ECE480 Final Project

GitHub Repo: https://github.com/jcho1260/ECE480-digit_classification

Jin Cho

table of contents

introduction

Problem Overview

Overview of the problem and dataset with an emphasis on how this problem can be applied in a broader context.

Problem Description

GOAL: Explore feature modeling approaches for Gaussian mixture model (GMM) representations of cepstral coefficient distributions. Modeling these distributions will then allow for classification through maximum likelihood classification calculations of datasets extracted from Arabic spoken digits of 0-9.

Dataset Overview

This dataset consists of a pre-divided training and testing datasets, each consisting of blocks on the order of the hundreds. These blocks each represent one utterance of a digit ranging from 0 to 9 spoken in Arabic. For each utterance, the speech signal is transformed into 13 time-varying Mel Frequency Cepstral coefficients (MFCCs), split into 35-40 frames of speech. These datapoints come from 44 unique female and 44 unique male speakers and each of them recorded the 10 digits 10 times. The train dataset represents the spoken digits of 66 spoken digits, 33 being by males and 33 by females. This left 22 for the test dataset, 11 males and 11 females[1].

Each spoken Arabic digit is a combination of different phonemes combined with transitions. Each is a unique combination, and although the cepstral coefficient values vary person-by-person for the same digit, these shared characteristics of phonemes and phoneme translations can help still recognize and classify a series of MFCC's as a certain digit. In order to do so, the distribution of these MFCC's must be modeled for each digit such that, when trying to classify spoken digits using just their MFCC, we can observe the likelihood probabilities for each digit distribution. The digit with the maximum likelihood would be the classification for that one block of a spoken digit.

Estimated Number of Phonemes:

Sifir (0): 4 | Wahad (1): 5 | Ithnayn (2): 5 | Thalatha (3): 6 | Araba'a (4): 6 | Khamsa (5): 5 | Sittah (6): 4 | Seb'a (7): 5 | Thamanieh (8): 7 | Tis'ah (9): 4

1. "Spoken Arabic Digit Data Set"

Exploring the Dataset

As can be seen in the time series plots of the MFCCs for one person saying ‘sifir’ and ‘Wahad’, there is a fluctuation of MFCC. These fluctuations can be understood as different phonemes and the transitions between them. Through these plots and the listening to these spoken digits, I determined the number of phonemes in each digit, which will be used to help determine the number of Gaussians later on.

Below are also two examples of relationships between the MFCCs of one spoken digit. It is not explicit in the splitting of the spoken words into clusters or individual distributions.

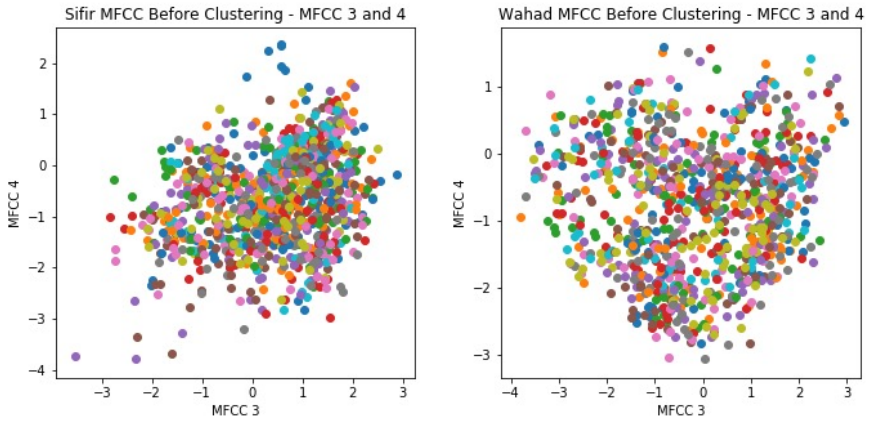


Fig 1. MFCC relationships for sifir (*left*) and wahad (*right*)

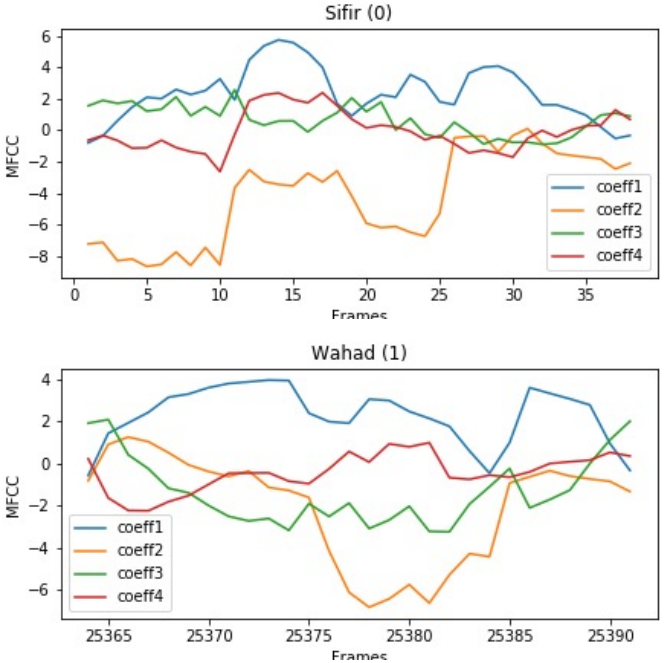


Fig 2. Time series plots of MFCCs for sifir (*left*) and wahad (*right*)

This problem and goal of classification can be broadened outside of just this case. Within the realm of sound, there are applications of examples such as voice-to-text. There is also voice recognition support in which certain words must be listened for and recognized to be responded to accordingly. Outside the realm of sounds, there are any other classification scenarios, making this process of classification widely applicable.

feature modeling

Gaussian mixture model (GMM) distribution

Exploring K-Means clustering and Expectation-Maximization (EM) to estimate the distributions representing the MFCC's of each Arabic spoken digit.

