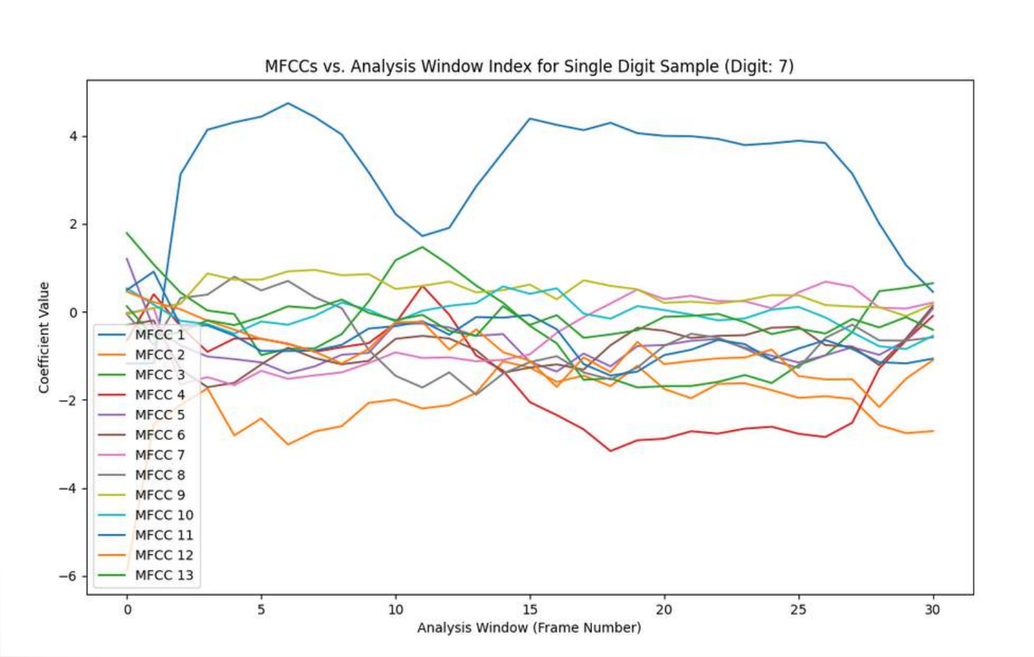


*Figure 1: MFCC Values vs. Analysis Window for Sample Digit 7*

# Project Goals

## Spoken Arabic Digit Classification - How are we achieving this?

Since each digit contains a unique set of phonemes, we expect that each digit can be represented by a unique set of **MFCC clusters**, where each cluster is a prominent **phoneme** within the digit. For each digit, we can create a model that captures these prominent clusters. In the end, we will have 10 different models. Our classification process will be based on **maximum likelihood**. For a new data point (time series of 13 MFCC values) we can calculate the likelihood that the data point belongs to each model. The model with the highest likelihood will be our prediction.

We will be using Python to create our models. We will be using *scikit-learn*[4] to assist our implementations of the different algorithms we implement (specifically K-Means and Expectation Maximization). We will also make use of the tools provided to us by *numpy*[5] and *matplotlib*[6] libraries for computation and visualization.

[4]Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.
[5]Harris, C.R., Millman, K.J., van der Walt, S.J. et al. Array programming with NumPy. Nature 585, 357–362 (2020). DOI: 10.1038/s41586-020-2649-2.
[6]J. D. Hunter, "Matplotlib: A 2D Graphics Environment", Computing in Science & Engineering, vol. 9, no. 3, pp. 90-95, 2007.

# Project Goals

## Note - Other Applications

The modeling framework outlined can be used for a vast number of applications other than speech classification. One specific scenario is **radar signal classification** in defense systems.[7] In modern defense systems, radar plays a crucial role in monitoring airspace or maritime environments. If incoming radar signals could be classified by their source (e.g. friendly aircraft, enemy ship, etc.), we can enable better **threat detection** and improve **operational efficiency**.

We can use a similar modeling approach to tackle this challenge. We can extract key **features** of the signal, such as frequency, pitch, phase etc, and train a **mixture model** to learn the probabilistic distribution of features for each class of signal. When a new radar signal is detected, we can use the model to assign probabilities to each class and classify the signal based on the **maximum likelihood.**

[7]Xuhua Gong, Huadong Meng and Xiqin Wang, "A GMM-based Algorithm for Classification of Radar emitters," 2008 9th International Conference on Signal Processing, Beijing, China, 2008, pp. 2434-2437, doi: 10.1109/ICOSP.2008.4697641.

# Data Modeling

# Gaussian Mixture Models (GMM)

## Model Assumptions

We will use a **Gaussian Mixture Model**[8] (GMM) to characterize the distribution of our dataset. A GMM is a weighted linear combination of Gaussian distributions, where the sum of the weights is one. The mathematical representation of a GMM is shown below, where $p(x)$ represents a GMM with K **mixture components**. Each Gaussian component has its own weight, mean and covariance matrix. These are the **parameters** we are finding when we train our model.

$$p(\vec{x}) = \sum_{i=1}^{K} \phi_i \mathcal{N}(\vec{x} \mid \vec{\mu}_i, \Sigma_i)$$

$$\mathcal{N}(\vec{x} \mid \vec{\mu}_i, \Sigma_i) = \frac{1}{\sqrt{(2\pi)^K |\Sigma_i|}} \exp\left(-\frac{1}{2}(\vec{x} - \vec{\mu}_i)^{\mathrm{T}} {\Sigma_i}^{-1} (\vec{x} - \vec{\mu}_i)\right)$$

$$\sum_{i=1}^{K} \phi_i = 1$$

*Figure 2: Gaussian Mixture Model*[9]

It makes sense to model the distribution of our data as a Gaussian Mixture Model because of the **natural clustering** that occurs with phonemes. Within a GMM for a digit, each Gaussian component can represent a unique phoneme. Thus, we can make the assumption that the number of mixture components will be equal to the number of phonemes in that digit, as an **entire GMM will represent a unique set of phonemes**.



