Recognizing Spoken Digits

ECE 480: Applied Probability for Statistical Learning

Ethan Hsu

Duke University

Fall 2023

Outline

Project Goal

The primary objective of this project is to develop an effective and accurate model for recognizing spoken digits (0 through 9) in the Arabic language. On top of that, the project aims to develope a systematic way to analyze the given dataset and the result. The project hopes to investigate the impact of different modeling choices on achieving maximum likelihood classification of the spoken digits.

Motivation

Speech recognition holds significant importance in the advancement of technology. The complexity arises not only from the uniqueness of individual voices but also from the subtle differences in phonemic sequences. Later on, the report will demonstrate the challenges of accurately distinguishing between similar-sounding phrases. In addition, this report will also discuss on how different modeling choice can impact the accuracy of the model.

This problem is of interest to researchers and practitioners in the fields of natural language processing, machine learning, and artificial intelligence. Through this project, the reader can understand the nature of this problem, and how engineers might tackle it. Most importantly, one should be able to develope further ideas that stem from the simple model and framework this project build.

Framework

The report is written using the five stage data science pipeline from the Computing Community Consortium (CCC) Big Data Pipeline[1]. The five steps are:

1. Acquisition/Recording – Acquiring the data
2. Extraction/Cleaning/Annotation – Cleaning the data
3. Integration/Aggregation/Representation – Exploring the data
4. Analysis/Modeling – Modeling the data
5. Interpretation – Explaining the result

This pipeline is standardized in most data science work and provide clear insights on the dataset that is being worked on.

[1] Divyakant Agrawal et al. ***Challenges and Opportunities with Big Data: A white paper prepared for the Computing Community Consortium committee of the Computing Research Association***. URL: https://cra.org/ccc/wp-content/uploads/sites/2/2015/05/bigdatawhitepaper.pdf.

Acquisition and Cleaning

This project uses the Spoken Arabic Digit Dataset[2]. The dataset includes 44 males and females Arabic native speaker. Each speaker pronounce the ten digits with ten repetitions. This mean there is total 8800 time series of the speech recordings.

To parse the audio recording, the dataset encoded the information in frames with 13-dimensional Mel-frequency cepstral coefficients (MFCCs). MFCCs serve as a standard method for extracting features from audio signals. The process involves framing the audio data and applying techniques such as Fast Fourier Transform and Discrete Cosine Transform. By mimicking the frequency patterns perceived by the human ear, MFCCs effectively capture the spectral information of the signal.

In this dataset, the MFCCs are computed with the following conditions:

- sampling rate: 11025 Hz, 16bits
- Window applied: hamming
- Filter pre-emphasized: $1 - 0.97Z^{-1}$

Lastly, the dataset is pre-partitioned into a training set and testing set. The train data consists of 6600 sample, and test data consists of 2200 sample.

[2] Mouldi Bedda and Nacereddine Hammami. ***Spoken arabic digit***. tex.howpublished: UCI Machine Learning Repository. 2010. URL: https://archive.ics.uci.edu/dataset/195/spoken+arabic+digit.

Data Preparation

Data preparation and parsing is needed to perform the third step of the data pipeline – Exploration. The given dataset is provided in txt file and formatted by lines. Each line gives 13 values that correspond to the 13 MFCCs at a given frame. These lines will form a complete block that represent a single speech utterance of a certain digit. Therefore, in this project, the dataset is stored as a Python class, where each datapoint is a 2D array with shape of (frames, 13 (MFCCs)), and correspond to an integer label as well as the gender.

Exploration – MFCCs

The first thing to do when analyzing time series data is to plot the temporal progression. Below, figures of the MFCCs for digit is plotted. From these figures, certain pattern for each digit can roughly be observed, which can represent the phonems and syllabus of the data. The model built for this dataset should be capable of capturing these pattern of the MFCCs.

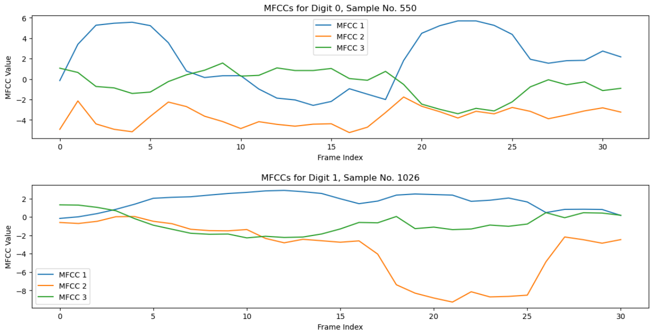


Figure 1

Exploration – MFCCs
Which MFCCs are relevant?

One important question to discuss is whether all 13 MFCCs are important to be included in the modeling. This question can be answered by comparing the result of the models, which will be done later. On ther other hand, domain knowledge can be used. From figure 2, it can be observed that MFCC 1, 2, and 3 are the one with obvious dynamic changes across time, where the rest of MFCCs seem to take similar trends and value across the time. Therefore, it will be an interesting question to explore on whether all MFCCs are relevant. Formally, the question can be answered by investigating the variable importance (feature importance) of the MFCC.

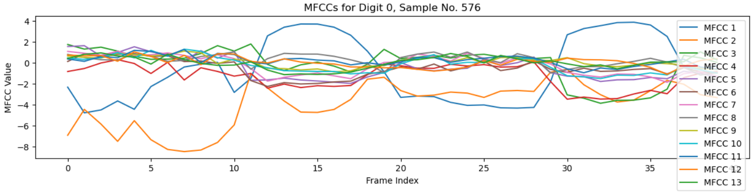


Figure 2

Exploration – MFCCs

Visualization without time window.

The model used in this project will discard the time information of the dataset, relying on the distribution of MFCCs relationship only. Therefore, it is important to visualize and understand the relationship of MFCC. In figure 3, the digit 1 and 2 are selected as example to visualize the shape difference of pairwise MFCC feature.

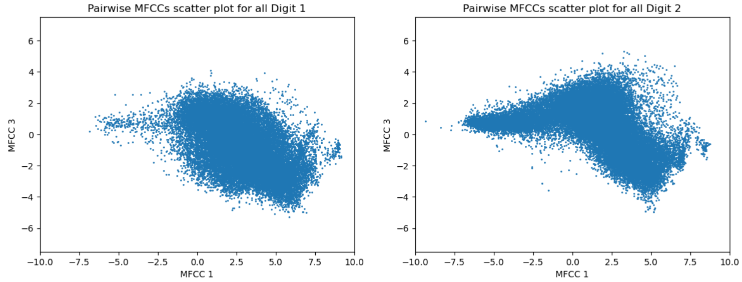


Figure 3

Exploration – MFCCs

Visualization without time window. (Cont.d)

In figure 4, the distribution difference between the two digit can be even more clearly observed when the third dimension is added. Therefore, it will be important to acknowledge that higher dimension can lead to better representation of the dataset.