High-level understanding

Due to the ability to break down each spoken Arabic digit into a unique set of phonemes and transitions between phonemes, we can interpret this set as a Gaussian Mixture Model (GMM). These phonemes and transitions between could individually be described with weighted Gaussian distributions, and the combination of them through a word results in a GMM, each with different influences on the final distribution as described by their weights.

- - -

Goal: Through a comparison of a K-means clustering approach and an expectation-maximization approach, we will be able to find the best GMM representation of each digit and will then be able to use these distributions as classifiers for unlabeled data.

In the process of determining these distributions, there are some modeling decisions that must be made such as what parts of the dataset I will use and the number of gaussian distributions each distribution is a mixture of.

K-Means Clustering Approach

Overview:

K-means clustering creates a specified number of clusters within a given dataset in a way that minimizes the distance between the points and the respective centroid within a cluster, making it a centroid-based algorithm. This is done through a repetitive process of setting a centroid, creating clusters from there, recalculating the centroid from that center, then redefining the cluster. This is continued until the centroids or clusters don't change or until the specified number of iterations is reached[2]. There is an existing package in Python that executed these processes of creating the clusters in the scikit-learn package (sklearn.cluster.Kmeans)[3].

In the context of GMM:

Each cluster will represent a Gaussian distribution to be included in the final GMM distribution representing each digit. The parameters that must be obtained from the k-means clusters are the mean vectors, covariance matrices, and weights for each of the clusters.

Obtaining gaussian distribution parameters:

From fitting the k-means cluster model to the training data for each digit, I obtained and grouped the frames by cluster. From then, we can extract parameters like so:

- **Mean vector:** take mean of each MFCC over the frames in each cluster
- **Covariance matrix:** take the covariance of the frames in each cluster as a matrix
- **Weights**: the proportion of total frames representing the digit that are of each cluster

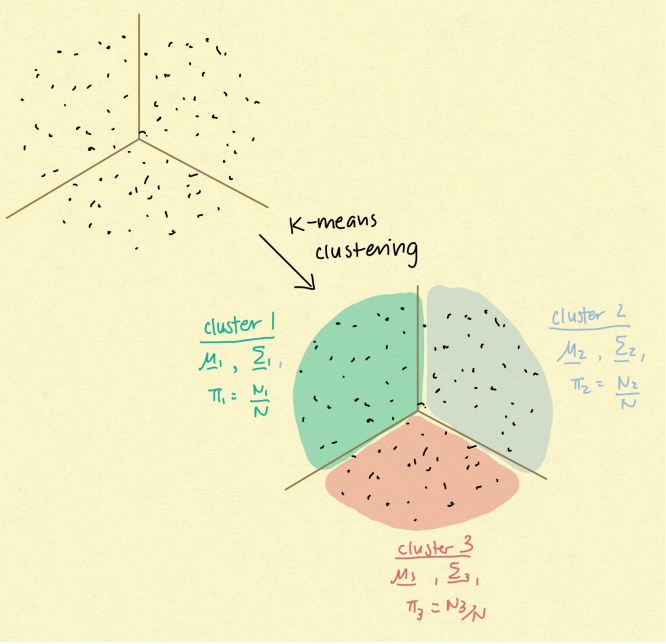


Fig 3. Interpretation of k-means clustering to derive GMM [4]

2. "Kmeans Clustering: K Means Clustering Algorithm in Python"
3. "Sklearn.cluster.kmeans.", *Scikit*
4. Tantum, Stacy. "Bayesian Model Comparison"

feature modeling

K-Means Clustering Approach Methodology

1. Determine number of clusters

I used scree plots for each digit to determine the optimal number of clusters for each

2. Derive all clusters for each dataset representing a digit

I used the built-in functions of sklearn.cluster.KMeans that fit the clusters of data

3. Extract Gaussian parameters from clusters

From the k-means model fit in the previous step, I extracted the cluster labels for each data point and grouped them by label

Determining Number of Clusters

Distortion is the distance measure used to form the clusters of data. It is calculated as the sum of squared distances of samples to their closest cluster centers[5].

A **scree plot**, also known as an elbow plot, is a visualization of the relationship between the number of clusters to the distortion values for each. From this plot, the number of clusters is determined to be the k value before the knee of the plot, or where the plot starts to equalize[5]. This is because we want to **avoid overfitting the data in the bias-variance tradeoff** by creating high variance and limited flexibility. As the clusters were determined specifically to the training data.

