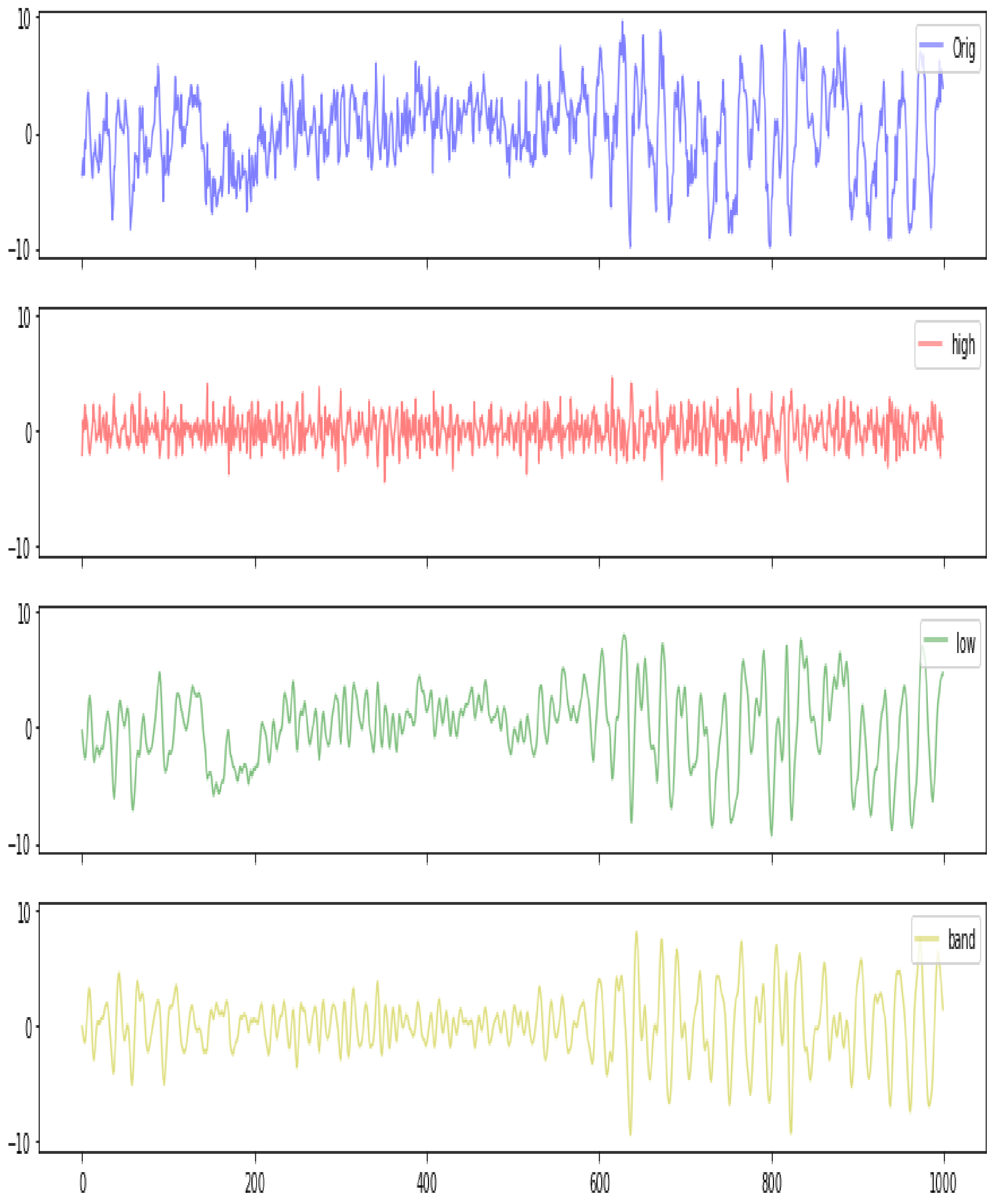
Binary Classification of Motor Imagery Data

