(GMM) Clustering Drowsiness While Driving

Fatin Yasin Austin Labowitz

1 Introduction

Automobile crashes and incidents are the leading cause of death in the United States. Driving is an activity that requires alertness and concentration; many of these crashes are retroactively attributed to driver fatigue, drowsiness, inattention, or distraction as factors. Electroencephalography (EEG) is the recording of brain activity through scalp electrodes. The measurement of these EEG waves are a key component of this report, as we connect them to drowsiness.

This project is a binary classification task. We are getting our data from a publicly available dataset in which researchers attempted to create a drowsiness alert system for drivers using a brain computer interface (BCI), which consisted of a brainwave sensor and a mobile application. The sensor collected brain signals, which were then processed and classified. The system then seeks to generate real-time alerts through the mobile app.

The original report was not able to break an accuracy of 70% so that will be our goal to beat. A higher accuracy means less false reports, meaning more consistent alerts that will actually help the driver instead of distracting them. Our method of achieving this goal will be to use Gaussian Mixture Modelling to generate meaningful clusters that classify the data to a higher accuracy.

2 Related Works

Gaussian mixture modelling existed as a statistical method long before the advent of machine learning, as early as 1846. In 1894, Pearson wrote the first paper that explicitly referred to this as a gaussian mixture model. He introduced the concept to analyze biological data that exhibited asymmetric distributions, proposing a mixture of two Gaussian distributions to explain overlapping subpopulations without prior knowledge of their identities. Originally he used the Method of Moments to estimate the parameters of each distribution; this bode well since he was dealing with only two groups.

The method of moments was highly computationally inefficient when the amount of groups was increased. Duda and Hart, in their book “Pattern Classification and Scene Analysis” dedicated a section to mixture modeling. They proposed that soft clustering methods had advantages over hard clustering methods like K-Means, because they were able to handle overlapping clusters. In 1977, Dempster et al developed the Expectation-Maximization algorithm which proved to be robust and efficient; this paper formalized EM as the primary parameter estimation method used in GMM.

3 Data

Using the data from Kaggle, we split data into a training and testing set, roughly 70 - 30. We had **3736** total data points, resulting in a **2615 - 1121** split.

Data Augmentation : We applied a Log-Transform to handle skewness, and then applied a StandardScaler to standardize the features. This augmentation method seemed to improve the accuracy of every method tested.

Features:

- attention : Proprietary measure of mental focus from 0-100
- meditation : Proprietary measure of calmness from 0-100
- delta : 1-3 Hz of power spectrum
- theta : 4-7 Hz of power spectrum
- lowAlpha : Lower 8-11 Hz of power spectrum
- highAlpha : Higher 8-11 Hz of power spectrum
- lowBeta : Lower 12-29 Hz of power spectrum
- highBeta : Higher 12-29 Hz of power spectrum)
- lowGamma : Lower 30-100 Hz of power spectrum
- highGamma : Higher 30-100 Hz of power spectrum

4 Methods

4.1 Baselines

To assess our Gaussian Mixture Model (GMM) against established approaches, we select the four classifiers originally evaluated by Abdelmonem et al. We will compare the benchmarks our model achieves against the accuracy and parameters the researchers provide for their models.:

Ridge Regression - The only model referenced that uses a probabilistic output to classify the output in a way similar to GMM. However, the simplicity of the model and inability to completely neglect ineffective variables raises doubt about how effective it will be at capturing complex interactions between variables.

k-Nearest Neighbors (kNN) - A simpler model that assigns class labels by the k closest points. While kNN can approximate arbitrarily complex decision boundaries, it suffers in high dimensions and provides no explicit probabilistic model of the data.

Support Vector Machine (SVM) - SVM is also similar to a GMM model in the sense that it gives a location based decision rule to determine what class a new data point is a part of. We believe that GMM will perform better because of its ability to better classify overlapping data.

Random Forest - Though vastly different from GMM, might be the hardest to beat. An ensemble of decision trees grown on bootstrap-resampled subsets of the data and features. Random Forests excel at capturing high-order feature interactions and reducing variance through averaging, two things that GMMs can struggle to do in comparison.

4.2 Implementation

We implemented Gaussian Mixture Models (GMM) utilizing the scikit-learn library's Gaussian Mixture package. We attempted different alternative approaches to parameter estimation including variational inference as an alternative to expectation maximization, and employed ensemble techniques through scikit-learn's RandomForestClassifier to deal with complex interactions between features if they arose.