Methodology:

For a bigger picture visualization of where the knee of the elbow plot is, I started with 1 cluster. I went up to 1 less than double the number of phonemes, which is essentially assigning a cluster to each phoneme and transition between the phonemes. Once fitting k-means clusters for each digit, I obtained the distortion (*kmeanModel.inertia_)* for each number of clusters and plotted them[6]. From these plots, I obtained the number of clusters I should use for the final model. To the right are some examples of the scree plots created.

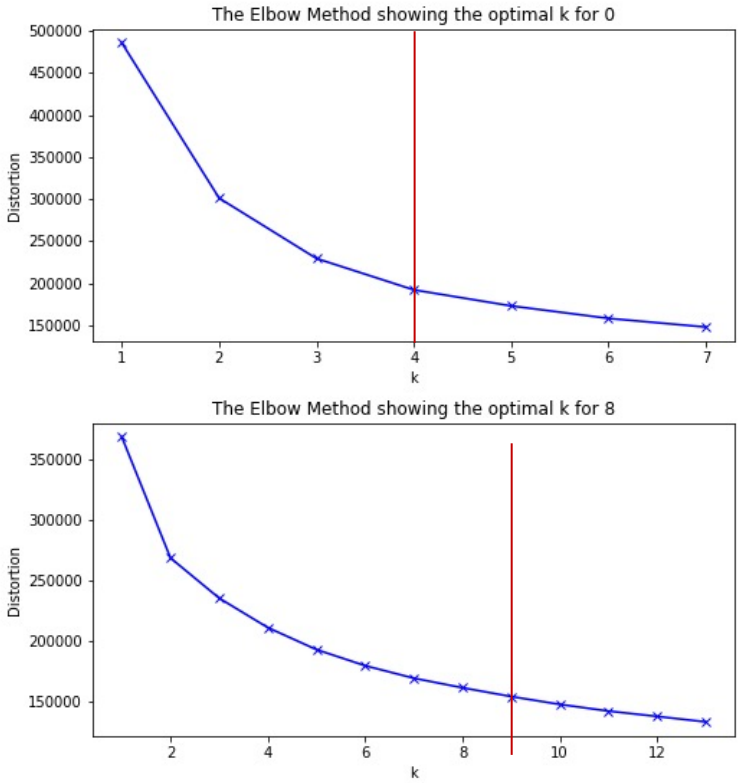


Fig 4. Scree plots for deciding # of clusters for '0' and '8'

Final cluster counts for each spoken digit:

sifir: 4 | wahad: 5 | ithnayn: 5 | thalatha: 6 | araba'a: 6 | khamsa: 5 |sittah: 4 | seb'a: 5 | thamanieh: 9 | tis'ah: 4

5. "Elbow Method."
6. Bonaros, Billy. "K-Means Elbow Method Code for Python"

Deriving Clusters for Each Digit

As explained previously, the goal of the computation of k-means clustering is to minimize the distortion until it equalizes. This process is already built into the sklearn.cluster.KMeans *.fit()* method[7]. Once a KMeans model is defined with the number of clusters, I fit it to data that represented all the frames of training data associated with each utterance.

With this computed k-means clustering, each frame is associated with a cluster label. This can also be obtained from the fit KMeans model (*model.labels_)* [7]. Examples of visualizations of how the data was clustered are to the right.

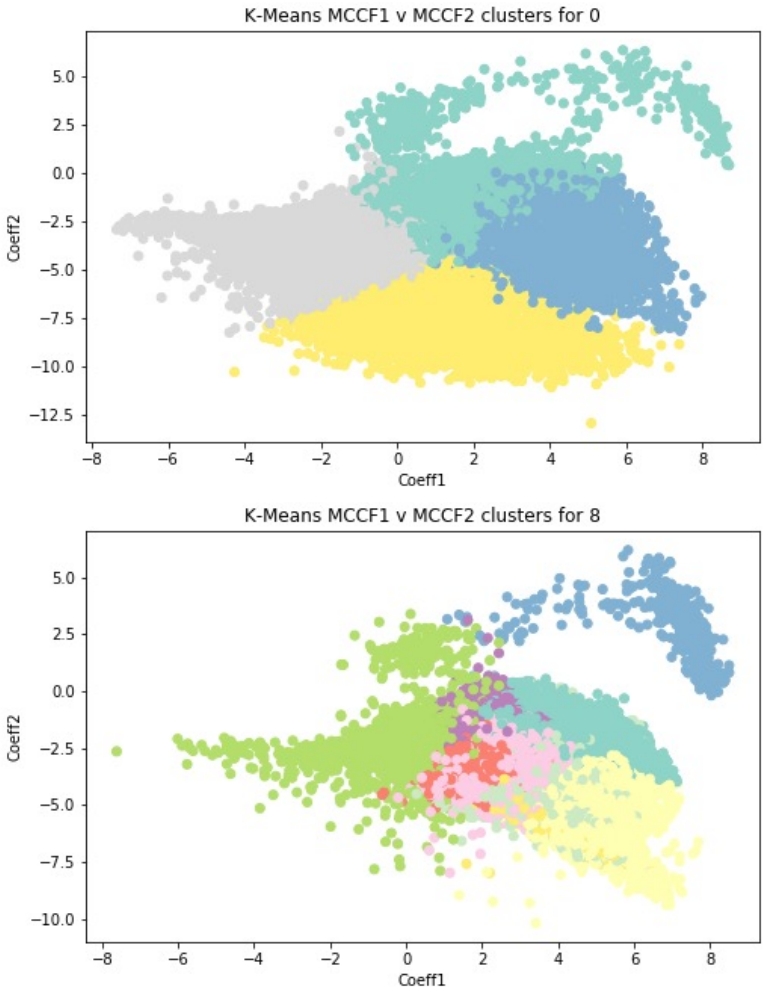


Fig 5. Resulting clusters for '0' and '8'