[8]Miin-Shen Yang, Chien-Yo Lai, Chih-Ying Lin, A robust EM clustering algorithm for Gaussian mixture models,Pattern Recognition, Volume 45, Issue 11, 2012, Pages 3950-3961, ISSN 0031-3203, https://doi.org/10.1016/j.patcog.2012.04.031.
[9]Gaussian Mixture Model. Brilliant.org. Retrieved 23:25, December 10, 2023, from https://brilliant.org/wiki/gaussian-mixture-model/

# Parameter Learning

## K-Means Clustering[10] - What is it?

In this project, we explore two different methods to estimate the parameters of a GMM. One method is **K-Means clustering**. In K-Means clustering, our goal is to partition our data into **K distinct clusters**. The steps for the K-Means algorithm is as follows:

1) **Initialization**: Choose K initial centroids
2) **Assignment**: Assign each data point a cluster based on the nearest centroid using Euclidean distance
3) **Update**: Recalculate the centroids as the mean of all of the data points assigned to that centroid's cluster
4) **Iterate**: Repeat the assignment and update steps until convergence - data no longer changes cluster assignment

Mathematically, we are attempting to minimize the following **objective function**. We loop through all *m* data points and minimize the distance between it and the mean of its assigned cluster. The binary indicator $w_{ik}$ is either 1 or 0 depending on if the data point is in the cluster *k*.

$$J = \sum_{i=1}^{m} \sum_{k=1}^{K} w_{ik} \|x^i - \mu_k\|^2$$

*Figure 3: K-Means Objective Function[11]*



[10]Lloyd, Stuart P. "Least squares quantization in PCM." Information Theory, IEEE Transactions on 28.2 (1982): 129-137.
[11]Dabbura, Imad. "K-Means Clustering: Algorithm, Applications, Evaluation Methods, and Drawbacks." Medium, Towards Data Science, 27 Sept. 2022, towardsdatascience.com/k-means-clustering-algorithm-applications-evaluation-methods-and-drawbacks-aa03e644b48a.

# Parameter Learning

## K-Means Clustering - Clustering MFCCs

In order to use K-Means clustering to estimate the parameters of our GMM, we can run K-Means to find K clusters within our dataset. For each cluster, we can find the **weight** of the mixture component representing that cluster by finding the proportion of observations in that cluster. We can also calculate the **mean** and **covariance** of the cluster. The **GMM** for each digit will be the weighted combination of the estimated Gaussian models.

The results of estimating GMM parameters with K-Means clustering are shown on the right. Since there are 13 MFCC values, we will be running the clustering in **13-dimensional space**. However, we can visualize the clustering taking place by plotting only the first two MFCCs against each other. In the figure on top, we find four clusters for the digit 1, as its pronunciation contains four phonemes. The **GMM parameters** are visualized as contour plots representing the mean and covariance of the distribution.

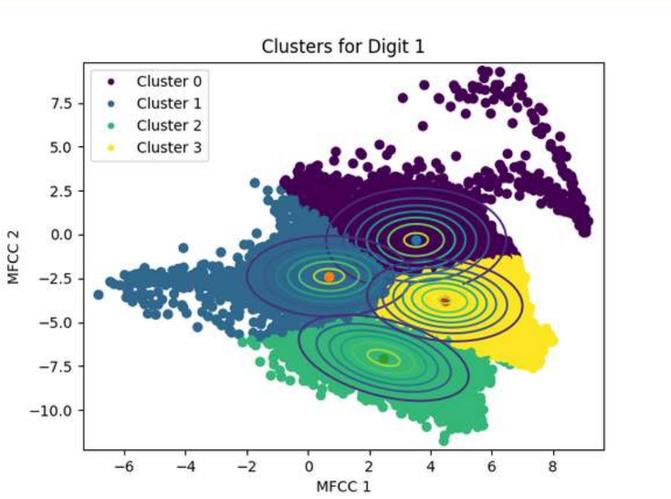


*Figure 4: Scatter Plot for MFCC 1 and MFCC 2 with Clusters (Digit 1)*

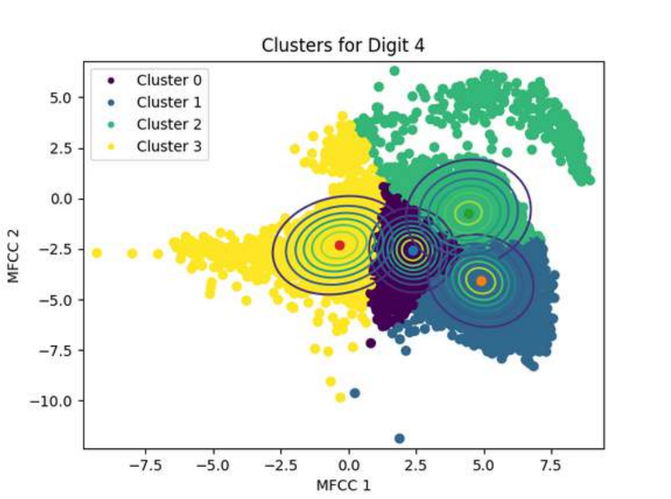


*Figure 5: Scatter Plot for MFCC 1 and MFCC 2 with Clusters (Digit 4)*

# Parameter Learning

## Expectation Maximization[12] (EM) - What is it?

The second way we can estimate the parameters of a GMM is through an iterative algorithm known as **Expectation Maximization.** EM will assign each data point a set of K **responsibilities** that represents how much each cluster is responsible for that specific data point, reflecting the **probability** of the data point belonging to each of the K clusters in the mixture model. The steps of EM is as follows:

1) **Initialization**: Start with initial estimates of weights, means and covariances
2) **Expectation**: Calculate the responsibility that each Gaussian component takes for each data point
3) **Maximization**: Update the means and covariances of each component based on the responsibilities; update mixture weights
4) **Iterate**: Repeat the expectation and maximization steps until convergence - changes in estimate fall below a threshold

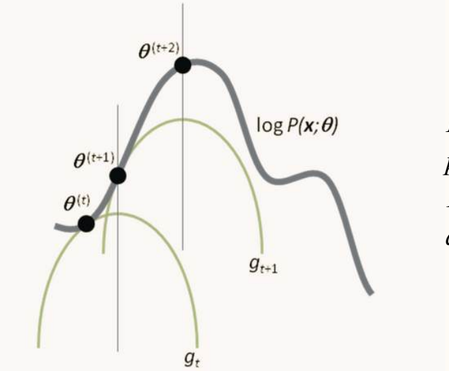


*Figure 6: Expectation Maximization Lower Bounded by Log-Likelihood[13]*

*Expectation aims to construct a function g(t) from initial parameters θ that is lower bounded by the Log-Likelihood. Maximization will find θ(t+1) where this function is maximized and the next function g(t+1) will be constructed from that point.*



[12]Miin-Shen Yang, Chien-Yo Lai, Chih-Ying Lin, A robust EM clustering algorithm for Gaussian mixture models,Pattern Recognition, Volume 45, Issue 11, 2012, Pages 3950-3961, ISSN 0031-3203, https://doi.org/10.1016/j.patcog.2012.04.031.
[13]Expectation Maximization (EM) Algorithm. Expectation Maximization (EM) Algorithm - Computational Statistics in Python 0.1 Documentation, people.duke.edu/~ccc14/sta-663/EMAlgorithm.html. Accessed 10 Dec. 2023.