Gaussian Mixture Models (GMM): The primary assumption behind a GMM model is that each class in our data has a gaussian distribution; this means that each class has a mean and a covariance matrix. This gives us the ability

to find the probability that a new, unknown point is from either distribution. The final classification of the data point is defined by whichever Gaussian distribution outputs the highest probability that the point is from their distribution. For example: In a case where you have three classes, and two predictors, you will find that each class forms a 2D ellipsoid around their respective means.

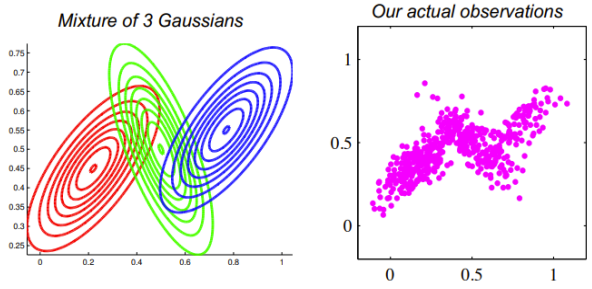

Figure 4.2: A visual representation of three Gaussian distribution's corresponding group means and covariance matrices in two dimensions[2]

At a high level, if all features in our dataset are used, each group forms a ten dimensional hyper-ellipsoid centered around the mean of their distributions.

Covariance Type

Choice of covariance type could vastly impact the performance of our model. Optimizing our covariance matrices is important to properly distribute each class in the data. The sklearn package provides access to four different types of covariance matrices to classify four different kinds of normal distributions in ascending complexity: spherical, diagonal, tied, and full.

- **Spherical:** uses equal variance with circular distributions. This struggles on complex data but is good for overfitting.
- **Diagonal:** independent variances that form ellipsoids. This is efficient and good for overfitting, but ignores interactions between features.
- **Tied:** shares the same matrix for every group, might overgeneralize.
- **Full:** adapts to each class, capturing complex distributions but needs compute, data, and regularization.

Expectation Maximization (EM) vs. Variational Inference(VI)

EM maximizes the likelihood that data points fit into their class of distribution. It adjusts the mean and covariance in each step until convergence. EM can't properly feature select, and risks overfitting the data because of the compute it demands.
VI automatically does feature selections by estimating the true posterior with a simple distribution. It updates parameters to preserve uncertainty, it does this by maximizing the evidence lower bound. VI is more sensitive and has inherent bias but could capture feature correlation better.

Meta Classification: Meta-classification is a stacking model built from multiple other models that combine the output of each to make a better classification. In our case, since our study is on GMMs, this model uses a random forest built from the log-likelihoods of each of the distributions, as well as the difference between log-likelihoods an interaction term to distinguish overlap. A Random Forest model is then used to learn the decision rules of the two classes along with the new predictor. A meta-classification model used in this way can create decision rules based on the interaction between classes in a way often beyond the capabilities of a GMM alone.

5 Results

5.1 Benchmarks

Since we took a unique approach towards processing our data, we ran the models Abdelmonem et al [1] reported; we kept the same unique parameters and these were our results.

Algorithm	Researcher Accuracy	Our Accuracy	Optimal values
Logistic regression	65.28%	65.01%	C:10 , penalty: L2
KNN	66.10%	66.09%	Number of neighbors: 29
Random forest	67.16%	66.13%	Criterion: gini, max depth: 7, max features: auto, number of trees: 200
SVM	66.73%	65.35%	C: 10, gamma: 1, kernel: rbf

The accuracies of our models were approximately the same as the initial report. This means that regardless of our data augmentation, the performance of the models was highly dependent on the chosen algorithm.

5.2 Evaluated Models

The large and complex dataset led us to believe that a traditional EM model would be a best starting place with a covariance type that fits high-dimensional data well. With this in mind, our first model was one that iterated through all possible covariance types and regularization terms 1e-6 to 1e-2 by powers of 10. Additionally, we sorted through all subset sizes because EM does not include built-in variable selection.

In alignment with our assumption that we would need a complex covariance type to capture the intricacies of the data, nearly every optimal model found had full covariance matrix type. However, when doing further testing, prioritizing this method, we ran into one of the major roadblocks when handling high-dimension analysis methods. Models using 8 or more predictors would crash because they didn't have enough data to train on. In response we restricted the maximum subset of predictors to seven. Alongside our EM model we implemented a meta-classification model made up of the log-likelihood of each of the groups along with their difference. This was then fitted with a Random Forest classifier with 500 estimators. To illustrate how each model performed on a wide array of training and testing data we ran this algorithm for each method 50 times to find the mean accuracy of each method along with the parameters that were used the most.