Limitation of K-means Clustering:

Due to the use of distortion in the optimization of the cluster divisions, there are clear boundaries around each cluster, which essentially forces a structure onto this unlabeled data. In contrast to the EM approach, which will be discussed later, it does not use a statistical approach to divide the data. This therefore influences the calculation of the mean and covariance, since both are influenced by the distance of data within clusters and the centroids. In research conducted to compare the classification abilities of these two approaches, there was also evidence that supported the conclusion that the k-means approach is less efficient/more time-consuming, which is another limitation[8].

7. "Sklearn.cluster.kmeans.", *Scikit*
8. Jung, Yong-gyu, "Clustering performance comparison using *K*-means and expectation maximization algorithms"

Extract Gaussian Parameters from Clusters

Each cluster was assumed to represent a Gaussian distribution that would come together as the GMM distribution for each digit. Therefore, the mean, covariance, and weights were obtained from the frames associated with each label. I used the built-in methods of **numpy.mean** and **numpy.cov** to calculate the vectors and matrices that would be used for each digit's GMM distribution. Weights were calculated as the proportion of all the data that fit into each cluster. This was because, similar to how the weights of each Gaussian in a GMM represents the influence of that individual distribution on the mixture model, the size of the cluster can be seen to contribute a proportional amount to the total representation of the data when the clusters come together.

These parameters are directly used in the final mixture model for each digit, which brings another limitation to this approach. It lacks the analytical method of optimization of parameters that the EM approach does have through expectation-maximization.

Expectation-Maximization (EM) Approach

Overview:

The expectation-maximization approach in deriving a GMM distribution can be split into a repeated cycle of two steps: E-step, which calculates the posterior distribution of the responsibilities each Gaussian of the mixture model has for each of the given observations, and the M-step, which calculates the parameters of each Gaussian and maximizes the log-likelihood. These two steps are repeated in cycle until convergence and those parameters define the Gaussians that make up the GMM[9].

E-Step:

Posterior calculation

$$\gamma(z_{nk}) = \frac{\pi_k N(x_n|\mu_k, \Sigma_k)}{\Sigma_{j=1}^{K} \pi_j N(x_n|\mu_j, \Sigma_j)}$$

Fig 6. Posterior calculation equation[9]

Benefits of this EM approach over K-means approach:

The k-means clustering approach is distance-focused while this EM approach determines optimized parameters through a process of maximizing the likelihood of the data given the parameters derived. With the focus of using likelihood in the parameter optimization process, the EM approach is favored and most likely to provide a more representative GMM distribution of each of the digits[9].

M-Step:

New mean calculation

$$\mu_k^{new} = \frac{1}{N_k} \Sigma_{n=1}^{N} \gamma(z_{nk}) x_n$$

Fig 7. Mean calculation equation[9]

New covariance calculation

$$\Sigma_k^{new} = \frac{1}{N_k} \Sigma_{n=1}^{N} \gamma(z_{nk})(x_n - \mu_k^{new})(x_n - \mu_k^{new})^T$$

Fig 8. Covariance calculation equation[9]

New Gaussian weights calculation

$$\pi_k^{new} = \frac{N_k}{N}$$

Fig 9. Weights calculation equation[9]

Likelihood calculation

$$ln\, p(X|\mu, \Sigma, \pi) = \Sigma_{n=1}^{N} ln \left\{ \Sigma_{k=1}^{K} \pi_k N(x_n|\mu_k, \Sigma_k) \right\}$$

Fig 10. Likelihood calculation equation[9]

9. Foley, Daniel. "Gaussian Mixture Modeling (GMM)"

EM Approach Methodology

1. Determine number of clusters

I used silhouette analysis as the method to find the optimal number of clusters

2. Derive all Gaussian parameters through EM algorithm

These were done with a built-in method in Python's *sklearn.mixture.GaussianMixture* package that uses the E- and M-steps mentioned earlier

Determining Number of Components

A **silhouette analysis** is a more generalized method from elbow plots that allows for selection of the optimized number of clusters. These values range from -1 to 1, with negative values indicating overlapping clusters (possible sign of incorrect assigning of clusters), and positive values indicating a far separation of a sample from its neighboring clusters. Therefore, these **silhouette coefficients are a measure of how far clusters are from each other/the separation between each of them**[10].

Here, similar to the respective step in the k-means approach, the range of # of clusters iterated over was **[# phonemes, 2 * #phonemes -1]**. The minimum was set to #phonemes rather than 1 since, due to the unique sound of each phoneme, there should be at least a minimum number of distributions that also equals this number of phonemes.