John Cooper *jlcooper@ucsd.edu* *A15720005*

Zoe Challacombe *zchallac@ucsd.edu* *A14970527*

COGS 118B Final Project



1 Abstract

In this project we attempt to classify right-left-handed motor imagery signals. While one of our goals is to correctly classify motor imagery signals, the main focus centers on the relationship between methods of feature extraction and increases in correct classifications. We used several methods of signal processing: using power spectral density transforms on epoched datasets, creating 1hz frequency bins, and performing t-tests to determine their statistical relevance of those bins. This gave us a large number of feature matrices that used a variety of naive and novel signal processing and feature extraction algorithms. From here we employed three different supervised models (Decision Trees, Support Vector Machines, and K-nearest neighbors) to examine the signal processing technique that created the most significant change in model accuracy and two unsupervised models to determine the natural patterns of the feature matrices. We originally expected decision trees to underperform both SVM and KNN, and for SVM and KNN to perform similarly. We were correct in these assumptions, with SVMs slightly outperforming KNNs, and DTs being less than optimal. We also expected the PSD feature matrix to produce good results and the naive selection feature matrices to produce bad results. The PSD results were just okay, the naive selection matrices did produce bad results, and wavelet transformed feature matrices performed the best.

2 Introduction

In this project, we work with the UCSD Neural Data Challenge on Kaggle (7-8) to explore classification and feature selection methods on EEG data. The UCSD Neural Data Challenge is a reduced form of a 2008 BCI Competition (7-1, 7-8), and our own analysis is a reduced form of the UCSD Neural Data Challenge, since test labels were not provided. As a result, instead of using the provided training and testing data, we split the provided training data into an $80\% - 20\%$ train-test split.

For this analysis, we draw rough insight and procedure from several sources. First we performed power spectral density transforms. Next, we binned our samples into 1hz intervals based on Dressler et al. (7-7). We decided to take some creative license and perform t-tests on all of our intervals to determine which ones were statistically significant and determine whether pre-filtering our data statistically would have an effect on model validity instead of using the probabilities that were discussed in Dressler et al. (7-7).

The models that we train in this analysis are Support Vector Machines (SVM), K-Nearest Neighbors (KNN), and Decision Trees (DT). Caruana et al (7 -10) illustrates that of the mid-tier models, SVMs and KNNs perform quite well, while DT is shown to be a low-tier model. In this way, this group of classifiers provides something of a benchmark for our model performances across feature matrices. Following this, we applied clustering algorithms to retroactively gauge whether or not natural in-group similarities were present in a select group of features matrices to begin with. Lastly, as discussed in Lee et al. (7-5), we used both continuous and discrete wavelet transforms to perform feature selection on our data.

3 Data and Problem Description

Data was drawn from the UCSD Neural Data Challenge, which provided both raw and epoched EEG data on nine participants over three [1] sessions. Within these sessions 120 trial runs took place, each around four seconds long, where the participant was required to imagine either left or right handed movement (7-1). During these trials, EEG data was recorded through three channels, namely 'C3','C4', and 'Cz'. For each channel, these four second intervals (or epochs) contain a thousand-entry vector corresponding to the EEG signal captured over this time frame (7-1). EOG signals were also recorded through three channels, as were event-start time frames. EOG signals were not necessary in the analysis (7-1) and time stamps were dropped. After removing start times and epoched EOG channel signals from our data, we were left with all epoched EEG signals over all channels, participants, and trials. The original epoched data consisted of 3680 samples and 9 features: Patient ID, Start Time, Event Type, C3, Cz, C4, EOG CH 1, EOG CH 2, and EOG CH 3. The data we actually work with consists of 3680 samples and 3 features: C3, Cz, C4 (before signal processing and feature selection).