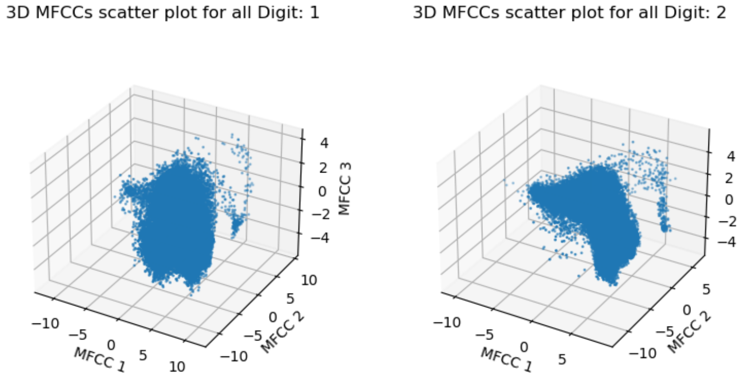


Figure 4

Goal
What should the model do?

Obviously, the goal would be building a model that is capable of accurately predicting the dataset. However, given that spoken digit recognition is not a challenging task in modern day, it is more important to use this opportunity to understand other question. From the exploration section and the nature of dataset, a few questions to be answered is listed.

- Understand how different algorithms would impact the model.
- Investigate the impact of parameter choice of the model.
- Investigate the variable of importance of the MFCCs.

Model Choice and Motivation

In this project, 10 Gaussian Mixture Model (GMM) is joined together as a large model. Each GMM will be trained on a subset of data, which should accurately capture the distribution of single digit. In the end, the joint model will predict new result by comparing the likelihood of each individual GMM. Details and formulation will be described in the next few slides.

The motivation of this model comes from a few starting point. One is that clustering method seems natural here as data point is grouped and form a clear distribution shape, as shown in previous figures. In addition, this method allow exploration to compare the number of clusters being used and other parameters. Lastly, using such model discard the time information, which mean less data preprocessing is needed. The hope is that the model will be capable of learning the phonemes, which can be used to predict the digit even without sequential relationship between them.

K-Means[3]

The K-Means clusting algorithm is quite simple to understand.

Input: Given N data points $\vec{x_1}, ..., \vec{x_N}$ and the number of cluster components K.

Output: Cluster centers $\vec{c_1}, ...\vec{c_K}$

To do this, the algorithm randomly initialize a cluster center $\vec{c_1}, ...\vec{c_K}$ and repeat the following steps for certain number of iterations or until convergence:

- Assign every data points to its closest cluster center.
- The center of each cluster is updated by calculating the average position of all data points assigned to that cluster.

The K-means algorithm is a very simple method. However, it can be seen from the algorithm that the step does not necessarily converge to a global optimum. Therefore, a common approach would be repeating the k-means algorithm with different random initialization and pick one that give the "best" clustering result. In addition, the value of K is also a hyperparameter that require cross validation to verify which value would be optimal to use.

[3] Marc Peter Deisenroth, A Aldo Faisal, and Cheng Soon Ong. "**Mathematics for Machine Learning**". en. In: ().

Gaussian Mixture Model (GMM)[4]

A Gaussian mixture model (GMM) assumes all data points are generated from a mixture of different Gaussian distribution. A GMM can be understood as generalizing k-means clustering by learning the information about the covariance structure of the data.

Any example with data points that are not perfectly spherical will demonstrate the difference between GMM and k-means.

Essentially, one can treat k-means as an algorithm to determine a good cluster center, but it does not assert any assumption or learning on the covariance shape of the data. When k-means is combined with GMM, the covariance shape of the data can be calculated using the cluster center of k-means.

In most case, the GMM is paired with the Expectation-Maximization (EM) algorithm to learn the best parameters, which will be introduced in the next slide.

[4] **A. P. Dempster, N. M. Laird, and D. B. Rubin.** "**Maximum Likelihood from Incomplete Data Via the *EM* Algorithm**". en. In: *Journal of the Royal Statistical Society: Series B (Methodological)* 39.1 (Sept. 1977), pp. 1–22. ISSN: 0035-9246, 2517-6161. DOI: 10.1111/j.2517-6161.1977.tb01600.x. URL: https://rss.onlinelibrary.wiley.com/doi/10.1111/j.2517-6161.1977.tb01600.x (visited on 12/11/2023).

Expectation-Maximization (EM) Algorithm

Formally defining the model

For the GMM model, recall the assumption that the dataset follows a mixture of Gaussian distribution, the model $g(x)$ is

$$g(x) = \sum_{k=1}^{K} \omega_k g_k(x)$$

If d is the dimensionality of the dataset, then g_k is the normal distribution with mean μ_k and a $d \times d$ covariate matrix Σ_k for each class $k = 1, 2, ..., K$. K is the total number of classses/components/mixtures, and $\omega_k \geq 0$ is the weight of each class where $\sum_{k=1}^{K} \omega_k = 1$. As mentioned before, the $\{\mu_k, \omega_k, \Sigma_k\}$ are the unknown parameters that needed to be learned to fit the mixture model.

The likelihood of the data also need to be defined for the prediction task. The log-likelihood is

$$l(\mu_k, \omega_k, \Sigma_k) = \log\left(\prod_{i=1}^{N} g(x_i)\right) = \sum_{i=1}^{N} \log\left(\sum_{k=1}^{K} \omega_k g_k(x_i)\right)$$

The EM algorithm aim to learn these parameters that minimize the maximize likelihood across all data. The algorithm is an iterative optimization method that can break into two step – Expectation step and Maximization step. The Expectation step calculates the responsibilities of each data point on each class. The Maximization step use the responsibilities to update the weighted means and covariances to "maximize" the likelihood.

The math is formally defined in the next slide.

Expectation-Maximization (EM) Algorithm (Cont.d)

For the Expectation (E) step, it computes the responsibility of each data point x_i for the class k at the t^{th} step as:

$$r_{i,t+1}^{(k)} = \frac{\omega_{k,t} g_{k,t}(x_i)}{\sum_{m=1}^{K} \omega_{m,t} g_{m,t}(x_i)}$$

where $g_{m,t} = N(\mu_m, \Sigma_m), m = 1, \cdots K$.