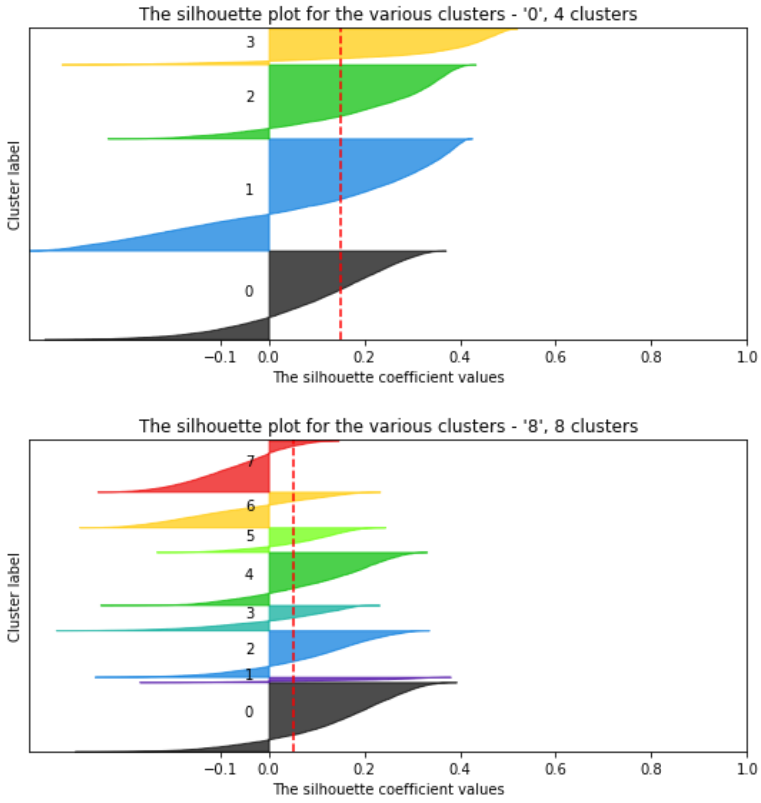


Fig 9. Silhouette plots for 0 and 8 data for the EM approach in deciding # components

Methodology:

Once fitting a Gaussian mixture model for each digit for each number of cluster I wanted to observe, I obtained the silhouette coefficient using existing methods from *sklearn.metrics.silhouette_samples and .silhouette_score* [10]. I then plotted these aggregated silhouette scores. To determine the number of clusters, I looked at which number of clusters resulted in the highest silhouette score as this indicated the least possibility of incorrect cluster assignment.

Final cluster counts for each spoken digit:

sifir: 4 | wahad: 6 | ithnayn: 6 | thalatha: 8 | araba'a: 6 | khamsa: 5 |sittah: 4 | seb'a: 5 | thamanieh: 8 | tis'ah: 4

10. "Selecting the Number of Clusters with Silhouette Analysis on Kmeans Clustering"

Deriving Gaussian Parameters

Because I used the built-in *sklearn.mixture.GaussianMixture*, when the *.fit()* was used, it already did the expectation maximization steps and the resulting parameters were the final values from the repetition of the E- and M-step calculations mentioned previously[11]. The final parameters could be extracted from this mixture model for each digit from the attributes *.means_*, *.weights_*, and *.covariances_*.

11. "Sklearn.mixture.gaussianmixture.", *Scikit*

additional model choices

Further optimizing classification model

Through exploring more aspects of the data of MFCCs and frames of utterances, we can further improve model performance.

additional model choices

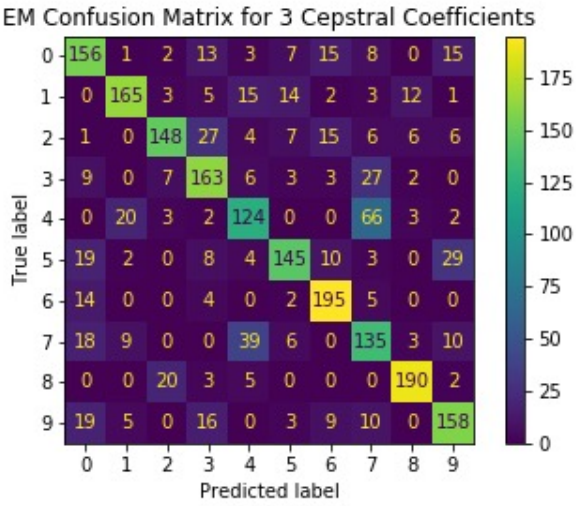
Optimizing the Number of MFCCs

Motivation:

When the sound signal s broken down into the 13 MFCC's, from the visualizations of these cepstral coefficients across the frames, it can be seen that some of the coefficients are similar in value or have a very limited range and the differentiation between the phonemes is difficult to see. Therefore, I thought that it may not be necessary to have all of the MFCC's in the datasets used. However, I also believed that there could not only be 1-2 coefficients used to represent the data, so I chose a minimum of 5 cepstral coefficients and went up t o13 in intervals of 2 to see which combination of cepstral coefficients resulted in the best performing model.

Methodology:

I created subsets of each of the train and test datasets that included the first 5, 7, 9, 11, and 13 MFCCs. With each of these, I designed and evaluated GMM distributions using both the K-means and EM approached. Although only two examples from the EM approach are pictured to the right, when observing the performance for both, overall the use of 11 cepstral coefficients resulted in the best performance. The results of the classification performance are summarized on the next page.



