An EEG Error-Related Potential Decoder

Frederick
Electrical and Computer Engineering
University of Texas

Mehra
Electrical and Computer Engineering
University of Texas

Baker
Biomedical Engineering
University of Texas

***Abstract*—Data for four subjects was taken from a larger experiment for error-related potentials (ERP) collected using a 16-channel EEG cap. The data was first filtered then analyzed using sliding window analysis to extract features for building decoders for each subject, to classify ERP. The classifiers were evaluated by comparing test and train accuracies and visualizing ROC curves. It was found that training the classifiers on an additional online data session resulted in increased performance of the binary classification models.**

I. Data Exploration

To visualize the raw data, the EEG signals were extracted into 1 second trials and split into two respective classes, "Error" and "No Error", where their grand averages were calculated and visualized. From the grand average plots we see distinct differences between the classes. Most notibally that "Error" trials experience significantly more variation with a positive peak around 300ms (P300) then followed by a negative peak around 600ms (N600). These are noticeable in the grand average of figure 1, but less so in individual trials. From the difference in the two classes we chose to use min, max, and integral features, focusing on total average area (TAR) since it was most representative of the area. The P300 and N600 are well documented in literature as a part of the "oddball paradigm" that has been observed in awake subjects experiencing an unexpected trigger. Indeed they also occur in error-related potentials [1].

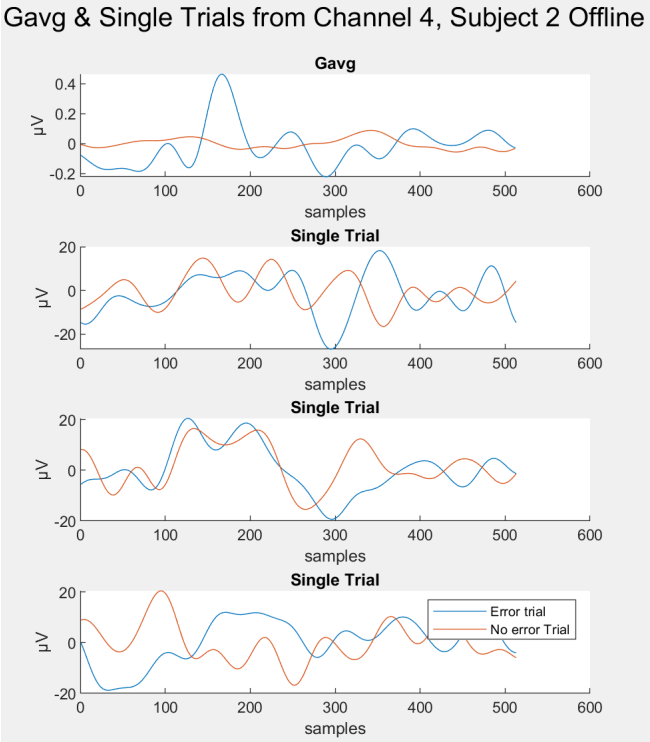

Figure 1. Grand Averages and individual trials for subject 2

II. Initial Processing

A. Spectral Filtering

For initial processing, the data was bandpass filtered from 1-10 Hz. This range was chosen based on previous literature for error-related potentials [2]. To confirm proper filtration of the correct band, Welch PSD was calculated to observe the power levels for the given range of frequency bands. As expected, power drops significantly in the frequency bands that are not of interest, as seen in figure 2. This filtering mainly helps to isolate the signals of interest and remove high frequency noise.

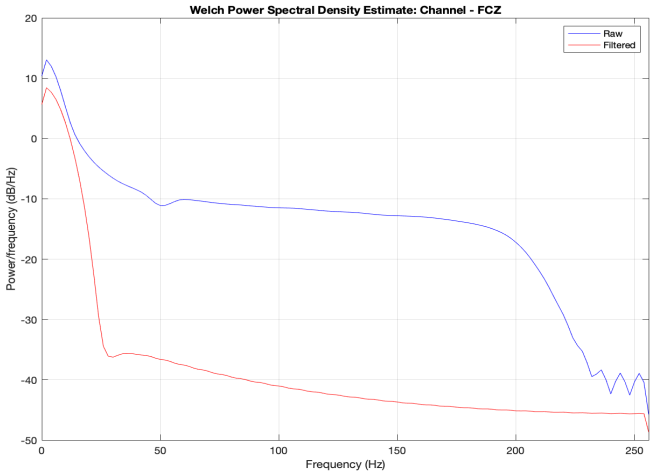

Figure 2. PSD of raw vs filtered data for FCZ channel for subject 1.