For the Maximization (M) step, it computes the means, covariances and weights using the new responsibilities $r_{i,t+1}$. Note that $r_{i,t}$ in the below equation is the $r_{i,t+1}$ calculated in E step:

$$\mu_{k,t} = \frac{\sum_{i=1}^{N} r_{i,t}^{(k)} x_i}{\sum_{i=1}^{N} r_{i,t}^{(k)}}, \Sigma_{k,t} = \frac{\sum_{i=1}^{N} r_{i,t}^{(k)} (x_i - \mu_{k,t})(x_i - \mu_{k,t})^T}{\sum_{i=1}^{N} r_{i,t}^{(k)}}, \omega_{k,t} = \frac{\sum_{i=1}^{N} r_{i,t}^{(k)}}{N}$$

The notation may seem intimidating, but the E step simply calculate the ratio of each data point of certain class against the sum of every class. The term $\omega_k g_k(x_i)$ appear in numerous place as it is the likelihood function, as mentioned in previous slides.

The M step is also simple. The weight, mean and covariance of each class is updated by recalculate the new "weighted average" based on the responsibility.

For GMM with k-means, the above iteration is run once to calculate the responsibilities, and the corresponding covariances and weights, whereas the mean μ_k is fixed with the k-means algorithm discussed in previous slides.

Covariance Constraint

One main design choice when training a GMM is to apply constraint on the covariance for each class. Often time, the purpose of applying such constraint is either to regularize the model for better generality, or chosen based on prior knowledge of the data distribution.

Figure 5 visualize the shape of different covariance constraint well. In this project, the full covariance, tied full covariance, diagonal covariance, and spherical covariance will be investigated, which is roughly in the order of most free to most rigid.

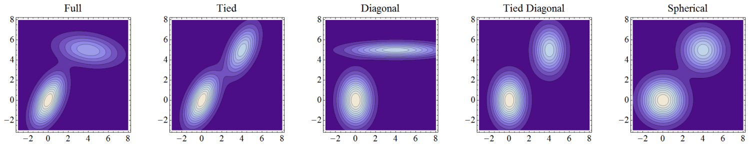
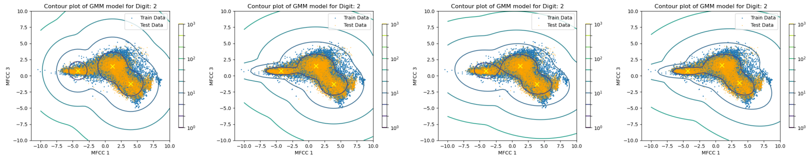


Figure 5: Plot of different covariance constraint, taken from[5]

[5] whuber (https://stats.stackexchange.com/users/919/whuber). ***Different covariance types for gaussian mixture models***. tex.eprint: https://stats.stackexchange.com/q/326678 tex.howpublished: Cross Validated. URL: https://stats.stackexchange.com/q/326678.

Covariance Constraint (Cont.d)

Figure 6 display how different covariance type can differ when applied on the same GMM model. It can be clearly observed that spherical is unable to fit the shape of the data well. On the other hand, it's difficult to tell whether diagonal covariance, tied, or full covariance is better here. It seems like diagonal covariance is capable of generalizing well such that the contour also covers the test set well. The detail accuacy will be explored later.



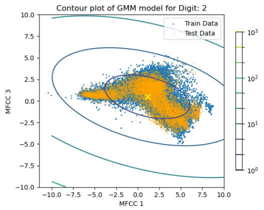
(a) Spherical (b) Diagonal (c) Tied (d) Full

Figure 6: GMM with EM fitted on train data. Both train and test data plotted.

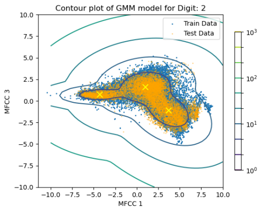
Number of Class/Component

The number of components to use for the GMM will also impact the prediction. A large number of components will result overfitting the distribution, and small number of components might not be capable of capturing the dataset shape.

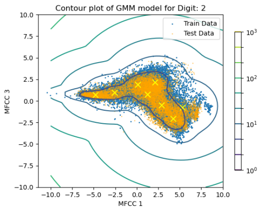
Figure 7 demonstrate how different number of components impact the result of the contour plot. Clearly 1 component is unable to fit the training data well, so the higher components method seem better. However, between 3 and 8, it is difficult to see significant improvements, so this might be an indication that 3 component is sufficient to model this particular data.



(a) 1 Component



(b) 3 Components



(c) 8 Components

Figure 7: GMM with EM fitted on train data. Both train and test data plotted.

Maximum Likelihood Classification

After covering the fundamentals of GMM, the next consideration is how to make predictions using such a clustering method. It's essential to recall the likelihood function, denoted as $l(\mu_k, \omega_k, \Sigma_k) = \sum_{i=1}^{N} \log\left(\sum_{k=1}^{K} \omega_k g_k(x_i)\right)$. This function yields a quantified measure indicating the likelihood of a specific data point belonging to the GMM.

In simpler terms, when presented with new spoken digit data, this function provides a numerical representation of how probable it is for the data to align with the trained model. Consequently, if ten GMMs are trained, each corresponding to a digit, the likelihood function quantifies the probability of the new data for each GMM. Maximum likelihood classification involves assigning the prediction based on the highest obtained value.

Formally, given 10 GMM model, the prediction is:

$$\underset{d}{\operatorname{argmax}}\, l^{(d)}(\mu^{(d)}, \omega^{(d)}, \Sigma^{(d)}) = \sum_{i=1}^{N} \log\left(\sum_{k=1}^{K} \omega_k^{(d)} g_k^{(d)}(x_i)\right)$$

The superscript (d) represent the 10 different GMM model for each digit.

Hyperparameter Selection and Cross Validation

To determine the optimal hyperparameters for the model, specifically the number of components, covariance constraint, and selected features, a cross-validation method will be employed.

Cross-validation involves dividing the training set into multiple "folds." For instance, in a five-fold cross-validation, the training data is partitioned into five subsets. In each iteration, one subset serves as the validation set, while the model is trained on the remaining four subsets. This process results in five distinct predictions. Subsequently, an evaluation metric can be employed to assess the model's performance based on the selected parameters.

The significance of the cross-validation process lies in its ability to avoid overfitting to the test set. If the test set is used during parameter evaluation, there is a high likelihood of overfitting. Moreover, in practical scenarios, the test set is typically unavailable to the model before deployment. Cross-validation offers a means to evaluate the model's performance in such situations. In fact, using test set to validate parameters is one of the most well-known ML issue known as "Leakage and the Reproducibility Crisis"[6]. Therefore, in this project, all parameter selection and evaluation will be done on the train set using cross validation method.