# Parameter Learning

## Expectation Maximization - Clustering MFCCs

Visualizing the clusters generated from expectation maximization is more complex than K-Means as each data point is not assigned one cluster, but rather shared responsibility between all clusters. Because of this **soft clustering assignment**, data points will look like gradients as they transition from one cluster to another.

The results of estimating GMM parameters with EM is shown on the right. Again, we are fitting our models and plotting only **2 MFCC values** instead of 13 for visualization purposes, but our models will be trained using all 13 MFCC values. In the figure on the top, we find 4 clusters for the digit 1. The **covariance contour maps** have significantly more overlap when compared to K-Means. Data points that seemingly fall in between two clusters share color (e.g. some points are yellow-green). This specific example gives us a **GMM** with 4 mixture components for digit 1, representing 4 unique phonemes within its pronunciation.

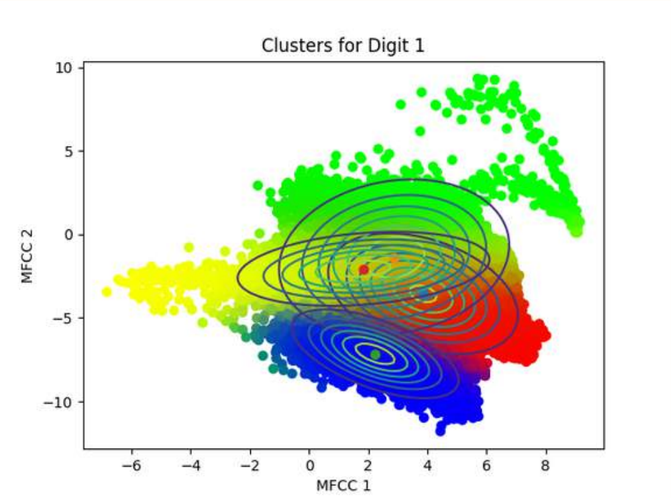


*Figure 7: Scatter Plot for MFCC 1 and MFCC 2 with Clusters (Digit 1)*

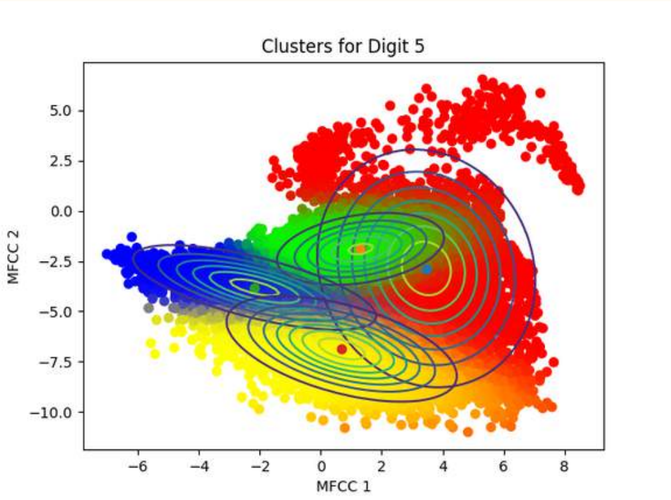


*Figure 8: Scatter Plot for MFCC 1 and MFCC 2 with Clusters (Digit 5)*

# Parameter Learning

## K-Means

**Advantages**

- K-Means guarantees **convergence**
- K-Means is **faster** and **easier** to compute and **scales** better as a result

**Disadvantages**

- K-Means is highly sensitive to **initialization**
- K-Means provides **hard clustering**, assigning one cluster to each data point
- K-Means **does not directly optimize** a GMM, but its clusters can be used as estimates for GMM parameters

## Expectation Maximization

**Advantages**

- EM provides **soft clustering**, assigning responsibilities for each cluster to each data point
- Resulting EM model is expected to be **more accurate**
- EM **directly optimizes** GMM parameters

**Disadvantages**

- EM is **slower** and requires more **computing power**, resulting in a **longer** training process
- EM also somewhat sensitive to **initialization**

# Other Modeling Choices

## Number of MFCCs to Include

So far, we've been choosing to include all 13 MFCC values as input features for our model. This is a **design choice** that has its own pros and cons. On one hand, including all 13 MFCC values will allow our model to incorporate all of the information available to us, making a **more accurate model.** However, using all 13 values means our input features are in 13-dimensions. **Higher dimensionality** means we need **more data points** to allow our model to capture relationships between those features. It also means more **computational resources** are necessary to process the input. Choosing to **restrict** the number of MFCC values we use in our input can be a possible optimization.

The bar chart on the right shows the **variance** of each MFCC in the dataset. The first 5 MFCCs seem to have the most variance. This may indicate that these coefficients are significantly more important when determining the phonemes than others. We can examine the effect of constraining MFCC values on classification accuracy.

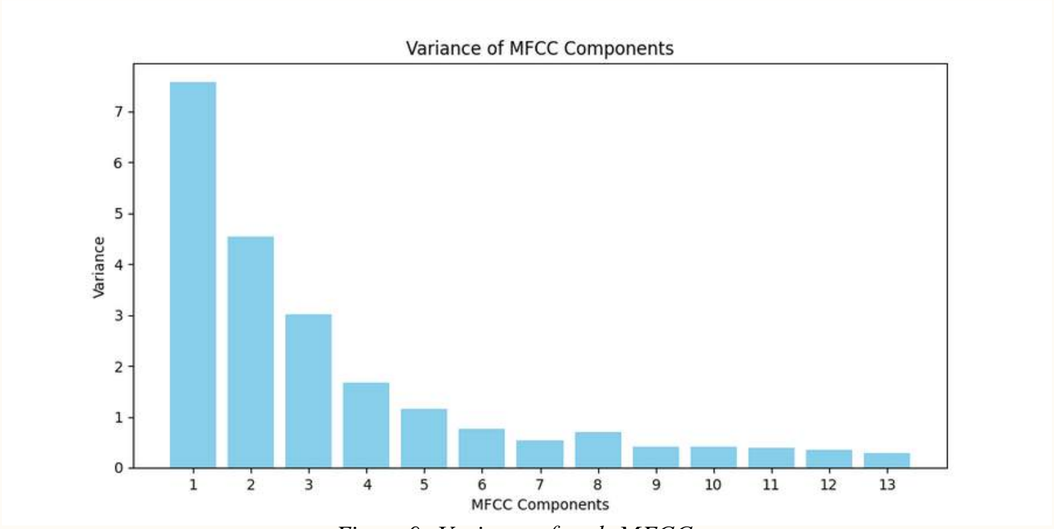


*Figure 9: Variance of each MFCC*

# Other Modeling Choices

## Constraining GMM Covariance Matrices Estimates

Another design choice we have is how we **characterize our covariance matrix** for each Gaussian component. Using a **full** covariance matrix will capture all of the relationships in the data, but may run the risk of **overfitting**. We can restrict the number of parameters in our covariance matrix as a form of **regularization** to reduce model complexity and prevent overfitting. We can instead use a **diagonal** or **spherical** covariance matrix to characterize our Gaussian components. The contour plot of each restriction is shown below. We can see that the more we constrain our covariance matrix, the more restrictive our covariance shape becomes.