B. Spatial Filtering

Two types of spatial filtering methods were explored in this analysis: Common Average Reference (CAR) and Common Spatial Patterning (CSP). Each filter was evaluated by observing SNR of the grand averages under the processing of each filter. Specifically, SNR in dB was calculated where the signal of interest was the grand average of the signals corresponding to the "Error" class, while the noise was the grand average of the signals corresponding to the "No Error" class. Specifically, we chose "No Error" to be equivalent to baseline since the ultimate goal is to separate the two classes with maximum discriminability. The SNR here is a way to measure the magnitude of difference between the baseline "No Error" case and the signal of "Error" and see which spatial filtering method would be best for helping distinguish between them. CAR spatial filtering demonstrated to be the most effective in increasing SNR as seen in Figure 3 and thus separating the two classes and was used for each subject.

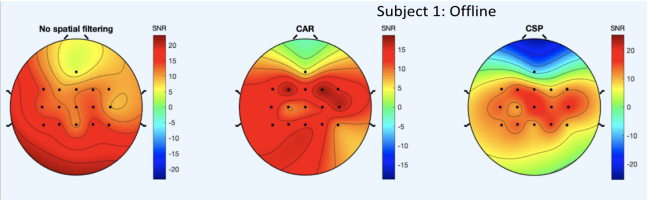

Figure 3. Topo plot evaluating SNR (in dB) after spatial filtering methods comparing: No filtering, CAR, and CSP, for subject 1 offline data.

III. Feature Extraction

To extract features of the EEG data, we chose to focus on the temporal features since error related potentials are time locked signals based on the start of a stimulus. Based on the grand average plots and previous literature on P300 signals, most of the discriminatively should be contained in the first 750 ms of the signal. To account for variations across subjects, we use a sliding window analysis with hyperparameters window size and overlap. From each window, we calculate the min, max, mean, and integral features. For integral features we tried the positive area (PAR), negative area (NAR), and total area (TAR). In a separate study, the TAR feature was found to be highly discriminant in by an independent sample t-test with 2 grouping variables [3].

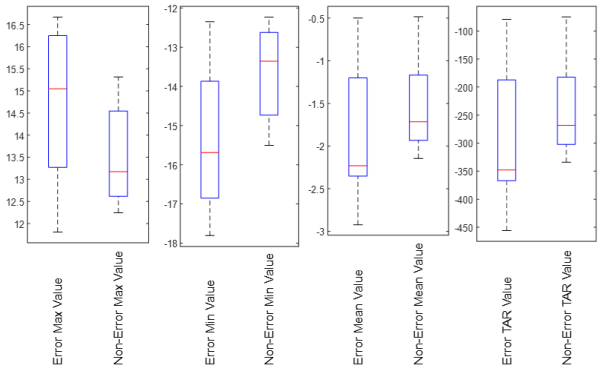

Figure 4. Bar chart showing the discriminant values for max, min, mean, and TAR for the error and no error trials. These have been averaged for all subjects across all trials.

A. Sliding Window Analysis

As part of our analysis, we decided to vary the widow size to see which window size and overlap allowed for the best discrimination between states. The window size varied between 0.1s and 1s, with an overlap between 0 and 50%. We ultimately used a window size of 0.2s with 0 overlap to do our final classification.

B. Feature Visualization

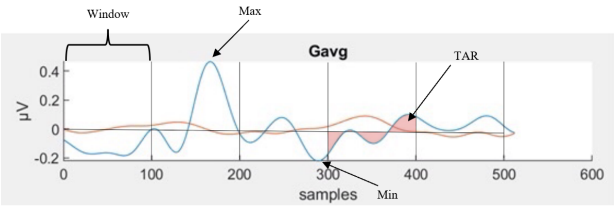

Figure 5. Example of the features used for discriminization on the grand average plot.

On the grand average plot, the features used for discriminaztion were fairly clear (Figure 5).

Feature	MAX	MIN	MEAN	TAR
p-value	0.0999	0.0120	0.175	0.175

Table 1. P-values comparing the error and no error trials showing statistically significant differences with alpha=0.05.