[6] Sayash Kapoor and Arvind Narayanan. "**Leakage and the reproducibility crisis in machine-learning-based science**". English. In: *Patterns* (Aug. 2023). ISSN: 2666-3899. DOI: 10.1016/j.patter.2023.100804. URL: https://www.cell.com/patterns/abstract/S2666-3899(23)00159-9.

Hyperparameter Space

While the idea of selecting based on cross-validation is appealing, the challenge lies in the computational inefficiency. Consider, for instance, two algorithms (EM and k-means), 10 candidate components (ranging from 1 to 10), four covariance constraints (spherical, diagonal, tied, and full), a subset of 13 MFCCs ($\sum_{n=1}^{13} \binom{13}{n}$), along with various other options like sampling rates or maximum number of iterations. The total number of computations is $2 \times 10 \times 4 \times 8191 \times \cdot \times 5$ folds, amounting to 3,276,400. Even with a one-second training time per model, this would exceed a month.

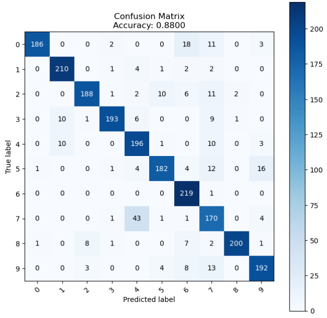
Consequently, the grid search method outlined, which scans the entire hyperparameter space, proves impractical. In this project, a more pragmatic approach will be adopted. Instead of evaluating combinations of every parameter, the focus will be on one or two parameters at a time, selecting a subset of suboptimal values. This iterative process continues until a limited number of optimal candidate values are identified. This method effectively narrows down the hyperparameter search space, rendering the computation time manageable.

Lastly, when splitting the data for cross validation, a stratified splitting method is employed. This mean each subset of the data will have equal number of data point for each labels. This guarantee a balanced dataset to work with and is a conventional method when dealing with binary or multiclass data.

Baseline Model

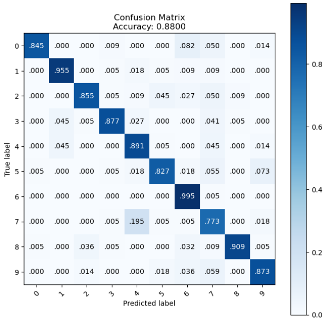
As previously stated, ten GMMs are used. The train data is divided into ten subsets based on labels, and each GMM is trained using its respective subset. In the prediction stage, the test set undergoes maximum likelihood classification. Each GMM computes a likelihood value for the given data. The digit corresponding to the model with the maximum likelihood is considered the prediction result.

In the case of the baseline model, GMM with k-means is used, the component is fixed at 3, and no covariance constraint (full) is imposed. The outcomes, as illustrated in Figure 8, reveal a decent performance for the baseline model.



True label \ Predicted label	0	1	2	3	4	5	6	7	8	9
0	186	0	0	2	0	0	18	11	0	3
1	0	210	0	1	4	1	2	2	0	0
2	0	0	188	1	2	10	6	11	2	0
3	0	10	1	193	6	0	0	9	1	0
4	0	10	0	0	196	1	0	10	0	3
5	1	0	0	1	4	182	4	12	0	16
6	0	0	0	0	0	0	219	1	0	0
7	0	0	0	1	43	1	1	170	0	4
8	1	0	8	1	0	0	7	2	200	1
9	0	0	3	0	0	4	8	13	0	192

(a) Number of Predictions



True label \ Predicted label	0	1	2	3	4	5	6	7	8	9
0	.845	.000	.000	.009	.000	.000	.082	.050	.000	.014
1	.000	.955	.000	.005	.018	.005	.009	.009	.000	.000
2	.000	.000	.855	.005	.009	.045	.027	.050	.009	.000
3	.000	.045	.005	.877	.027	.000	.000	.041	.005	.000
4	.000	.045	.000	.000	.891	.005	.000	.045	.000	.014
5	.005	.000	.000	.005	.018	.827	.018	.055	.000	.073
6	.000	.000	.000	.000	.000	.000	.995	.005	.000	.000
7	.000	.000	.000	.005	.195	.005	.005	.773	.000	.018
8	.005	.000	.036	.005	.000	.000	.032	.009	.909	.005
9	.000	.000	.014	.000	.000	.018	.036	.059	.000	.873

(b) Percentage of Predictions

Figure 8

Hyperparameter Selection – Number of Components

To maintain simplicity, the number of components for each Gaussian Mixture Model (GMM) within the joint model is the same. Figure 9 demonstrates the affect of number of components for GMM with k-means algorithm. It can be observed that the best number of components would be around 10. The shaded part of the figure display the standard deviation of the cross validation score, which will be used for later figures as well.

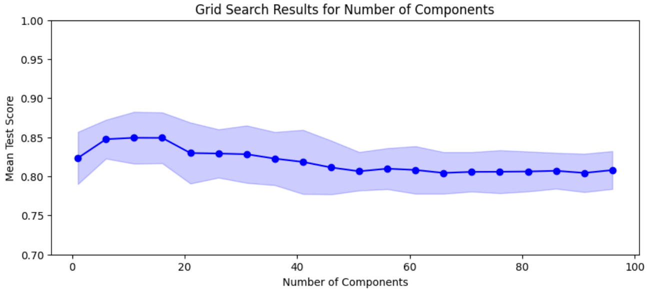


Figure 9

Hyperparameter Selection – Covariance Constraints

Figure 10 display the accuracy of the model using different covariance cosntraint. As discussed previously, spherical covariance constraint performs worst, while other covariance seems feasible. In the end, a diagonal covariance constraint is chosen as it has the best accuracy and the lowest standard deviation. This result reveals the possibility that not the most complicated parameter is needed to provide better accuracy.

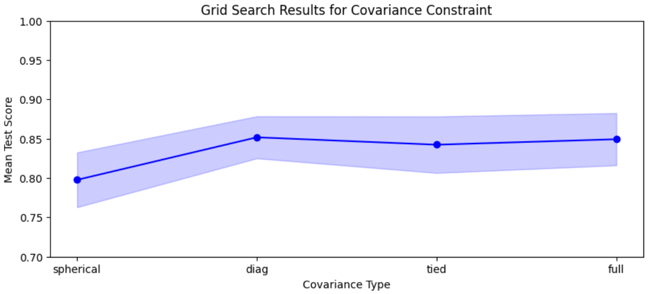


Figure 10

Hyperparameter Selection – MFCCs

So what MFCCs should be chosen? From the data exploration part, it is assumed that the lower valued MFCCs are more relevant. To make selection easier, let's assume the MFCCs are always included in sequential order, only cutoff starting from a number. So, the model may see MFCCs from 1 to 3, 1 to 9, or 1 to 13. This choice is done because it is too computationally inefficient to try out different combinations $\sum_{k=1}^{13} \binom{13}{k}$.