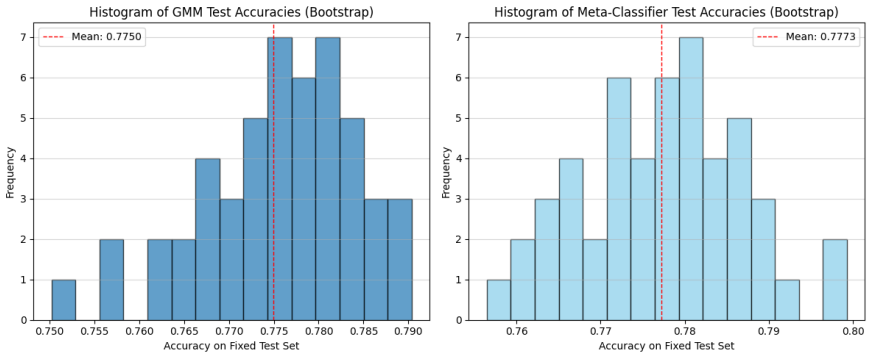

Figure 5.2.1: Bootstrap of accuracy across 50 samples for our final EM model and a meta-classifier model combining two EM models

From our bootstrap, we were able to find a mean accuracy of 0.775 for our original EM model and a mean accuracy of 0.777 for our meta-classification using an interaction term. The most used combination of parameters was one with 6 included features and a regularization term of 1e-3. The six most used features were:

- highGamma - 92% usage
- highBeta - 60% usage
- delta - 54% usage
- attention - 44% usage
- theta - 38% usage
- lowAlpha - 4.0% usage

Implementation of a variational inference model did not improve over our established EM model with an average accuracy of 0.729. However, our meta- classification model combining VI models resulted in the highest mean accuracy out of all our models with an accuracy of 0.784.

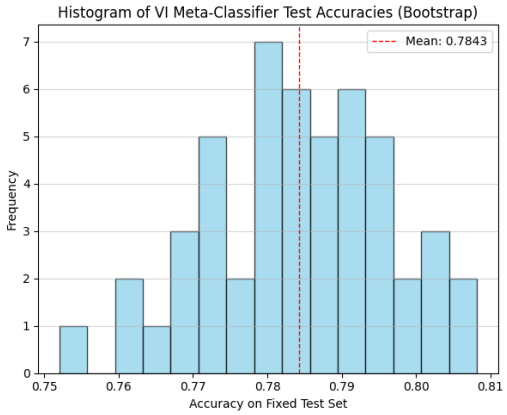

Figure 5.2.2: Bootstrap of the accuracy of a meta-classifier model combining two VI models along with an interaction term. This model performs variable selection automatically and seems to generalize the best out of all our models.

6 Conclusion

6.1 Analysis

All our models topped those of the original researchers, the original benchmark we set out to beat. However, there is still a long way to go in the realm of research on brain waves. Our hypothesis that indicators of brain function may follow a Gaussian distribution seems to have fallen somewhat short in terms of results. Regardless, machine learning has already made major headway in the field of brain research. Continued research is necessary to solve these major questions one day, no matter how big or small the goal.

6.2 Future Improvements and Implications

Our models were only able to perform up to an upper bound of 80% accuracy, which while satisfactory in regards to improving against already existing methods, fall flat in terms of usability in a real world application. In addition, extra data could also improve our models to allow us to use the full set of features consistently. Upon doing further research, we found that using deep learning and neural networks could be a worthwhile approach, however, this would come at the expense of compute and efficiency. We could improve the performance by bringing in outside domain knowledge of neurology to better understand the EEG readings beyond just numerically. Perhaps we could engineer new features or gather new data based on what readings are actually relevant to the brain functions active during driving. Being able to model the brain could yield incredible results not just in this application but predicting human behavior itself.

7 References

[1] Abdelrahman Khaled Abdelmonem, Hussein Ahmed Abdelrahman, Nada Mohamed Abdelrahman, Omar Mohamed Abdelrehim. *Software Proposal Document for Be Alert*. Supervised by Prof. Alaa Hamdy, September 10, 2023.

[2] Christopher M. Bishop. Pattern Recognition and Machine Learning. Springer, 2006. p. 433
.
[3] Pearson, K. (1894). *Contributions to the Mathematical Theory of Evolution*. Philosophical Transactions of the Royal Society of London. A, **185**, 71–110.

[4] Duda, R. O., & Hart, P. E. (1973). *Pattern Classification and Scene Analysis*. Wiley.
ISBN: 978-0471223610

[5] Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). *Maximum Likelihood from Incomplete Data via the EM Algorithm*. Journal of the Royal Statistical Society: Series B (Methodological), **39**(1), 1–38.