C. Feature Scoring

Feature significance was determined by calculating their weights using neighborhood component analysis (FSCNA). This method is non-parametric and is used for maximizing prediction accuracy of regression and classification algorithms [4]. We presumed this to be a reasonable alternative to fisher score. To verify our features are reasonable and stable across sessions we scored our features for every session of a subject individually and then averaged them across the three sessions. In this analysis we notice a few things. Firstly, a larger window size tends to lead to higher feature weights. Second, FSNCA feature weights were highly dependent on the number of features, number of data points, and range of feature scores. Lastly we also found that FSNCA would produce vastly different features scores when rerun on the same data. (Figure 6)

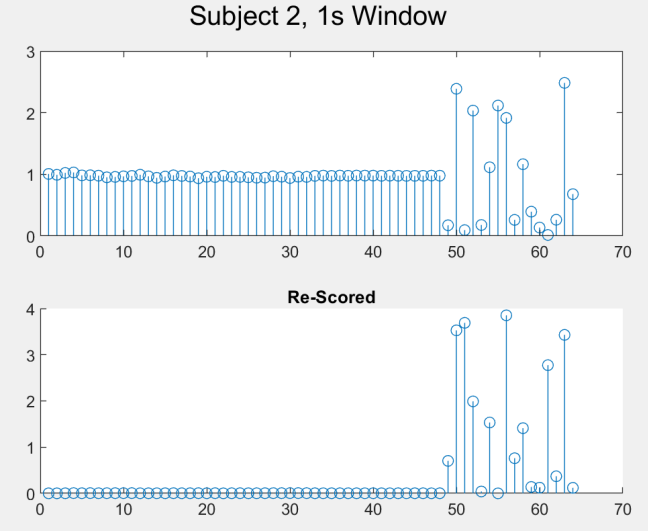

Figure 6. Feature weights calculated on all sessions of subject 2. 64 total features: 16 channels x 4 feature types x 1 window. 1-16 Max, 17-32 Min, 33-48 Mean, 49-64 TAR. Contrary to figure Y TAR features score best

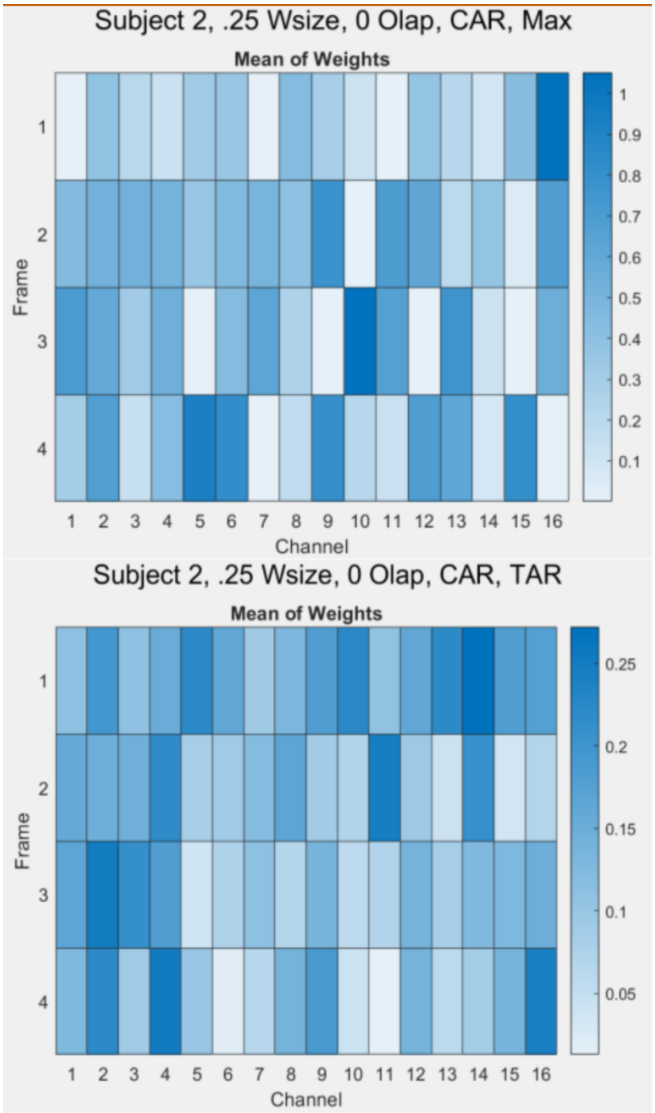

Figure 7. Feature weights calculated individually for each session of subject 2 and then averaged. 1 second trial.