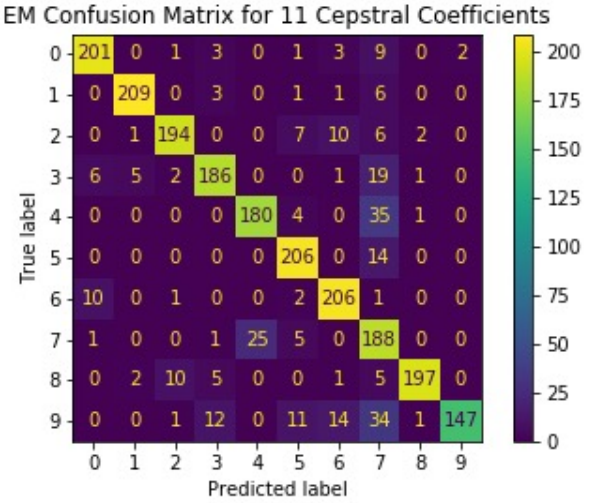


Fig 10. Confusion matrix for 3 and 11 MFCCs to optimize # coefficients

		Digits									
		0	1	2	3	4	5	6	7	8	9
# of MFCCs	5	0.82	0.95	0.57	0.66	0.78	0.82	0.95	0.56	0.78	0.81
	7	0.81	0.95	0.59	0.75	0.81	0.82	0.97	0.71	0.82	0.83
	9	0.91	0.95	0.65	0.81	0.87	0.83	0.95	0.79	0.82	0.86
	11	0.90	0.95	0.70	0.84	0.86	0.86	0.95	0.81	0.81	0.88
	13	0.87	0.94	0.67	0.82	0.84	0.85	0.95	0.80	0.79	0.86

Table 1. Classification model performance for varying number of MFCCs in dataset – k-means Approach

		Digits									
		0	1	2	3	4	5	6	7	8	9
# of MFCCs	5	0.83	0.93	0.67	0.67	0.68	0.82	0.94	0.62	0.81	0.81
	7	0.85	0.95	0.73	0.78	0.78	0.87	0.93	0.69	0.78	0.85
	9	0.91	0.95	0.80	0.89	0.83	0.89	0.91	0.75	0.90	0.82
	11	0.93	0.96	0.83	0.86	0.90	0.91	0.93	0.77	0.90	0.88
	13	0.90	0.96	0.82	0.91	0.90	0.92	0.94	0.82	0.89	0.86

Table 2. Classification model performance for varying number of MFCCs in dataset – EM Approach

Additional Feature of Gender

Although I did not implement this use of the additional feature of gender, I believe that it provides significant information about the original sound signal itself, and therefore the MFCCs. Due to anatomical differences, people of different genders are likely to say the digits differently and at different frequencies. They are also likely to me similar within gender more than across gender. Therefore, the inclusion of gender could provide additional information for the models to train on and have a better understanding of how to describe the data with a distribution.

Methodology:

This would have been done by having an additional feature within the data that classified the frame as one from a male or female, which was information provided in the dataset description. Since this is known information for both the test and train data, we would be able to do this for both, giving the model more information to train on and look for in the test data in the prediction and classification steps.

maximum likelihood classification

Utterance Classification Method

With the GMM distributions of each digit, I used maximum likelihood classification to classify the test data as respective utterances 0-9 and analyzed their classification performances.

maximum likelihood classification

Overview of Calculations and Concepts

Likelihood is the calculation of the probability or how likely the data is given a set of parameters. The maximization of this likelihood and the parameters of this maximized likelihood can be used to describe the distribution of the data at hand.

$$\log\ L(\boldsymbol{\Theta}|\mathcal{X}) = \log\ p(\mathcal{X}|\boldsymbol{\Theta}) = \sum_{j=1}^{N}\ \log\left(\sum_{k=1}^{K}\alpha_k p_k(\mathbf{x}_j|\boldsymbol{\theta}_k)\right)$$

This can be used in classification since we have a set of different parameters, each representing a GMM of a digit 0-9. We can see the likelihood of the distribution being a representation of each of the test datasets, and the one with the greatest log likelihood would be the closest distribution representation to that dataset. This would then be the prediction for classification of this digit.

Conceptual Breakdown:

Since the GMM is a mixture of Gaussians that hold different weights to the total distribution of the data, we must look at each Gaussian and its likelihood separately for each data point. Therefore, if you look at the likelihood calculation above, we are taking the sum of the weighted likelihoods for each Gaussian for all of the data.

12. Ari, Caglar. “Maximum Likelihood Estimation of Gaussian Mixture Models Using Stochastic Search.”

Application of Maximum Likelihood Classification

Steps:

1. Go through all of the GMM distributions for each of the digits and calculate likelihood for each.
2. Classify the dataset as the digit represented by the GMM with the maximum likelihood

EM Approach Likelihood Calculation:

With the use of the Python package *sklearn.mixture.GaussianMixture*, I was able to simple use the scoring method built in (*gmm.score(test_data)*). This is a calculation of the log likelihood, therefore, in using this value for classification, we must take the exponential of the value to get the likelihood value.