4 Methods

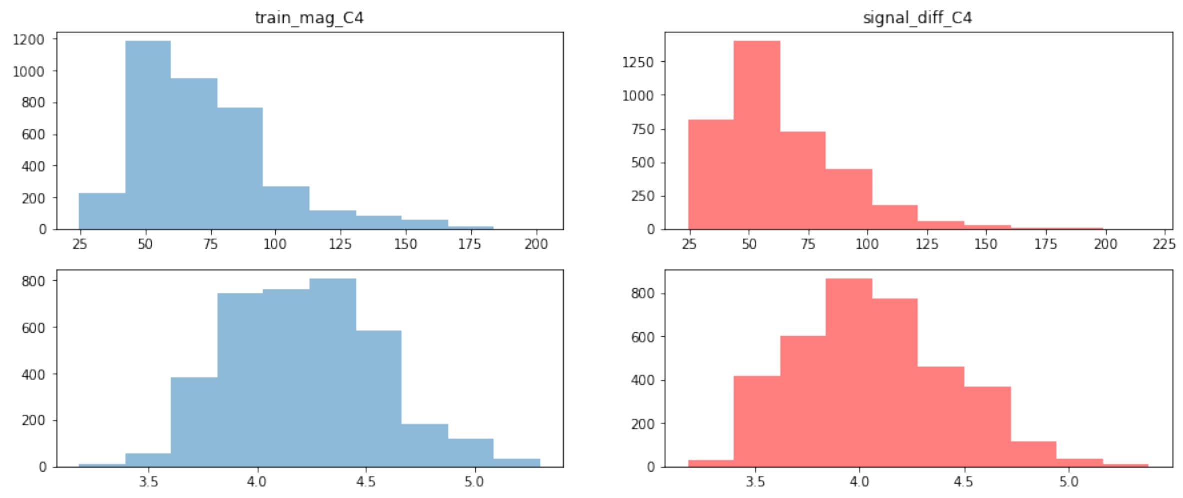
4.1 Signal Processing and Feature Extraction

The data set provided had already been band-pass filtered at at 0.5-100hz (7-1). We opted to process the epoched signals in five different ways.

1. **Naive Processing - Log Signal Magnitudes (Mag)**: The magnitude of each channel trial signal (length 1000) was taken and transformed with natural logarithm, yielding a feature matrix.

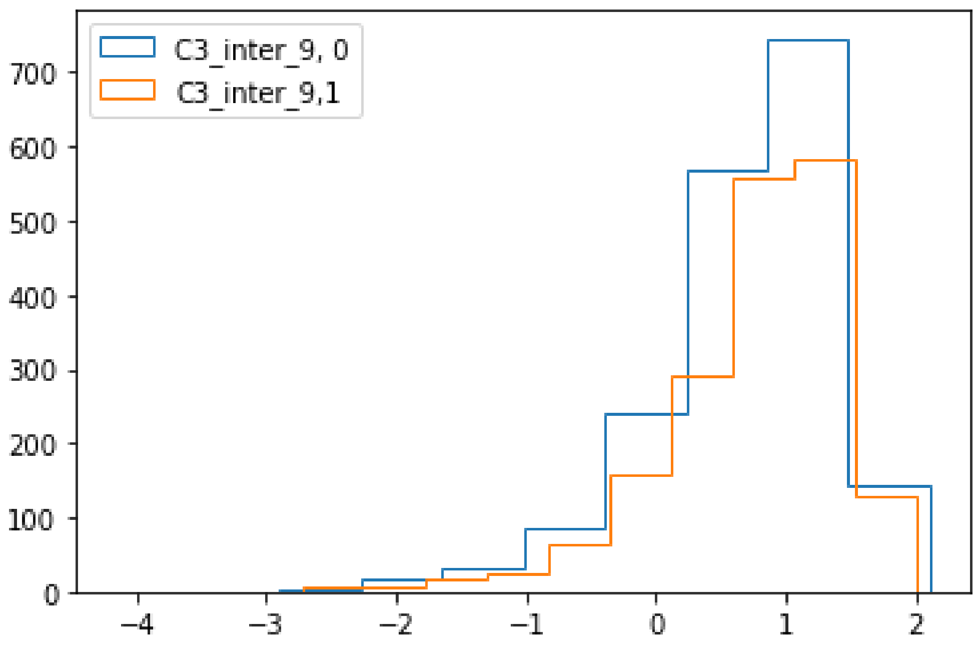
2. **Naive Processing - Log Signal-Difference Magnitudes (DiffMag)**: First, three differences were taken between the channels (columns), namely 'C3-C4', 'C3-Cz', and 'Cz - C4'. Next, the magnitude of each channel trial signal was taken and transformed with natural logarithm, yielding a feature matrix. No other differences were necessary due to symmetry. Note that since we applied StandardScaler() to our data before training, log transforming the data helped with this normalization process, despite reducing variance. Below we see the effects of transformation on two channel entries for different feature methods.