IV. Feature Selection

To train the classifiers, only the most significant features for each subject were selected. The most significant features were determined by first manually applying an obvious threshold after observing all feature weights calculated by FSNCA, for each subject's offline data. Next, only the features with weights above that threshold were chosen to build the classifier. This can be seen in Figure 8 where a cutoff of 0.1 was used to select significant features. In future analysis, PCA could be used instead of manually thresholding our features. This would be more valuable since FSNCA led to significant variation in feature weights.

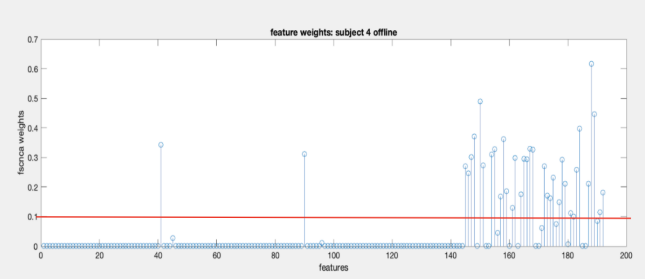

Figure 8. Feature weights for subject 4, found using offline data. Only features with weights above the red line threshold were selected to build the classifier.

V. Classification Models

Four classifiers were used to build on the offline data for each subject: they were linear discriminant analysis (LDA), support vector machines (SVM), XGBoost, and CatBoost. LDA and SVM were used because distinguishing between Error and No-Error classes is a binary classification problem, which LDA and SVM typically generalize well on limited data. In both of these models, there are boundaries created based on the training data that separates the data and allows for future predictions. XGBoost and CatBoost were chosen as well because they are both gradient boosting algorithms designed to be highly efficient and strengthen a model with weak predictions. For XGBoost, the data must first be prepared by scaling numerical features, then it combines the results of many different models, called base learners, to make predictions where each decision node is assigned a real-value score to help categorize the data. Catboost is similar to XGBoost, but features a different implementation of gradient boosting that allows for better classification at the cost of speed. Both of these methods are well documented online for use in Python.

To train the models, each classifier's hypermaraters were tuned using a gridsearch. A stratified K-fold cross validation was used to evaluate the performance of the models during training. The sampling was not randomly shuffled in order to preserve the temporal order of the data, because of the time-varying, non-stationary nature of EEG signals. The mean accuracy of the folds during training was then compared to testing accuracy. Chance level was also calculated for each subject as another indicator of model performance, as indicated by the red lines in the figures comparing training and testing accuracy.

A. Offline Training, S2 & S3 Testing

Each model was trained on the offline data. Mean training accuracy was quite high, yet test accuracy was noticeably lower. It is important to note the significant difference between the testing and training accuracies (Figure 9). Moreover, the test accuracies perform at or around chance level, showing that the test cases of the models did not have particularly good performance. Chance level was calculated by the ratio of "No Error" trials to total trials. Fundamentally this is the accuracy of a classifier that only picks "No Error". The stark difference between test and train accuracies as well as the testing accuracies performing at chance level are indicative of the model overfitting of the data, meaning that it is lacking robustness.

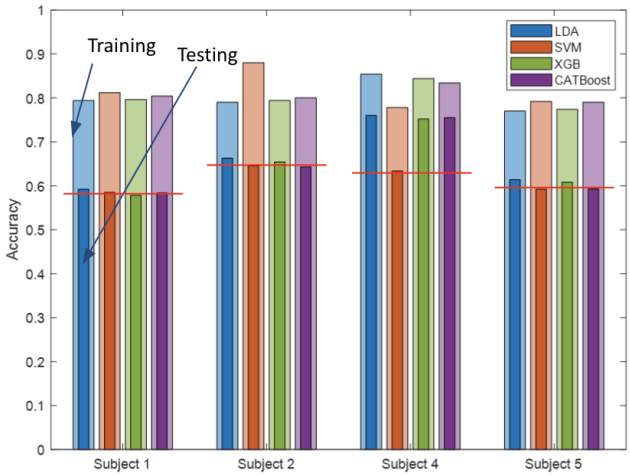

Figure 9. Classification accuracy on all the modified models, training vs testing accuracy, for every subject. The red lines are the chance level.