K-means Approach Likelihood Calculation:

Because I did not use a package, I manually did the calculations for the likelihood for the GMM distributions defined from the k-means clustering approach. From the clustering, I obtained the means and covariance matrices to calculate the probability of each of the frames in the data, multiplied that by the weights, then took the sum. I did this over all the Gaussians in the GMM, taking the log of the likelihood for each Gaussian and summing them.

In this step, the main difficult aspect was proper manual calculation of the likelihood to be maximized for the k-means approach. However, because the logic was the same as that of the scoring function of GMM, I was able to cross-check the manual calculations with the actual score to ensure that the implementation of the likelihood calculation was correct.

maximum likelihood classification

K-Means Approach Classification Performance

K-means Parameters:

- num_clusters: [4, 5, 5, 6, 6, 5, 4, 5, 9, 4]
- Train/test data: all data but only first 11 MFCCs

Performance Analyses:

All classification models across all digits performed well with over 80% accuracy other then ‘2’. For ‘2’, it was most incorrectly classified as wither ‘3’, ‘6’, or ‘7’. This might be due to the fact that they all have similar number of phonemes and also have a similar sound or ‘s’ or ‘th’, which could have caused similarities in some frames of MFCCs. It seems that overall, many of the misclassified digits were predicted as ‘7’. ‘1’ and ‘6’ were best performing. ‘1’ has fairly unique phonemes when listening to the pronunciation which may have made it easier for the model to learn and recognize. As for ‘6’, although ’2’ was most incorrectly predicted as ‘6’, ‘6’ were not incorrectly predicted much.

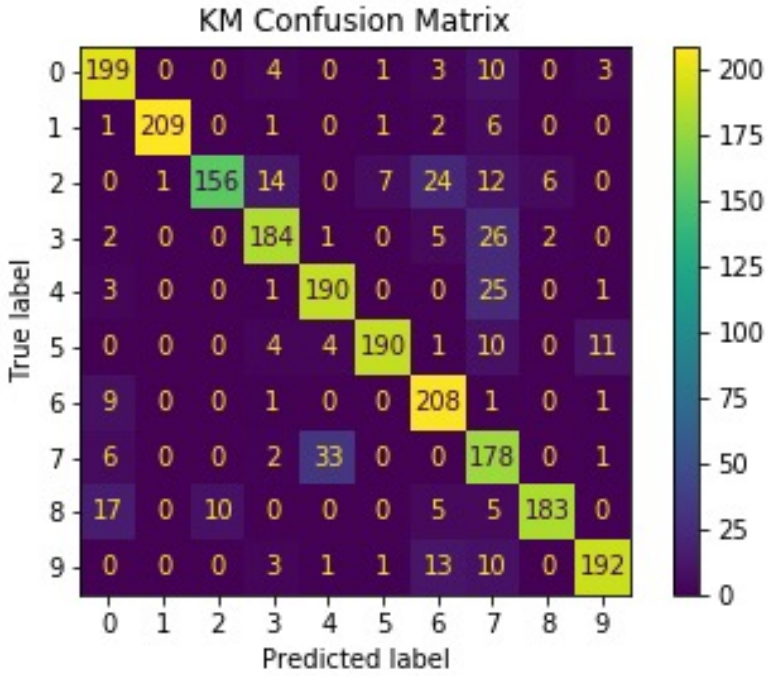


Fig 11. Confusion matrix for final model using Kmeans approacb

	Digits									
	0	1	2	3	4	5	6	7	8	9
% accuracy	0.905	0.950	0.709	0.836	0.864	0.864	0.945	0.809	0.832	0.873

Table 3. Summary of probability of correct classification – Kmeans Approach

EM Approach Classification Performance

GMM Parameters:

- num_components: [4, 6, 6, 8, 6, 5, 4, 5, 8, 4]
- Train/test data: all data but only first 11 MFCCs

Performance Analyses:

As seen in the probabilities and confusion matrix, it can be seen that this approach overall performs better than the kmeans approach to modeling GMM. In this case, the best performing ones remain the same in addition to '5', which could be an indication that overall, these digits are easier to recognize and learn by models due to their more unique and clear phoneme separations. However, the worst performing in this case was '9' and it was by quite a bit. It was most commonly incorrectly predicted as '7', which like the k-means, was the one that was used as the incorrect label the most. Both have similar endings in sound and are similar in length, although with different number of phonemes. But this similarity in sound may have made it hard for the model to separate the two, leading to inaccurate predictions.