[1]The original data actually consisted of **five** sessions (7-8). Because of the contest-format, labels were not provided for the last two sessions, and so we were forced to only use the first three.

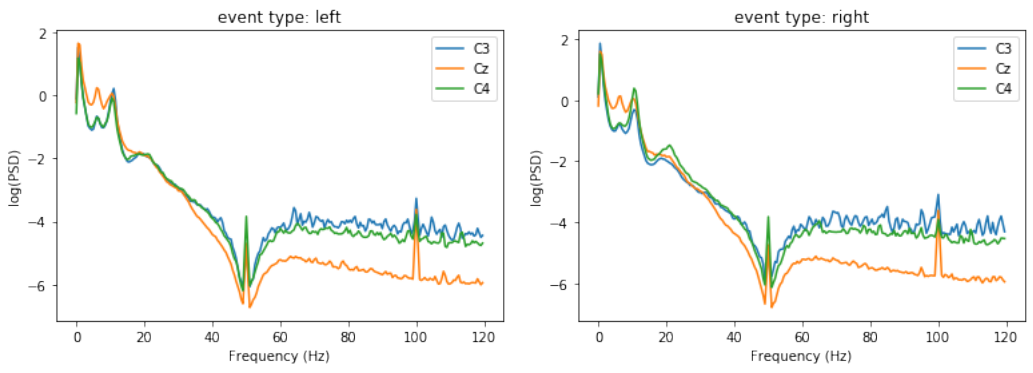


3. **Novel Processing - 1hz PSDs**: The idea to work with mean power spectral densities for their predictive power came from Dressler et al (7 -7). While we follow the 1 hz decomposition proposed in that paper, we do not manufacture these intervals into probabilities (7-7). We instead use hypothesis testing.

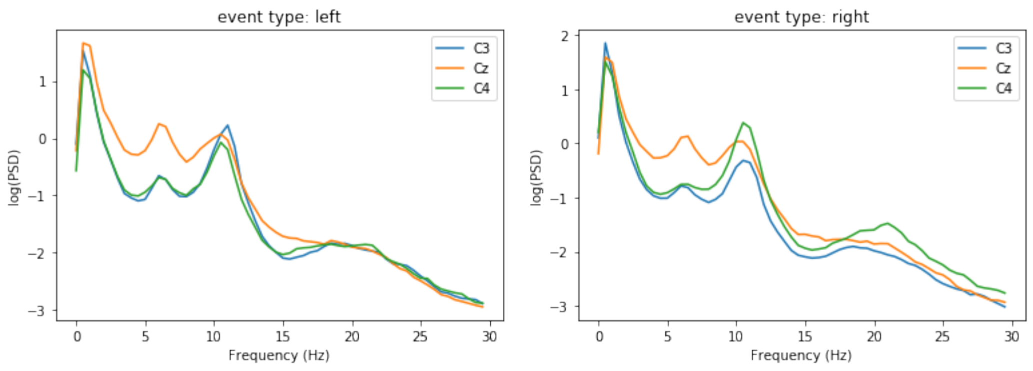
   First, the mean power spectral densities (MPSD) of each channel trial signal were taken. Next, under each channel, these MSPDs were decomposed into 1hz intervals. For example, a single epoched signal under 'C3' transformed into its MSPD would yield a vector representing the mean power of the signal at each half-hz, spanning 125 hz total. After applying this process to every entry entry in the original feature matrix, we obtain a N-Samples x (3-Channel x 125 1hz intervals) matrix. Following this, the entire data set was split by its classification and paired t-tests were performed on matching columns to determine whether or not the means of these 1hz intervals were significantly different between oppositely-classified trials. Following this, the negligible 1hz columns were removed from the features matrix, yielding a final features matrix. The following is an example of one of the 1hz sub-channels that was kept for its statistical significance. Although C3Inter9 has a slightly skewed distribution, these PSD intervals follow an approximately normal trend which permitted us to peform t-testing.