Figure 11 provide a great insights on how the number of MFCCs used would impact the model accuracy. An important trend to observe is that the accuracy barely increase after 8 MFCCs are selected. Note that there is possibilities that certain combination of MFCCs that is not sequential also provide great result. However, it is still clear from the figure that high number of MFCCs are needed to provide good model result.

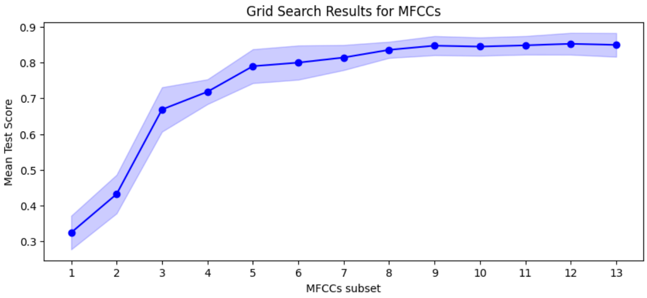
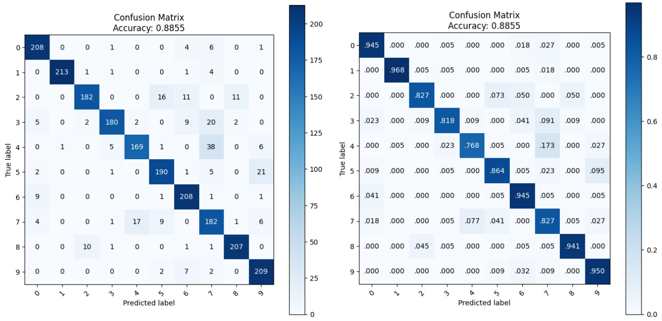


Figure 11

Hyperparameter Result for GMM with K-Means

Lastly, the algorithm choice need to be explored. Combining the parameter learned, the below is the prediction score of GMM with EM using 12 components, MFCCs 1-12, and diagonal covariance. This give the result in figure 12, which show a small boost of accuracy, but nothing great.



True \ Predicted	0	1	2	3	4	5	6	7	8	9
0	208	0	0	1	0	0	4	6	0	1
1	0	213	1	1	0	0	1	4	0	0
2	0	0	182	0	0	16	11	0	11	0
3	5	0	2	180	2	0	9	20	2	0
4	0	1	0	5	169	1	0	38	0	6
5	2	0	0	1	0	190	1	5	0	21
6	9	0	0	0	0	1	208	1	0	1
7	4	0	0	1	17	9	0	182	1	6
8	0	0	10	1	0	0	1	1	207	0
9	0	0	0	0	0	2	7	2	0	209

(a) Number of Predictions

True \ Predicted	0	1	2	3	4	5	6	7	8	9
0	.945	.000	.000	.005	.000	.000	.018	.027	.000	.005
1	.000	.968	.005	.005	.000	.000	.005	.018	.000	.000
2	.000	.000	.827	.000	.000	.073	.050	.000	.050	.000
3	.023	.000	.009	.818	.009	.000	.041	.091	.009	.000
4	.000	.005	.000	.023	.768	.005	.000	.173	.000	.027
5	.009	.000	.000	.005	.000	.864	.005	.023	.000	.095
6	.041	.000	.000	.000	.000	.005	.945	.005	.000	.005
7	.018	.000	.000	.005	.077	.041	.000	.827	.005	.027
8	.000	.000	.045	.005	.000	.000	.005	.005	.941	.000
9	.000	.000	.000	.000	.000	.009	.032	.009	.000	.950

(b) Percentage of Predictions

Figure 12

Hyperparameter Selection (EM) – Number of Components and Covariance Type

The focus now turns on to GMM with EM algorithms. Based on figure 13, 5 components and full covariance type is selected. As stated in the background section, it is expected that GMM with EM is capable of modeling the data distribuion exceptionally well. Therefore, it is unsurprising that lower number of components are needed.

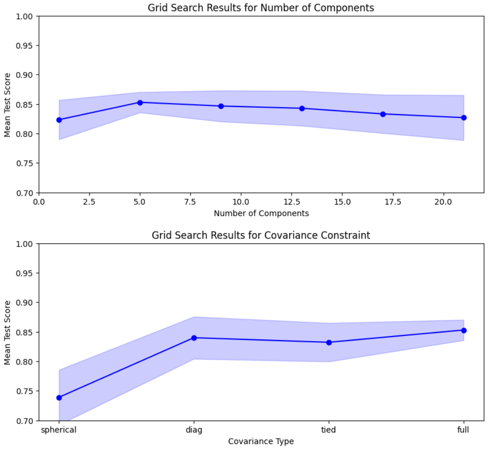


Figure 13

Hyperparameter Selection (EM) – MFCCs

Figure 14 gives similar result to the ones in kmeans. However, the optimal subset to use is now 1-11. So an additional MFCC is dropped to give better generality. It is worth noticing that for both MFCCs cross validation result, it is safe to assume that using around 8 to 9 MFCCs features are sufficient to build the model for spoken digit prediction. In this case, where a an accuracy "plateau" is reached the "elbow method" is used, which is a graphical method that select hyperparameters based on the point where the accuracy or the loss function stop increasing/decreasing drastically. In this case, 8 is where there's no more significant increase in accuracy.

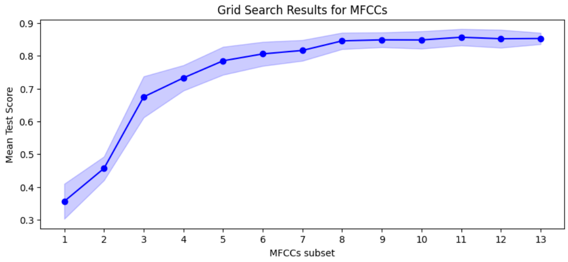
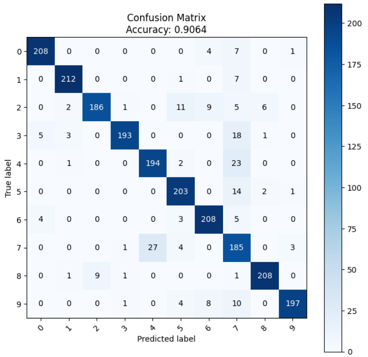


Figure 14

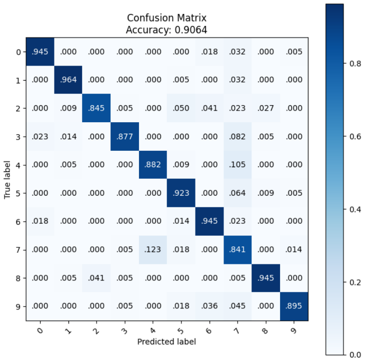
Hyperparameter Result for GMM with EM

Below, figure 15 show the accuracy of the model using GMM with EM. A great accuracy is achieved using only 8 MFCCs, 5 components, and full covariance. This provides a promising result that it is possible to use small amount of information and simple parameters to achieve high accuracy.