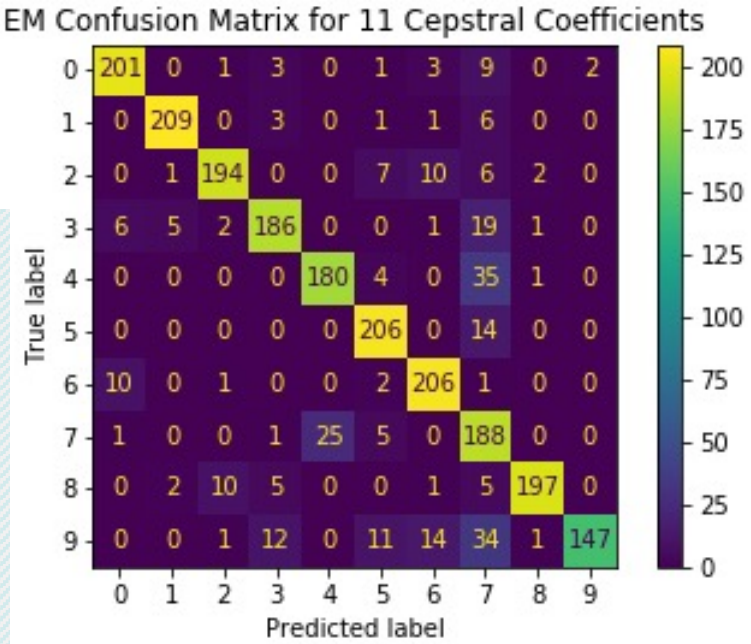


Fig 12. Confusion matrix for final model using EM approacb

	Digits									
	0	1	2	3	4	5	6	7	8	9
% accuracy	0.914	0.950	0.882	0.845	0.818	0.936	0.936	0.855	0.895	0.668

Table 4. Summary of probability of correct classification – EM Approach

conclusion

Overall Conclusions of Modeling Processes

conclusion

Conclusions

Modeling Choices:

1. Defining model distribution details: Details of the distribution are what influence aspects such as balancing the bias-variance tradeoff and therefore control how the data is learned and predicted. This is where we control the impacts and potential of underfitting and overfitting. In our case, this was controlled by the number of Gaussians we defined.
2. Dataset-related: Aspects such as the previously mentioned number of MFCCs are important to spoken digit classification because they are aspects that make up and influence the sounds. With more information on these aspects of the sound, it helps the model have more information about the data to learn, therefore helping improve model performance. However, these aspects do not define the model themselves but rather the data onto which the models are trained. Therefore, this doesn't have as large an impact on performance as the previously explained modeling choices.

Best System

I briefly touched on this throughout, but I believe the EM approach in deriving the GMM distribution for each digit is the better system in comparison to the use of k-means clustering. As stated in comparing the models in previous parts of this presentation, the k-means approach in determining the clusters and therefore the Gaussians in the mixture model is limiting and a distance-focused approach rather than an analytical approach, making it limited in accurately representing the data distributions. The EM approach to creating the model uses an analytical approach using likelihood and maximizes the likelihood of the data with the parameters it uses to represent the Gaussians. This strengthens the accuracy of the parameters.

conclusion

Reflection

Overall, I believe that I was able to learn a lot about modeling practices, and GMM and maximum likelihood classification especially, through this project. I would have liked to explore more ways to use or group the data before training the models on them to obtain the distributions to see how different factors influence the models. I think the way I structured the code in an organized manner with classes and class methods allowed for easy flow of logic in the code and allowed me to focus on the modeling process rather than line-by-line in the code.

references

Arı, Çağlar, et al. “Maximum Likelihood Estimation of Gaussian Mixture Models Using Stochastic Search.” *Pattern Recognition*, vol. 45, no. 7, 2012, pp. 2804–2816., https://doi.org/10.1016/j.patcog.2011.12.023.

Bonaros, Billy, et al. “K-Means Elbow Method Code for Python.” *Predictive Hacks*, 19 Aug. 2020, https://predictivehacks.com/k-means-elbow-method-code-for-python/.

“Elbow Method.” *Scikit - Yellowbrick*, https://www.scikit-yb.org/en/latest/api/cluster/elbow.html.

Foley, Daniel. “Gaussian Mixture Modelling (GMM).” *Medium*, Towards Data Science, 3 Jan. 2021, https://towardsdatascience.com/gaussian-mixture-modelling-gmm-833c88587c7f.

Jung, Yong Gyu, et al. “Clustering Performance Comparison USINGK-Means and Expectation Maximization Algorithms.” *Biotechnology & Biotechnological Equipment*, vol. 28, no. sup1, 2014, https://doi.org/10.1080/13102818.2014.949045.

“K Means Clustering: K Means Clustering Algorithm in Python.” *Analytics Vidhya*, 26 Aug. 2021, https://www.analyticsvidhya.com/blog/2019/08/comprehensive-guide-k-means-clustering/.

“Selecting the Number of Clusters with Silhouette Analysis on Kmeans Clustering.” *Scikit*, https://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_silhouette_analysis.html.

“Sklearn.cluster.kmeans.” *Scikit*, https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html.

“Sklearn.mixture.gaussianmixture.” *Scikit*, https://scikit-learn.org/stable/modules/generated/sklearn.mixture.GaussianMixture.html.

“Spoken Arabic Digit Data Set.” *UCI Machine Learning Repository*, UCI, https://archive.ics.uci.edu/ml/datasets/Spoken+Arabic+Digit.

Tantum, Stacy. ”Bayesian Model Comparison” Applied Probability for Statistical Learning, 10 Nov. 2021, Duke University, Durham.

notes on collaboration

Throughout the process from gaining a conceptual understanding of the processes within this process to making model choices to the final evaluation of analyses, I would talk to fellow classmate Sam Lamba. Although we didn't share code directly, we would discuss our approaches, packages we used, and our reasoning behind design choices to see how we could improve our models. We would also work together to overcome obstacles and refer to resources to answer our questions. Outside this, I referred to many online resources which I reference throughout this presentation and have listed.

thank you