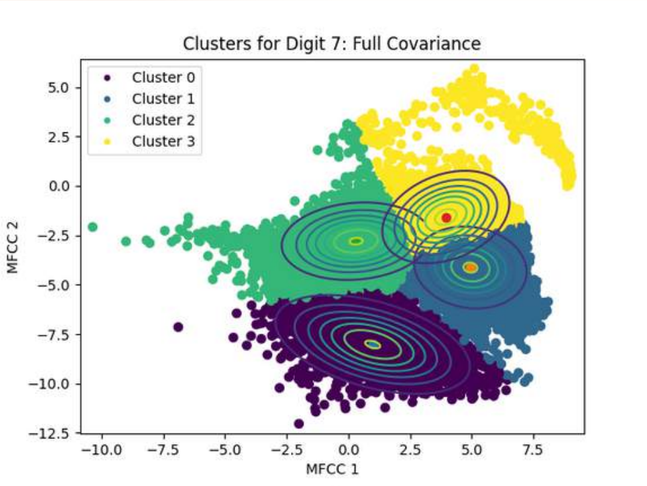


*Figure 10: Full Covariance*

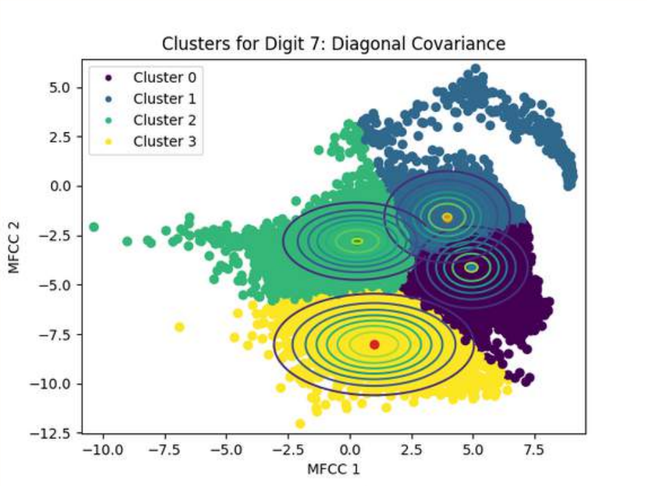


*Figure 11: Diagonal Covariance*

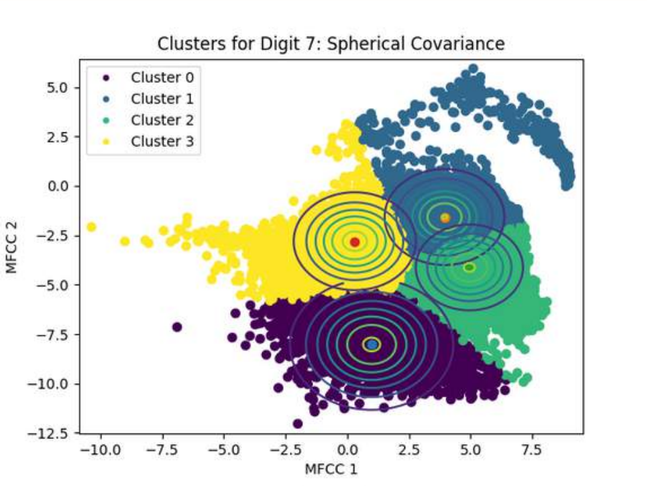


*Figure 12: Spherical Covariance*

# Other Modeling Choices

## Including Latent Variable Gender

A **latent variable** is a variable that cannot be directly observed or measured within the dataset, but is assumed to exist and can affect our accuracy. In this dataset, we know that half of the samples are male while the other half is female. The figures below show the MFCC values over time for a male and female sample. It is easy to see that there is a difference in how the MFCC values are grouped and how they transition over time. Thus, it makes sense to analyze the impact of creating **separate models for males and females** on our classification accuracy.

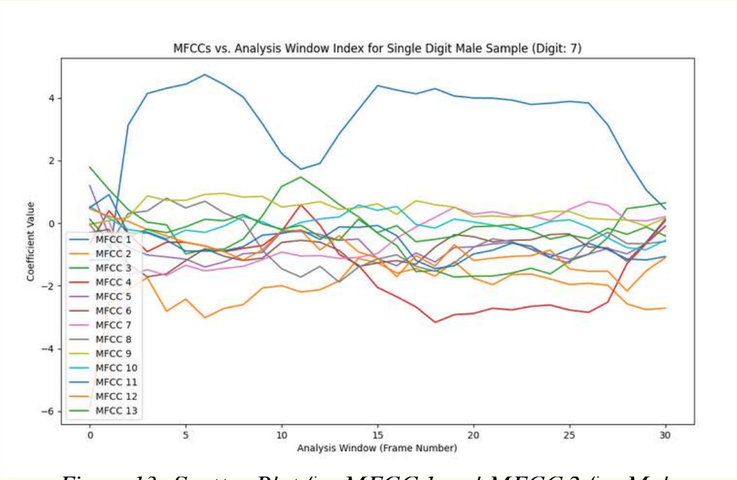


*Figure 13: Scatter Plot for MFCC 1 and MFCC 2 for Male*

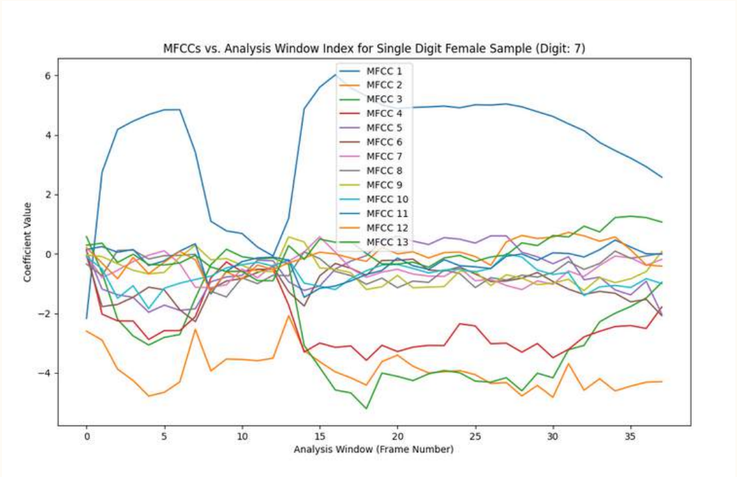


*Figure 14: Scatter Plot for MFCC 1 and MFCC 2 for Female*

# Maximum Likelihood Classification

# Maximum Likelihood Classification (MLE)

## MLE - How do we classify digits?

At this point, we have a GMM representing each digit. Each GMM has a number of mixture components equal to the number of phonemes in the word such that the GMM represents a unique set of phonemes that exist in the word. The GMM parameters are found using either K-Means or EM on the training data and the GMM takes into account all 13 MFCC values in a given digit utterance.