True label \ Predicted label	0	1	2	3	4	5	6	7	8	9
0	208	0	0	0	0	0	4	7	0	1
1	0	212	0	0	0	1	0	7	0	0
2	0	2	186	1	0	11	9	5	6	0
3	5	3	0	193	0	0	0	18	1	0
4	0	1	0	0	194	2	0	23	0	0
5	0	0	0	0	0	203	0	14	2	1
6	4	0	0	0	0	3	208	5	0	0
7	0	0	0	1	27	4	0	185	0	3
8	0	1	9	1	0	0	0	1	208	0
9	0	0	0	1	0	4	8	10	0	197

(a) Number of Predictions



True label \ Predicted label	0	1	2	3	4	5	6	7	8	9
0	.945	.000	.000	.000	.000	.000	.018	.032	.000	.005
1	.000	.964	.000	.000	.000	.005	.000	.032	.000	.000
2	.000	.009	.845	.005	.000	.050	.041	.023	.027	.000
3	.023	.014	.000	.877	.000	.000	.000	.082	.005	.000
4	.000	.005	.000	.000	.882	.009	.000	.105	.000	.000
5	.000	.000	.000	.000	.000	.923	.000	.064	.009	.005
6	.018	.000	.000	.000	.000	.014	.945	.023	.000	.000
7	.000	.000	.000	.005	.123	.018	.000	.841	.000	.014
8	.000	.005	.041	.005	.000	.000	.000	.005	.945	.000
9	.000	.000	.000	.005	.000	.018	.036	.045	.000	.895

(b) Percentage of Predictions

Figure 15

A Glance of the Result

Interpretation – Last Step of Data Science Pipeline

The model described in previous slide achieve average accuracy of 0.9064. Table 1 display the accuracy of each individual digit. It can be say that the model have difficulty predicting digit 2, 3, 4, 7, and 9 accurately. In addition, from the confusion matrix displayed in previous slide (Figure 15). It can be seen that multiple digits are often misclassified as 7. Therefore, Figure 16 is plotted by evaluating each digit's likelihood for the GMM that's built with digit 7 inputs. From the graph, it can be easily seen that digit 4 and 5 are often predicted due to their higher averaged likelihood.

Digit:	0	1	2
Accuracy:	0.945	0.964	0.845
Digit:	3	4	5
Accuracy:	0.877	0.882	0.923
Digit:	6	7	8
Accuracy:	0.945	0.841	0.945
Digit:	9	Total	
Accuracy:	0.895	0.9064	

Table 1: Accuracy of Individual Digit

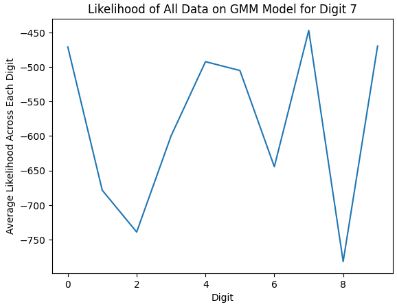


Figure 16

Question to Ask

and perhaps... an answer for them

There's a lot of interesting question to ask for the model here. Other than concluding what is the most suitable parameters based on experiments, domain knowledge and intuition is important here. A few question is asked, and some will be given an open-ended explanation that can be explored further in the future as a more in-depth project:

- What result in the advantages of choosing lower components instead of higher components?
- What cause the difference between digit accuracy?
- Why is only a subset of MFCCs required to provide a high accuracy?
- Is the model robust?
- How can the model be further improved?
- What can be done differently?

In the later slides, a few question is selected to be discussed.

Number of Components and Digit Prediction I

Does the component capture the phonetics properties?

One key clustering method design consideration is because languages essentially comprise various phonemes. Ideally, the number of phonemes should correspond to the number of components required to capture them. Figure 17 illustrates this concept, with the left graph demonstrating decent accuracy even when utilizing only the first half of frames from test data utterances. The right graph delves into a randomly selected test set sample, displaying normalized likelihoods (divided by the number of frames). Notably, a trend of higher likelihoods emerges.

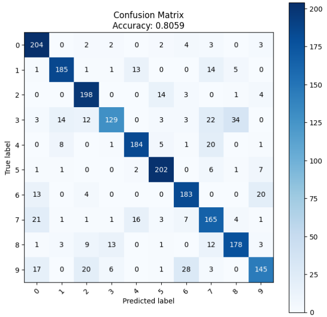
For each curve in the right graph, the number of frames are cutoff based on the fraction. When given a fraction of frames, the model's maximum likelihood is still assigned to digit 2. This implies that diminishing the number of phonemes associated with a digit, but leaving a few left, still allow the model to recognize the digit. This can be further explained when, for instance, both digits 2 and 6 share the initial sound "i," leading to a peak in likelihood for these digits. Overall, as frame information decreases, the model becomes adept at recognizing only the initial phonemes, resulting in heightened likelihoods for digits 2 and 6, but overall other digit as well. So this give some good clues on what the model is clustering, which provides strong reasoning on why a lower number of components, even if the data dimension is high (more than 10), is suitable. In addition, it also satisfy the hypothesis that clustering method is a good design choice to recognize the phonies

This result also allow us to understand why 4 and 7 seems to be always "mixed" together. There are 27 digit 7 predicted as 4 when using full frames in Figure 15, but

Number of Components and Digit Prediction II

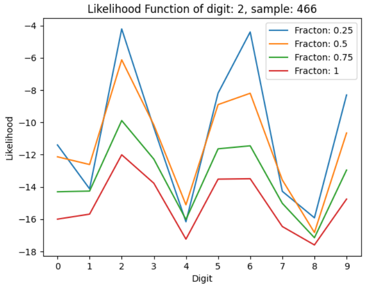
Does the component capture the phonetics properties?

only 16 of them in Figure 17(a). The digit 4 and 7 in arabic share some phonemes in common. Therefore, when cutting the frames, some phonemes are cutoff as well which allow potentially better classification. So this provides a great insights on why each certain digits have much higher accuracy.