   Here, we show pictures of the PSD transforms scaled to 0-120 hz.



If we zoom in on a specific region, say 0-30hz, we can see some evident differences between the two.



The evident differences in peaks and troughs over certain intervals is why we chose to test for statistical significance in magnitude.

4. **Guided Processing - Wavelets 1 (Wavelet1)**: In Lee, et al., wavelet transforms and the feature vectors based on those wavelets are used to increase the accuracy of feature selection. In our project, we hoped that introducing wavelet transforms to increase the impact of our feature selection before running an SVM would be a way to increase the accuracy of our model without adding more data or increasing the time it takes to run. Following the example mentioned above, we created two separate transforms using pywavelets, one continuous (7-3), and one discrete (7-4).

   Like in the other signal processing methods, this transform is applied to each channel trial signal, i.e each epoched entry of the original matrix. For the continuous wavelet transformation (CWT), we roughly follow the procedure set out by Lee et al. First, we transform all epoched signals using a Morlet wavelet on a scale ranging from 6hz to 30hz (7-4). This process is repeated for all three channels. Next, we apply the same process, but with a discrete wavelet transformation (DWT) using the daubechies wavelet 4. Finally, the two matrices are concatenated to provide a feature matrix consisting of features from both the CWT and DWT (7-4).

5. **Guided Processing - Wavelets 2 (Wavelet2)**: The process in Wavelets 1 is repeated, but with a daubechies wavelet 4 which uses a five level decomposition. Note that in Lee et al, it seems a 24-level decomposition is used (7-4).

4.2 Models and Model Selection

We chose these models for several reasons, with the hope that because none of them are the most successful modern machine learning algorithms, according to Caruna and Niculescu-Mizil, changes in feature inputs would have a more significant effect on the outcomes.

For every model discussed below, except unsupervised methods, we also standardised the data using scikit-learn's (7-5) StandardScalar method. We ran three trials of five fold grid search cross validation to identify and select the best parameters on each data set, for each model.

1. **SVM**: Although in many ways a basic machine learning model, Support vector machines are actually highly robust, performing well on many different types of data (7-10) while still being easy to implement and train. Additionally, as they are used in conjunction with wavelet transforms to increase accuracy in Lee et al. (7-5), it was extremely important to maintain a standard model that would accurately describe differences in accuracy given the array of methods we used for feature selection.

   Parameters: Kernels used were radial basis function (rbf) and polynomial kernels. For rbf kernel, gamma took values in $\{0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1, 2\}$. Polynomial degrees took values in $\{2, 3\}$. The regularization parameter for both kernels took values between 10^{-6} and 10^{6}.

2. **KNN**: K nearest neighbors is a fairly standard algorithm which performs similarly to SVM (Caruana, 7-10). We decided to use KNN because it maintains the context of the samples within the feature space.

   Parameters: The number of neighbors varied from 1 to 200 in steps of 10. Weights took values in $\{uniform, distance\}$.

3. **DT**: Standard decision trees, are on the whole, generally not as good as many other models. So, in using a decision tree classifier with no bagging or boosting, we have a model that will likely under-perform, giving us a range of supervised techniques and their potential outcomes.

   Parameters: Criterion took values in $\{gini, entropy\}$. Max depth took values between 1 and 8, min leaf sample threshold took values between 1 and 5, and min split threshold took values between 2 and 7, with each sequence spaced by 1 step.

4. **Unsupervised Methods** We also used Kmeans and spectral clustering to see where the border between right and left hand signals naturally falls, if it exists. We used the Fowlkes Mallows Score to determine the similarity between our test cluster and our "ideal" cluster of the ground truth labels.

   Parameters: Parameters were not searched here, as this was meant to be more of a heuristic. For Spectral Clustering, gamma was taken to be uniformly 1. For KMeans, tolerance was set to be uniformly .0001.