Another metric we used to evaluate model performance was by observing the receiver operating characteristic (ROC) curves for each classifier. ROC curves are used as graphical illustrations of binary classifiers; the more area under the curve in a 90 degree fashion, the better the performance. By observing Figures 10-13, it can be seen that the effectiveness of the decoders is quite poor, as they are close to being a diagonal line. However, the performance of the classifier for subject 4 is quite above the rest, suggesting that the features used to decode subject 4 error potentials have more discriminability. This could be due to the way features were extracted or the physiology of subject 4, or a combination of the two. The ROC of SVM results in a straight line for every subject–this means that it has the performance equivalent to flipping a coin, which is in agreement with the SVM test accuracies performing at exactly chance (red line). This classifier is picking the no error class all the time. Perhaps one reason for SVM displaying this level of performance is because of the imbalance of class samples. In the dataset, there was a

significantly higher number of No Error samples than Error samples. In other words, the SVM classifier is biased towards no error and chooses that class every time. One potential solution to fix this problem is to increase the training data in the error class to create a more balanced data set the SVM can train on. For example, adding to the error class dataset by repeating the same data could balance the data and punish the model for only picking the "No Error" class.

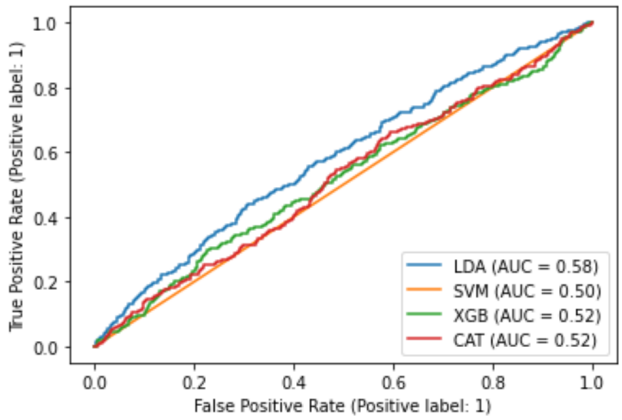

Figure 10. ROC curves of all the decoders for subject 1.

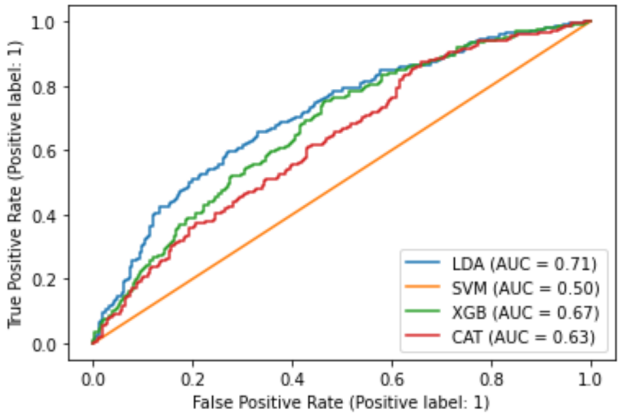

Figure 11. ROC curves of all the decoders for subject 2.

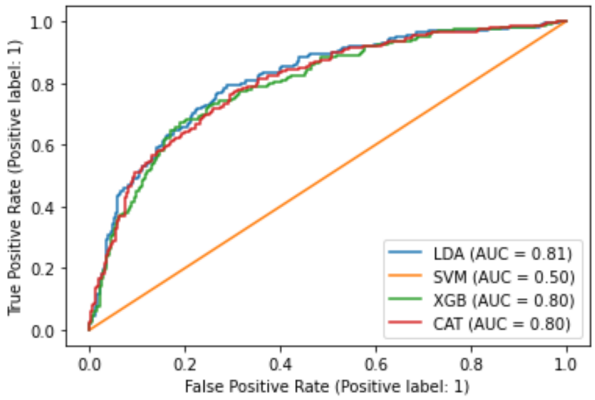

Figure 12. ROC curves of all the decoders for subject 4.