Now, we need to be able to predict what digit is being said given an unseen block of MFCC values. We need a classification method that takes into account the **complexity of the data** and provides a **probabilistic framework**. We can use an approach called **Maximum Likelihood Classification** (MLE). With MLE, we calculate the likelihood of observing the unseen data point (a time series of 13 MFCC values) given our GMM. We do this for each of the 10 GMMs that we have trained and select the digit that resulted in the **highest likelihood.**

# Maximum Likelihood Classification

## MLE - Equation and Implementation

The equation for calculating the likelihood of one digit is shown below. The likelihood of a time series of *N* frames of Cepstral coefficients **X** given the *M*-component mixture model parameters **Δ** and **Π** for the d-th spoken digit is

$$p(\mathbf{X}|\mathbf{\Delta}_d, \mathbf{\Pi}_d) = \prod_{n=1}^{N} \sum_{m=1}^{M} \pi_{m,d} p(\mathbf{x}_n|\Delta_{m,d}),$$

*Figure 15: Maximum Likelihood Classification*[14]

The likelihood of N frames is simply the product of the likelihoods for each frame of 13 MFCC values. This works well for our problem because it is easy to compute the likelihood of an observation given our GMMs. In practice, however, calculating the product of the likelihoods for *N* different analysis frames can be computationally inefficient and difficult to implement. Instead, I made use of *sklearn's score_samples*[15] method, which calculates the **log-likelihood** of the samples. If we calculate the log-likelihood of each frame of MFCC values, we can sum the log-likelihoods to get an equivalent expression for our MLE, since the log of a product is equal to the sum of the log of each individual element. This optimization is much **easier to implement** in practice and is more **computationally efficient.**

[14]Tantum, Stacy. "Course Project: Recognizing Spoken Digits" Duke University. Dec 2023.
[15]Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.

# Classification Results

# GMM Results

## GMM trained with K-Means - Testing Accuracy: 88.2%

The **GMM** model trained with **K-Means** has a testing accuracy of **88.2%**. The **confusion matrix** on the right displays what our model predicted for each class. The diagonal outlines correct predictions, while the other cells identify misclassifications. We can see that the model generally performed well for all digits. The highest performing digit was **6**, with an accuracy of 97.7% (215/220). The lowest performing digit was **2**, with an accuracy of 73.2%.

It classifies the digits **1**, **4**, and **6** particularly well. This may be due to those digits having a more unique set of phonemes (e.g. 1 has a ‘w’ sound that is distinct among the digits). The digits **2** and **7** performed the worst, each having more than 50 misclassifications. This may be due to those digits having a combination of more common phonemes (e.g. 7 contains an ‘s’ and ‘ah’ sound that is common among the digits). The model has a specific problem with classifying **7** as **4** (46 errors). This makes sense as the pronunciation for both of these digits are very similar.

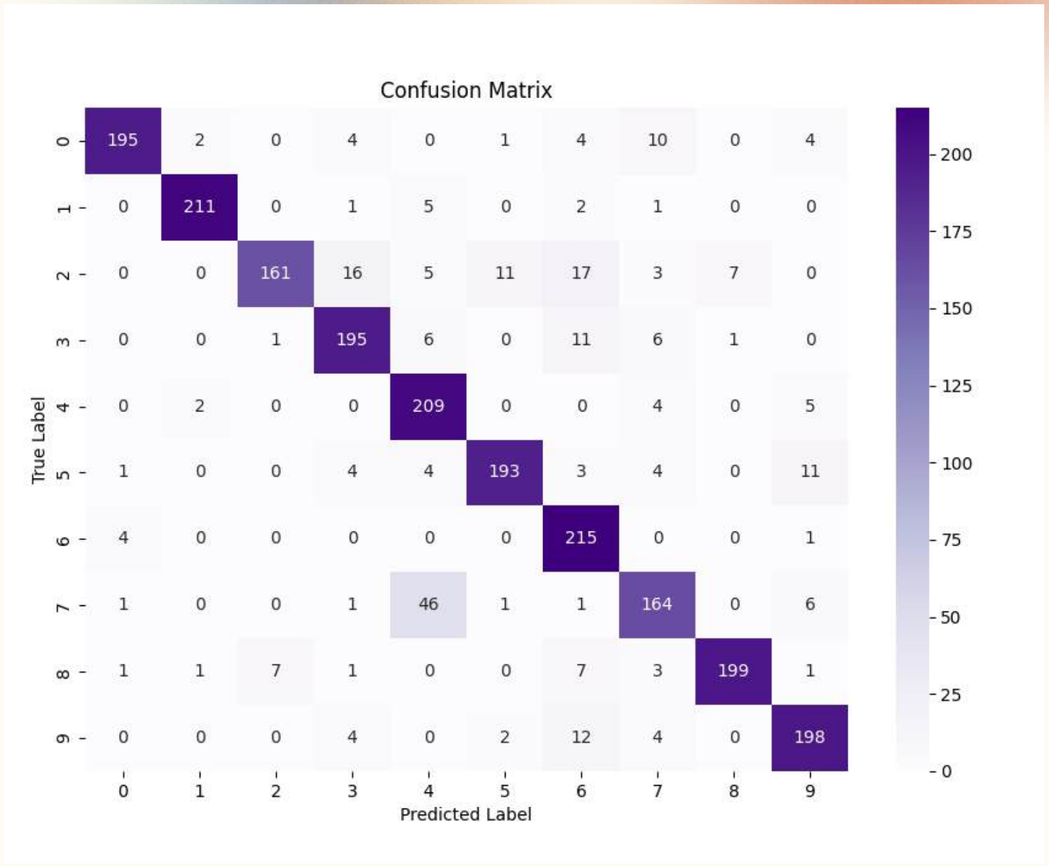


*Figure 16: Heatmap of Results, K-Means*

# GMM Results

## GMM trained with EM - Testing Accuracy: 89.7%

The **GMM** model trained with **EM** has a testing accuracy of **89.7%**. The **confusion matrix** on the right displays what our model predicted for each class. We can see that this model performs slightly better. The highest performing digit was **1**, with an accuracy of 96.3% (212/220). The lowest performing digit was **7**, with an accuracy of 70.4%.