5 Results

We ran three trials over gridsearch for each feature-matrix-classifier pair. This resulted in three different scores for each feature matrix, per classifier. Each entry within the chart represents the mean accuracy of the classifier over three tests. In this chart "refiltered" refers to us mistakenly band-pass refiltering the epoched signals over 7-30 hz. We also did not run KMEANS or Spectral Clustering on feature matrices other than Wavelet1 and Wavelet2, since we were inclined to believe that these would have the most in-built separation

- also, time considerations and long running times. Given that our dataset was considerably balanced (7-8), we decided to gauge all classifiers on accuracy. Finally, as noted before, all feature matrices were log transformed and scaled before training.

Table 1: Performance of each feature matrix on classifiers

	SVM	KNN	DT	MEAN
Mag(final)	0.576	0.554	0.534	0.555
DiffMag(final)	0.561	0.549	0.551	0.554
PSD1hz(final)	0.659	0.649	0.574	0.627
Wavelet1(final)	0.666	0.652	0.625	0.648
Wavelet2(final)	**0.702**	**0.664**	**0.643**	**0.669**
Mag(refiltered)	0.642	0.643	0.623	0.636
DiffMag(refiltered)	0.592	0.597	0.572	0.587
PSD1hz(refiltered)	0.658	0.634	0.575	0.622
Wavelet1(refiltered)	0.663	0.643	0.615	0.640
Wavelet2(refiltered)	0.679	0.649	0.618	0.649

Table 2: Best Performance

	Best Overall Performance
Classifier	SVM
Classifier Features	C = 10, Gamma = 0.1, Kernel = RBF
Feature	Wavelet2
MeanAcc	0.702

Table 3: Clustering results for Wavelets (Fowlkes Mallows Score)

	KMEANS	SPECTRAL
Wavelet1	.515	0.705
Wavelet2	0.503	0.542

6 Discussion-Conclusion

Even though our accuracies weren't as high as we had hoped, we did get some promising results. Since the wavelet2-SVM pair performed the best across all classifiers, we know that our discrete and continuous wavelet transforms were at least somewhat correct. Furthermore, SVM usually outperformed KNN except when spatial relationships were the focus of the feature transform (magnitude and the magnitude difference). We could be reading far too much into that however, and it could be that those transforms were just bad.

One large analysis concern is that our clusters did not have the scores we had expected. Due to the nature of the wavelet transform, the relationships between points on the mesh should not have changed very much; therefore, if the original clusters were distinct, we would expect to see distinct clusters after the transform. This was not the case and the clustering scores are lower than we had hoped. This either means that our wavelets are wrong, some of our feature selection methods are wrong, or the data did not cluster distinctly from the beginning.

ISSUES:

1. When first attempting to implement the power spectral density transform, we used 3 intervals {(0,7), (8,31),(32,125)} per channel as our features. In retrospect, these features were too wide and the transformed feature were similar enough that we got an accuracy of barely 58% on the best classifier. This also lead to extremely long train times for the SVM as there was likely not enough feature differentiation to lead to convergence, even with an iteration maximum of 1,000,000.

2. We originally performed a 7-30hz bandpass filter on the dataset, which had already been filtered. This resulted in the filtering out of important frequencies and interfered with feature selection, especially when attempting to perform t-tests on the 1hz PSDs. When we rectified this, we got better results for the the majority of trials with models trained on 1 hz features, but surprisingly, this dropped the scores for the models trained on large feature intervals by almost 5%, from low 60's to mid 50's. This also meant that we had to completely redo all of our analysis to get the right data.

3. There were many potential issues with our use of wavelet transforms, as this was our first experience working with them. We did our best, but it's likely that our selection of parameters and implementation of the transforms wasn't optimal.

4. In addition to testing for mean in PSD 1hz, it would have been valuable to also use variance as a means of further differentiating class PSD signals to increase the robustness of the transform.

5. We ran a few trials on statistically significant feature matrices and matrices with both significant and insignificant features, and the one with insignificant features was slightly higher, but not significantly ($< 1\%$), but the significant feature only matrices trained more than 30% faster. However, because we didn't test this method rigorously, we cannot conclude that using a t-test is a valuable method of feature selection.