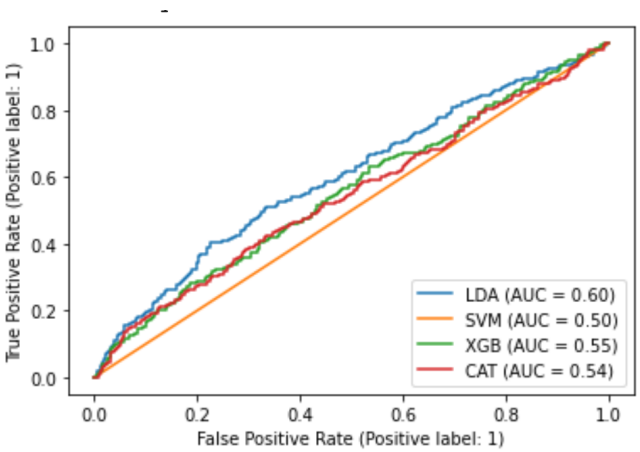

Figure 13. ROC curves of all the decoders for subject 5.

B. Offline & S2 Training, S3 Testing

Using the same features as before to build the decoders, adding session 2 data to training the model noticeably increased decoding performance. The training accuracies performed lower and the testing accuracies performed higher than the models trained strictly on the offline data. This is because the models are attempting to optimize given more data, becoming more robust. In other words, this is a good sign the model is generalizing well and not overfitting, as the test and train converge to similar performances. It is also worth noting that the test accuracies now perform much higher than the chance level, which was not the case when the model only trained on offline data.

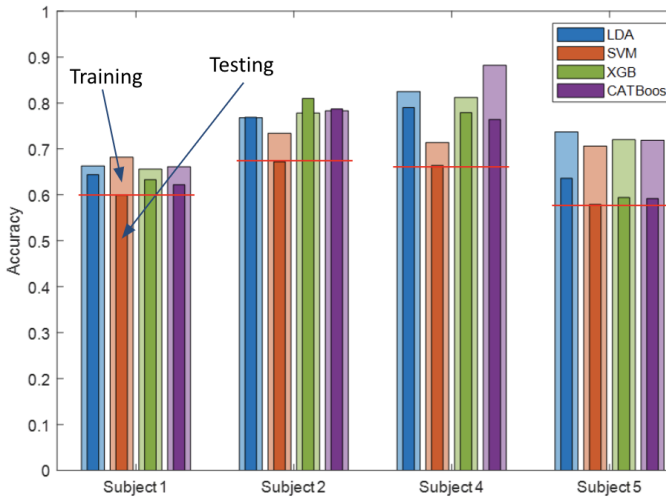

Figure 14. Classification accuracy on all the modified models, training vs testing accuracy, for every subject. The red lines are the chance level.

The ROC curves for each classifier were again used to evaluate model performance. There is a larger area under the curve for each subject, meaning the models are performing better than before. There is an especially large improvement in the performance of the classifiers for subject 2. The increased performance of the classifiers across all subjects is due to the classifiers learning from more data.

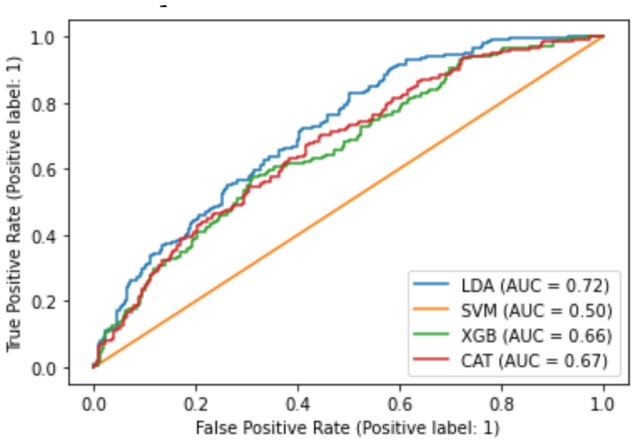

Figure 15. ROC curves of all the decoders for subject 1.

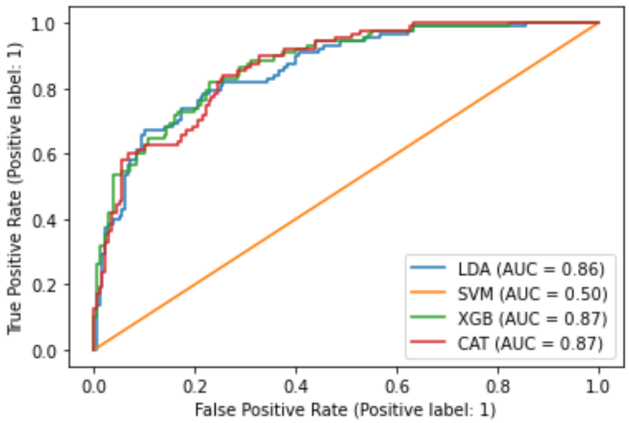

Figure 16. ROC curves of all the decoders for subject 2.