It classifies the digits **0**, **1**, **3**, **4**, **5**, **6**, and **8** particularly well. The improvement in performance is generally present among all digits, with most of the digits having over 200 correct classifications. The digit **7** performed the worst, having 65 misclassifications. This is again due to 7 having a similar pronunciation to other digits (contains common phonemes like ‘s’ and ‘ah’). Again, there is a specific problem with misclassifying **7** as **4**, probably due to their similar sounds.

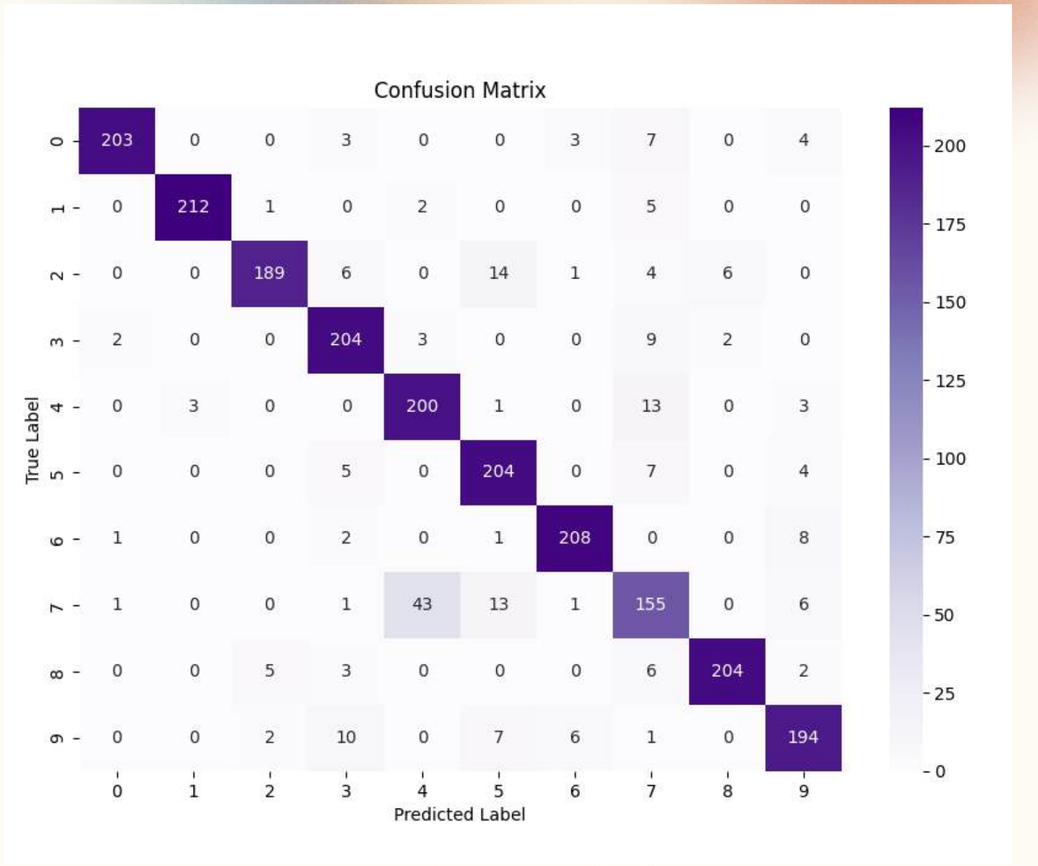


*Figure 17: Heatmap of Results, EM*

# GMM Results

## EM performs slightly better

The table on the right shows the **testing accuracies** of 5 models trained with K-Means and 5 models trained with EM (varying random initial state). On **average**, the GMM trained with EM had an accuracy of 88.7%, while the GMM trained with K-Means had an accuracy of 87.6%, a **marginally better performance.**

The slight performance boost from EM is due to many varying factors. Some of these include the fact that EM is less sensitive to **initialization** than K-Means, making it more **robust** to changes in random state. EM also **guarantees convergence** to a local optimal, whereas K-Means simply works to minimize variance within a cluster. EM provides a **probabilistic framework** that allows for increased uncertainty, functioning better with data points that fall in overlapping clusters. However, EM takes significantly **more time** to train than K-Means.

| **K-Means** (Test Acc.) | **EM** (Test Acc.) |
|---|---|
| *88.2* | ***89.7*** |
| *87.3* | ***87.8*** |
| *87.7* | ***89.0*** |
| *87.8* | ***88.4*** |
| *87.2* | ***88.4*** |
| *Avg: 87.6* | ***Avg: 88.7*** |

*Figure 18: Testing Accuracies for 5 Trials*

# Further Observations

## Number of MFCCs to Include - Results

In order to test the effect of changing the numbers of MFCCs we include in our model, I trained multiple models (GMM trained with EM) with a decreasing number of dimensions in its input features. The results are outlined in the table below.

| MFCC Window | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Test Acc. | 88.5 | 89.7 | 89.5 | 86.2 | 85.9 | 85.8 | 80.8 | 80.0 | 78.6 | 72.7 | 72.3 | 50.8 | 36.3 |

*Figure 19: Testing Accuracies of Different Number of MFCCs Included*

When we initially decrease the number of MFCCs included in the model to 12, there seems to be a **small performance boost**. This continues until the model uses 10 MFCCs. At this point, decreasing the number of MFCCs in the model negatively impacts performance and decreases classification accuracy. The performance boost by using 12 MFCC features as input may be due to the **improved regularization**. Removing a feature may prevent our model from **overfitting** to the training data and allow it to **generalize** to a broader distribution of features.

# Further Observations

## Constraining GMM Covariance Matrices Estimates - Results

I experimented with three different types of covariance matrices - **full**, **diagonal** and **spherical**. I also measured the classification accuracy on both types of training methods - **K-Means** and **EM**. As expected, restricting the covariance matrix does lead to a **decrease in accuracy**. Diagonal covariance had a slight decrease in accuracy from full covariance. But spherical covariance seemed to bring accuracy down significantly. This makes sense as restricting the **number of parameters** we have decreases the **bias** within our model, preventing it from picking up patterns within the data. It is interesting to note that an EM with a diagonal covariance performs better than K-Means with a full covariance. This highlights the ability of EM to overcome restrictions in parameters and generalize better than K-Means.

| | K-Means | EM |
|---|---|---|
| **Full Covariance** | *87.7* | ***88.4*** |
| **Diag Covariance** | *87.1* | ***88.2*** |
| **Spherical Covariance** | ***79.1*** | *75.6* |