	0	1	2	3	4	5	6	7	8	9
0	204	0	2	2	0	2	4	3	0	3
1	1	185	1	1	13	0	0	14	5	0
2	0	0	198	0	0	14	3	0	1	4
3	3	14	12	129	0	3	3	22	34	0
4	0	8	0	1	184	5	1	20	0	1
5	1	1	0	0	2	202	0	6	1	7
6	13	0	4	0	0	0	183	0	0	20
7	21	1	1	1	16	3	7	165	4	1
8	1	3	9	13	0	1	0	12	178	3
9	17	0	20	6	0	1	28	3	0	145

(a) Prediction Result



(b) Likelihood of each Digit

Figure 17: Evidence of GMM capturing phonies.

Number of Components and Digit Prediction III

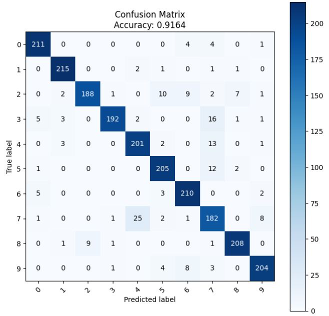
Does the component capture the phonetics properties?

What can be taken from here? This allows for further exploration within the linguistic domain to assess the clustering method's ability to generalize and recognize various phonemes in spoken language. Several project ideas can emerge from this exploration. One idea involves investigating whether the clusters in each GMM can be associated with a specific phoneme. In this approach, the GMM could not only indicate the likelihood of a given data point belonging to a distribution but also specify the cluster (and corresponding phoneme) to which it belongs. This approach offers a valuable means to comprehend how the GMM predicts each digit. Unlike the current maximum likelihood classification, which aggregates the likelihood of all frames, this proposed method leverages individual frame information for a more nuanced understanding. Some complicated model design can be developed further from this idea. For example, if this can be done, the time-variant of the data then can be utilized to assign a sequential order of phonies. This information could help improving the prediction task more.

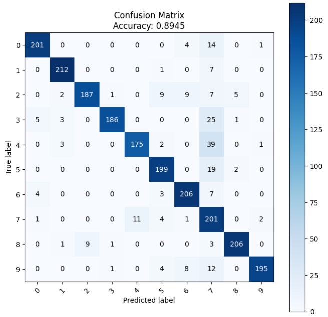
How can the Model be Improved?

What about different components for each GMM?

The current model enforces same parameters for each GMM, limiting its flexibility. Modifying individual parameters is a potential improvement, but complex dependencies among GMMs lead to unpredictable performance drops. Adjusting one parameter can make a GMM overly influential, causing misclassification of other digits. Figure 18 illustrates accuracy fluctuations, posing challenges in fine-tuning individual GMM parameters. As mentioned in the hyperparameter space, grid searching over 10 models results in an exponential (x^{10}) increase in computation time.



(a) Prediction Result with 6 Components for GMM Digit 7



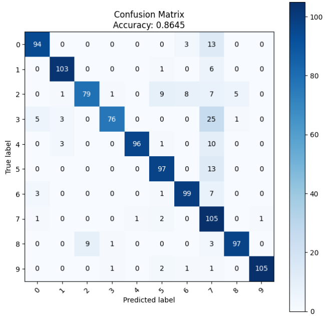
(b) Prediction Result with 9 Components for GMM Digit 7

Figure 18

Is the model robust? I

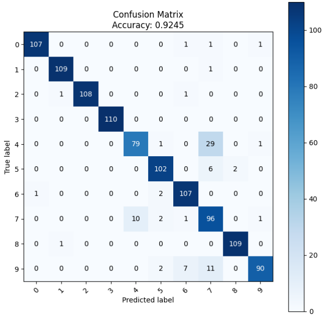
Is there bias and spurous correlation?

Building a robust model that is immune from spurious correlation and distribution shift is a difficult task. Spurious correlation describe the scenario where the model relies on non-intended features to perform prediction task. In case of multiclassification task, it is less of an obvious issue to be detected. However, in figure 19, very distinct performance difference across male and female dataset is observed.



True label \ Predicted label	0	1	2	3	4	5	6	7	8	9
0	94	0	0	0	0	0	3	13	0	0
1	0	103	0	0	0	1	0	6	0	0
2	0	1	79	1	0	9	8	7	5	0
3	5	3	0	76	0	0	0	25	1	0
4	0	3	0	0	96	1	0	10	0	0
5	0	0	0	0	0	97	0	13	0	0
6	3	0	0	0	0	1	99	7	0	0
7	1	0	0	0	1	2	0	105	0	1
8	0	0	9	1	0	0	0	3	97	0
9	0	0	0	1	0	2	1	1	0	105

(a) Prediction Result for Male Data



True label \ Predicted label	0	1	2	3	4	5	6	7	8	9
0	107	0	0	0	0	0	1	1	0	1
1	0	109	0	0	0	0	0	1	0	0
2	0	1	108	0	0	0	1	0	0	0
3	0	0	0	110	0	0	0	0	0	0
4	0	0	0	0	79	1	0	29	0	1
5	0	0	0	0	0	102	0	6	2	0
6	1	0	0	0	0	2	107	0	0	0
7	0	0	0	0	10	2	1	96	0	1
8	0	1	0	0	0	0	0	0	109	0
9	0	0	0	0	0	2	7	11	0	90

(b) Prediction Result for Female Data

Figure 19

Is the model robust? II

Is there bias and spurous correlation?

This result indicates there's issue with the model that is not robust against distributional shift of gender. This may not seems problematic when only consider the average accuracy of the model. However, for machine learning model to be applicable, these bias performance need to be fixed to reduce issues like ML fairness and other problems.

What can be done? As usual, this project provides few ideas and potential method to mitigate this bias. One approach involves altering the model's metrics. For instance, during cross-validation, the data could be assessed within the two gender subsets, and the lower accuracy would be adopted. This "worst loss" method aims to ensure the model performs adequately, at the very least, on the most challenging group.

The implications of employing this method are twofold. On one hand, it's possible that the model discovers a set of parameters that yield high performance for both genders. On the other hand, it's plausible that the model fails to converge due to "spurious correlation." In the latter scenario, one could deduce that trade-offs in gender prediction are occurring. To delve deeper into this, further analysis involving audio could be conducted. Exploring open-ended questions related to gender vocal range differences and other methods would be key to understanding the nuances of the model's performance.

How can the Model be Improved?