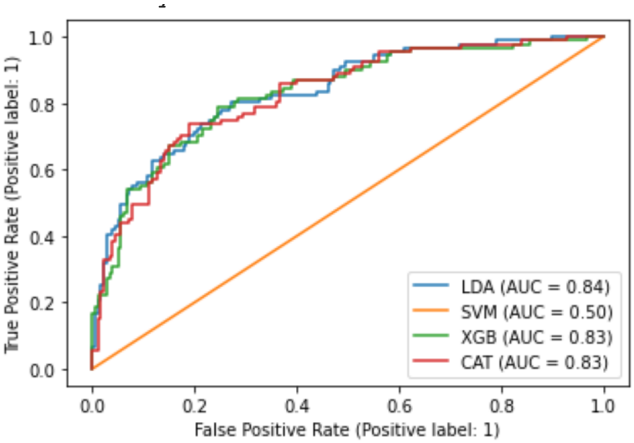

Figure 17. ROC curves of all the decoders for subject 4.

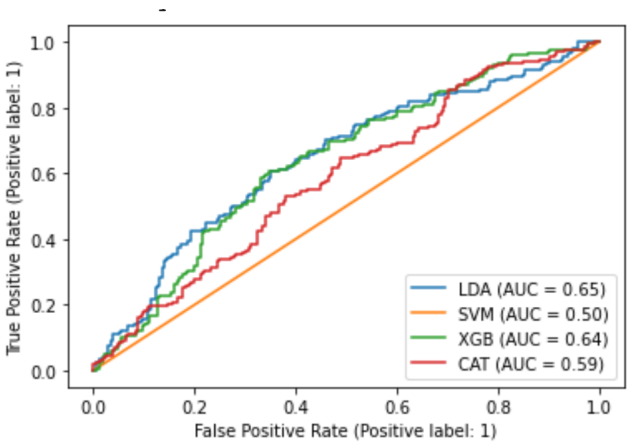

Figure 18. ROC curves of all the decoders for subject 5.

VI. Conclusion

While the grand average plots suggest that some of the features (max, min, TAR) should be highly discriminant, when looking at individual trials, these features are not nearly as predominant (Figure 1). Additionally, low performance may be the use of the

FSNCA, which we do not believe works well with features of varying range. Perhaps normalizing the TAR features would result in more appropriate scoring. Alternatively, Fischer's method would be more consistent. PCA would also be a statistically significant method for feature selection.

Ultimately, the best way to improve the performance of our decoders is to be able to include more data for training. The model accuracy improved significantly, especially in the ROC curves (Fig 15-18) when the offline and session 2 data was used for training rather than just the offline session. Using even more data for training would likely increase the accuracy even more, and is essential for making a BCI personalized for each user

VII. References

[1] C. Başar-Eroglu, E. Başar, T. Demiralp, and M. Schürmann, "P300-response: possible psychophysiological correlates in delta and theta frequency channels. A review," *International Journal of Psychophysiology*, vol. 13, no. 2, pp. 161–179, Sep. 1992, doi: 10.1016/0167-8760(92)90055-G.

[2] Chavarriaga, Ricardo, Iturrate, Iñaki, and Millan, Jose Del R., "Robust, accurate spelling based on error-related potentials", doi: 10.3217/978-3-85125-467-9-15.

[3] V. Abootalebi, M. H. Moradi, and M. A. Khalilzadeh, "A new approach for EEG feature extraction in P300-based lie detection," *Computer Methods and Programs in Biomedicine*, vol. 94, no. 1, pp. 48–57, Apr. 2009, doi: 10.1016/j.cmpb.2008.10.001.

[4] W. Yang, K. Wang, and W. Zuo, "Neighborhood Component Feature Selection for High-Dimensional Data," *JCP*, vol. 7, pp. 161–168, Jan. 2012, doi: 10.4304/jcp.7.1.161-168.