*Figure 20: Varying Covariance Matrix and GMM Estimation Types*

# Further Observations

## Including Latent Variable Gender - Results

In order to create a model that is **gender aware**, I trained two GMMs (with EM) per digit - one with **male** samples and another with **female** samples. My maximum likelihood classification would calculate the likelihood of a new input against all 20 GMMs (2 models/digit x 10 digits) before selecting the class with the highest likelihood. I also wanted to test the effects that gender awareness had given **varying covariance estimation** types. Having a gender aware model seemed to **negatively impact** a GMM when trained with full covariance. This may be because EM with full covariance outfits our model with high enough bias and adding more specificity within our classification criteria results in **overfitting**. However, gender awareness seems to **improve the accuracies** of GMMs trained on diagonal and spherical covariances. A gender aware diagonal covariance GMM achieves the **highest accuracy** (90.6%) thus far.

| | Without Gender | With Gender |
|---|---|---|
| **Full Covariance** | ***89.8*** | *87.5* |
| **Diag Covariance** | *88.2* | ***90.6*** |
| **Spherical Covariance** | *76.0* | ***83.0*** |

*Figure 21: Varying Covariance Matrix and Gender Awareness within Model*

# Conclusions

# Conclusions

## What modeling choices were important?

**The number of MFCCs should be properly calibrated.** Initially, I believed that having all 13 MFCC values would naturally mean that the model would have the most information and, thus, the highest accuracy. However, removing the last coefficient actually improved generalization and reduced overfitting. Calibration is required to ensure we are optimizing for the right dimensionality of input features. In practice, however, this can be challenging as recomputing model accuracy for all options of dimensionality can be computationally inefficient.

**Covariance type significantly affects accuracy.** As expected, restricting the shape of our covariance matrix by decreasing the number of parameters we have ended up negatively affecting the classification accuracies. This makes sense as we reduce bias within the model in favor of higher variance. The difference in accuracies between full and diagonal covariances was marginal, but the negative impact of spherical covariances was noticeable due to how limiting spherical covariances are on modeling the distribution of our data.

**Combinations of regularization techniques should be computed.** I believed that a GMM with full covariance and gender awareness would have the highest accuracy. Instead, the model with the highest accuracy ended up having a diagonal covariance and gender awareness, despite observing a trend that restricting covariance shape decreases classification accuracy. Again, this most likely is due to the latter model having more access to information while also being robust to overfitting.

# Conclusions

## What modeling choices were not important?

**Expectation Maximization only marginally performs better than K-Means.** Despite having a number of advantages over K-Means, such as being more robust to initializations and increased performance with overlapping clusters, EM only provides a 1~3% increase in classification accuracy. EM also takes significantly more time to train, indicating that K-Means may be a more viable option if computational resources are scarce.

**Full Covariance with gender awareness did not improve our model.** Interestingly enough, adding more information to our GMM with a full covariance matrix actually hurt its performance. As mentioned before, this is most likely indicative of overfitting to our training data. We have reduced bias even further by adding more features, but have increased the variance and sensitivity of our model to unseen data points.

# Conclusions

## The Optimal System

### Testing Accuracy: 91.2%

If we wanted to **maximize** our classification accuracy, we would train a **GMM with EM** and **maximum likelihood classification** on only **12 MFCC input features**, **diagonal covariance** and **gender awareness**. Doing so gives us a testing accuracy of **91.2%**.

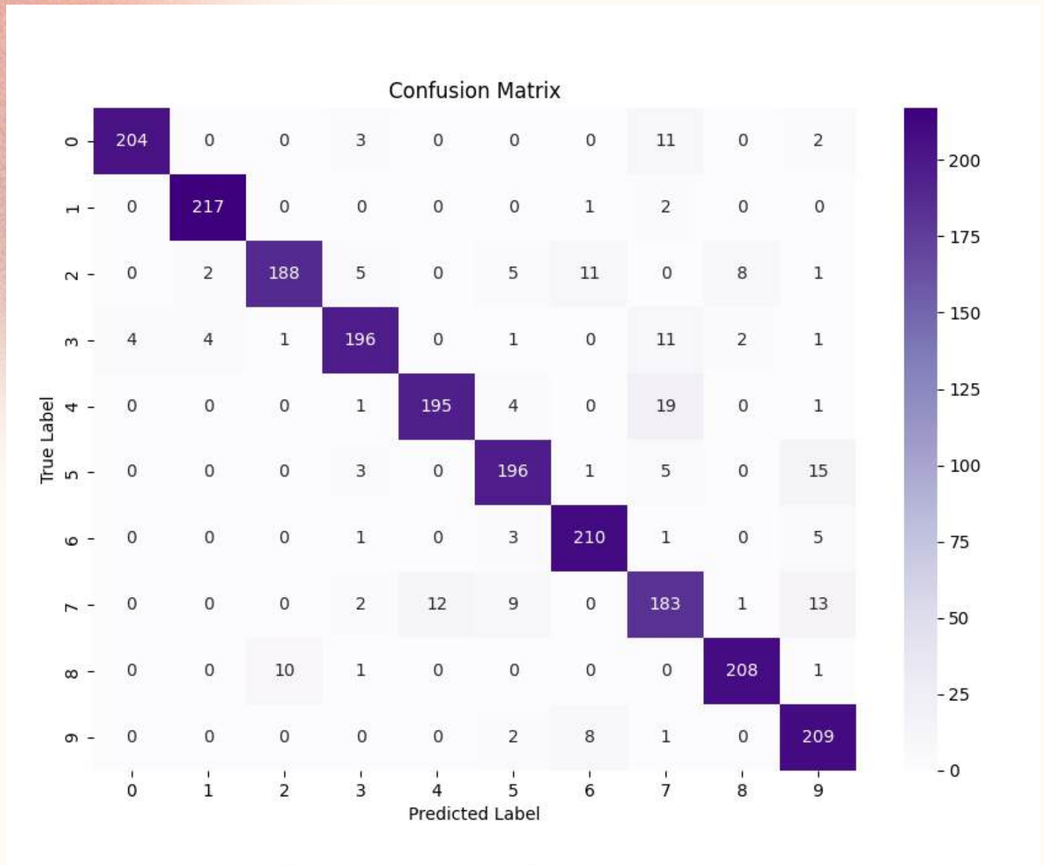


*Figure 22: Heatmap of Results, Optimal Model*

# Conclusions

## About the Modeling/Classification System

**Great things about the system -** This data modeling and classification system provides **flexibility** to work with a large variation of input data distributions, allowing it to work particularly well with **varying audio signals** and even adapt to **different languages.** GMMs provide us with a probabilistic inference on each class, allowing us to measure **confidence values** for each digit as well. GMMs are also easy to train and calculate, leading to **faster training times**.