6. In Lee et al, many statistics were computed from the multi-level DWT transforms (Lee et al), which we did not use. Instead, we simply took the magnitude and log transformed the DWT transforms. Granted that we had followed this protocol, we would have expected to see better performance. These missing statistics (taken verbatim) include:

   (a) "Mean of the absolute values of the wavelet coefficients in each sub-band"

   (b) "Average power of the wavelet coefficients in each sub-band"

   (c) "Standard deviation of the wavelet coefficients in each sub-band"

   (d) "Ratio of the absolute mean values of adjacent sub-bands"

   (e) "Energy of the wavelet coefficients in each sub-band"

   (f) "Entropy of the wavelet coefficients in each sub-band"

   (g) "Skewness of the wavelet coefficients in each sub-band"

   (h) "Kurtosis of the wavelet coefficients in each sub-band"

7. There are some potential calibration issues here, given that we want to assess both classifier performance as well as the effect which feature-extraction has on classifier performance. A question which arises, and which we did not discuss, is whether or not it is even correct to gauge feature-extraction performance across classifiers. Intuitively, it is clear that if a features matrix is contributing uniformly to better performance across classifiers, then it underwent the better feature-extraction method. But is there a more precise way to gauge this? There is a clear extension into testing features-classifier pairs over several data sets, one which we wish we could have addressed had there been more time.

7 Bibliography

1. Brunner, C., Leeb, R., Muller-Putz, G., Schlogl, A. and Pfurtscheller, G., 2008. BCI Competition 2008 – Graz Data Set A. [online] Bbci.de. Available at: `http://www.bbci.de/competition/iv/desc_2a.pdf`

2. Scikit-learn.org. 2020. Scikit-Learn: Machine Learning In Python — Scikit-Learn 0.23.2 Documentation. [online] Available at: `https://scikit-learn.org/stable/index.html` [Accessed 16 December 2020].

3. Wasilewski, F., 2020. Continuous Wavelet Transform (CWT) — Pywavelets Documentation. [online] Pywavelets.readthedocs.io. Available at: `https://pywavelets.readthedocs.io/en/latest/ref/cwt.html`[Accessed 16 December 2020].

4. Wasilewski, F., 2020. Discrete Wavelet Transform (DWT) — Pywavelets Documentation. [online] Pywavelets.readthedocs.io. Available at: `https://pywavelets.readthedocs.io/en/latest/ref/dwt-discrete-wavelet-transform.html` [Accessed 16 December 2020].

5. Lee, D., Park, S. and Lee, S., 2020. Improving The Accuracy And Training Speed Of Motor Imagery Brain–Computer Interfaces Using Wavelet-Based Combined Feature Vectors And Gaussian Mixture Model-Supervectors. [online] Available at: `https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5677306/`

6. Ng, W., Saidatul, A. and Chong, Y., 2019. PSD-Based Features Extraction For EEG Signal During Typing Task. [online] Available at: `https://www.researchgate.net/publication/334097808_PSD-Based_Features_Extraction_For_EEG_Signal_During_Typing_Task`

7. Dressler, O., Schneider, G., Stockmanns, G. and Kochs, E., 2004. Awareness And The EEG Power Spectrum: Analysis Of Frequencies. [online] Available at: `https://academic.oup.com/bja/article/93/6/806/266623` [Accessed 16 December 2020].

8. Kaggle.com. 2020. UCSD Neural Data Challenge — Kaggle. [online] Available at: `https://www.kaggle.com/c/ucsd-neural-data-challenge/overview` [Accessed 16 December 2020].

9. `https://www.researchgate.net/figure/Example-of-a-one-minute-segment-of-raw-EEG-data`
`fig2_307912094`

10. Caruana, R. and Niculescu-Mizil, A., 2006. An Empirical Comparison Of Supervised Learning Algorithms. [online] Cs.cornell.edu. Available at: `https://www.cs.cornell.edu/~caruana/ctp/ct.papers/caruana.icml06.pdf`[Accessed 16 December 2020].