Time Series Model and Neural Nets

Another project idea involves employing a time series model such as a Hidden Markov Model (HMM). The HMM is a widely used method for statistical analysis of time series data and finds extensive application in areas like speech recognition. For a comprehensive understanding, this paper[7] provide great introduction. In essence, an HMM assumes that the data follows a Markov process and assigns probabilities to sequential data from the training set to effectively model its distribution.

On the flip side, contemporary technologies like Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and attention mechanisms reign supreme in the realm of speech recognition. CNNs leverage the characteristics of neighboring frames, enabling them to capture relative differences, such as peaks and valleys, in time-variant data. RNNs, on the other hand, capitalize on the sequential order of the data, allowing them to model the probabilities of phonemes occurring after specific phonemes.

[7] L. Rabiner and B. Juang. "**An introduction to hidden Markov models**". In: *IEEE ASSP Magazine* 3.1 (Jan. 1986). Conference Name: IEEE ASSP Magazine, pp. 4–16. ISSN: 1558-1284. DOI: 10.1109/MASSP.1986.1165342. URL: https://ieeexplore.ieee.org/document/1165342 (visited on 12/12/2023).

Conclusion

In this project, the final joint Gaussian mixture model comprises 5 components, full covariance, and MFCCs 1-8, achieving an impressive 90% accuracy. The significance of each parameter is thoroughly discussed throughout the project. It can be concluded that the GMM with the EM algorithm outperforms k-means marginally, with observations indicating that the baseline model performs decently already. Most of the improvements was achieved by selecting a more optimal subset of MFCCs and adjusting the component value. The covariance constraint is revealed to be less critical, as long as it is non-spherical.

Overall, this project demonstrates how a straightforward method and model can effectively address challenging tasks. Emphasizing the importance of choosing the right framework and exploring various data correlations yields valuable insights and explanatory power for the results. The evidence presented indicates that the model maintains performance even with extremely limited information (reduced frame, reduced MFCCs).

The project suggests several directions for model improvement, including experimenting with the number of components for each GMM in the model. Importantly, it opens avenues for further exploration, offering potential project ideas and research questions. In essence, this project serves as a showcase of what a simple model can achieve and prompts consideration of intriguing questions for future investigation.

Collaboration and Resources

In this project, Michael Jang is the main collaborator. The discussions have revolved around various parameters and domain knowledge that can enhance the project. Notably, Michael's model incorporates a different number of components for each Gaussian Mixture Model (GMM), leading to some accuracy improvements. The collaborative process involves the sharing of figures and numerical results, providing a comprehensive comparison to better understand the model's result. Ideas are exchanged on why certain parameters impact the model and how model behave under different feature selection.

Beyond collaboration, the framework and experimental methodology draw heavily from the one initially introduced in Duke ECE 687D Theory of Machine Learning and Algorithm. Additionally, insights from past research experiences have been integrated into the project's design and execution.

Remarks

This project not only showcases the effectiveness of a simple model in addressing complex tasks but also provides a roadmap for potential improvements and further exploration. If given another chance, the author would like to spend more time investigating the robustness of this model. This includes how may the model perform under audio signal noises, loss of information (frames and MFCCs), and immune to bias (gender). The analysis of parameters and observations lay a foundation for future project in this topic.

References I

[1] whuber (https://stats.stackexchange.com/users/919/whuber). *Different covariance types for gaussian mixture models*. tex.eprint: https://stats.stackexchange.com/q/326678 tex.howpublished: Cross Validated. URL: https://stats.stackexchange.com/q/326678.

[2] Divyakant Agrawal et al. *Challenges and Opportunities with Big Data: A white paper prepared for the Computing Community Consortium committee of the Computing Research Association*. URL: https://cra.org/ccc/wp-content/uploads/sites/2/2015/05/bigdatawhitepaper.pdf.

[3] Mouldi Bedda and Nacereddine Hammami. *Spoken arabic digit*. tex.howpublished: UCI Machine Learning Repository. 2010. URL: https://archive.ics.uci.edu/dataset/195/spoken+arabic+digit.

[4] Marc Peter Deisenroth, A Aldo Faisal, and Cheng Soon Ong. "Mathematics for Machine Learning". en. In: ().

[5] A. P. Dempster, N. M. Laird, and D. B. Rubin. "Maximum Likelihood from Incomplete Data Via the *EM* Algorithm". en. In: *Journal of the Royal Statistical Society: Series B (Methodological)* 39.1 (Sept. 1977), pp. 1–22. ISSN: 0035-9246, 2517-6161. DOI: 10.1111/j.2517-6161.1977.tb01600.x. URL: https://rss.onlinelibrary.wiley.com/doi/10.1111/j.2517-6161.1977.tb01600.x (visited on 12/11/2023).

[6] Charles R. Harris et al. "Array programming with NumPy". In: *Nature* 585.7825 (Sept. 2020). Publisher: Springer Science and Business Media LLC, pp. 357–362. DOI: 10.1038/s41586-020-2649-2. URL: https://doi.org/10.1038/s41586-020-2649-2.

References II

[7] J. D. Hunter. “Matplotlib: A 2D graphics environment”. In: *Computing in Science & Engineering* 9.3 (2007). Publisher: IEEE COMPUTER SOC, pp. 90–95. DOI: 10.1109/MCSE.2007.55.

[8] Sayash Kapoor and Arvind Narayanan. “Leakage and the reproducibility crisis in machine-learning-based science”. English. In: *Patterns* (Aug. 2023). ISSN: 2666-3899. DOI: 10.1016/j.patter.2023.100804. URL: https://www.cell.com/patterns/abstract/S2666-3899(23)00159-9.

[9] F. Pedregosa et al. “Scikit-learn: Machine learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[10] L. Rabiner and B. Juang. “An introduction to hidden Markov models”. In: *IEEE ASSP Magazine* 3.1 (Jan. 1986). Conference Name: IEEE ASSP Magazine, pp. 4–16. ISSN: 1558-1284. DOI: 10.1109/MASSP.1986.1165342. URL: https://ieeexplore.ieee.org/document/1165342 (visited on 12/12/2023).

[11] Stephen M. Stigler. “The Epic Story of Maximum Likelihood”. In: *Statistical Science* 22.4 (Nov. 2007). arXiv:0804.2996 [stat]. ISSN: 0883-4237. DOI: 10.1214/07-STS249. URL: http://arxiv.org/abs/0804.2996 (visited on 12/11/2023).

[12] Michael L. Waskom. “seaborn: statistical data visualization”. In: *Journal of Open Source Software* 6.60 (2021). Publisher: The Open Journal, p. 3021. DOI: 10.21105/joss.03021. URL: https://doi.org/10.21105/joss.03021.