**Things I would improve about the system -** Exploring **additional features** and other **modeling choices** may improve **bias** and reduce **variance**. For example, we kept the assumption that the number of Gaussian components for a digit is simply the number of expected phonemes in that digit. Changing the **number of mixture components** could improve generalization across datasets. This system also does not take into account the **order of phonemes**. For example, the words “car” and “rock” contain the same phonemes in different order. A GMM in our system would not be able to differentiate between the two.

# Conclusions

## Lessons Learned

**What would you do differently next time -** Reflecting on what I learned, I found myself having to use quite a bit of trial and error in order to reconfigure my **hyperparameters** (Number of MFCCs, covariance type). Next time, I would **automate** the process by testing through multiple combinations of model types to find the most optimal one. I would also want to increase **interpretability** in my model. Some of the findings in this project were based more in theory and assumptions rather than being motivated by observations. More **visuals**, more **metrics** and deeper **analysis** will allow for a further understanding of why certain phenomena take place within our models.

**What worked really well, and you would do the same way next time -** Implementing a GMM through ***sklearn's*** packages worked extremely well and I found no reason to implement these algorithms from scratch. Using **confusion matrices** to learn about the abilities of our model was quite effective. **Maximum likelihood classification** was also an interesting classifier to use as it effectively provides a likelihood value for each class.

# References

# References

*“If I have seen further, it is on standing on the shoulder of giants.” -Issac Newton*


## Background Information

[1]S. P. Yadav, A. Gupta, C. Dos Santos Nascimento, V. Hugo C. de Albuquerque, M. S. Naruka and S. Singh Chauhan, "Voice-Based Virtual-Controlled Intelligent Personal Assistants," 2023 International Conference on Computational Intelligence, Communication Technology and Networking (CICTN), Ghaziabad, India, 2023, pp. 563-568, doi: 10.1109/CICTN57981.2023.10141447.
[2]Z. K. Abdul and A. K. Al-Talabani, "Mel Frequency Cepstral Coefficient and its Applications: A Review," in IEEE Access, vol. 10, pp. 122136-122158, 2022, doi: 10.1109/ACCESS.2022.3223444.
[3]Bedda,Mouldi and Hammami,Nacereddine. (2010). Spoken Arabic Digit. UCI Machine Learning Repository. https://doi.org/10.24432/C52C9Q.
[4]Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.
[5]Harris, C.R., Millman, K.J., van der Walt, S.J. et al. Array programming with NumPy. Nature 585, 357–362 (2020). DOI: 10.1038/s41586-020-2649-2.
[6]J. D. Hunter, "Matplotlib: A 2D Graphics Environment", Computing in Science & Engineering, vol. 9, no. 3, pp. 90-95, 2007.
[7]Xuhua Gong, Huadong Meng and Xiqin Wang, "A GMM-based Algorithm for Classification of Radar emitters," 2008 9th International Conference on Signal Processing, Beijing, China, 2008, pp. 2434-2437, doi: 10.1109/ICOSP.2008.4697641.

## GMM Model Estimation

[8]Miin-Shen Yang, Chien-Yo Lai, Chih-Ying Lin, A robust EM clustering algorithm for Gaussian mixture models,Pattern Recognition, Volume 45, Issue 11, 2012, Pages 3950-3961, ISSN 0031-3203, https://doi.org/10.1016/j.patcog.2012.04.031.
[9]Gaussian Mixture Model. Brilliant.org. Retrieved 23:25, December 10, 2023, from https://brilliant.org/wiki/gaussian-mixture-model/
[10]Lloyd, Stuart P. "Least squares quantization in PCM." Information Theory, IEEE Transactions on 28.2 (1982): 129-137.
[11]Dabbura, Imad. “K-Means Clustering: Algorithm, Applications, Evaluation Methods, and Drawbacks.” Medium, Towards Data Science, 27 Sept. 2022, towardsdatascience.com/k-means-clustering-algorithm-applications-evaluation-methods-and-drawbacks-aa03e644b48a.
[12]Miin-Shen Yang, Chien-Yo Lai, Chih-Ying Lin, A robust EM clustering algorithm for Gaussian mixture models,Pattern Recognition, Volume 45, Issue 11, 2012, Pages 3950-3961, ISSN 0031-3203, https://doi.org/10.1016/j.patcog.2012.04.031.
[13]Expectation Maximization (EM) Algorithm. Expectation Maximization (EM) Algorithm - Computational Statistics in Python 0.1 Documentation, people.duke.edu/~ccc14/sta-663/EMAlgorithm.html. Accessed 10 Dec. 2023.

## Maximum Likelihood Estimation

[14]Tantum, Stacy. "Course Project: Recognizing Spoken Digits" Duke University. Dec 2023.
[15]Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.

## Visualizations from Other Sources

[9]Gaussian Mixture Model. Brilliant.org. Retrieved 23:25, December 10, 2023, from https://brilliant.org/wiki/gaussian-mixture-model/
[11]Dabbura, Imad. “K-Means Clustering: Algorithm, Applications, Evaluation Methods, and Drawbacks.” Medium, Towards Data Science, 27 Sept. 2022, towardsdatascience.com/k-means-clustering-algorithm-applications-evaluation-methods-and-drawbacks-aa03e644b48a.
[13]Expectation Maximization (EM) Algorithm. Expectation Maximization (EM) Algorithm - Computational Statistics in Python 0.1 Documentation, people.duke.edu/~ccc14/sta-663/EMAlgorithm.html. Accessed 10 Dec. 2023.
[14]Tantum, Stacy. "Course Project: Recognizing Spoken Digits" Duke University. Dec 2023.

## Toolboxes/Packages

[4]Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.
[5]Harris, C.R., Millman, K.J., van der Walt, S.J. et al. Array programming with NumPy. Nature 585, 357–362 (2020). DOI: 10.1038/s41586-020-2649-2.
[6]J. D. Hunter, "Matplotlib: A 2D Graphics Environment", Computing in Science & Engineering, vol. 9, no. 3, pp. 90-95, 2007.

# Collaborations

# Collaborations

*"Great things in business are never done by one person" -Steve Jobs*

**Who did you share and debate ideas with while working on this project?**

I discussed ideas with Darren Wu. We parsed through understanding the objectives and theoretical underpinnings of this project.

**Who did you share code with while working on this project?**

I did not share code with anyone on this project. I relied on tutorials, documentation from *sklearn,* and StackOverflow/Medium articles to debug and write my code.

**Who did you compare results with while working on this project?**

I did not compare results with anyone while working on this project. I felt pretty confident about my results and the changes that led to them.

**Who did you help overcome an obstacle while working on this project?**

I did not help anyone overcome an obstacle while working on this project.

**Who helped you overcome an obstacle while working on this project?**

Nobody helped me overcome an obstacle while working on this project. I found myself searching up issues and bugs on Google/StackOverflow to overcome code problems.