# RIMS: A finger flexion ECoG decoder

Nathaniel Nyema, Ola Owoputi

May 6, 2020

## Overview


**Figure 1** Overview of the algorithm's training and testing process.

RIMS is a collection of random forest regression models which individually predict DataGlove flexion angles for specific fingers for a given patient based ECoG data. The model uses a number of per-channel ECoG features including band power, energy, and line length, along with judicious usage of feature selection. By training on fingers 1, 2, 3, and 5, we achieved a hidden test set correlation of 55%.

## The Algorithm

We train the models by first re-referencing the signal at each electrode using a common average reference (CAR) montage. This involves subtracting the mean voltage across electrodes at a given time point from the voltage at each electrode at that time point[1]. We then use an FIR band pass filter with cutoffs 0.15 Hz to 200 Hz and order 100. We then extract the energy, signal mean, line length, kurtosis and power in the frequency bands 8–12 Hz, 18– 24 Hz, 75–115 Hz, 125–159 Hz, and 159–175 Hz[2] for each electrode in windows

[1] Kamrunnahar et al, 2009
[2] Schalk et al, 2007

of length 80 ms with 40ms overlap across segments. These values are chosen to match the 25 Hz sampling rate of the DataGlove data as done by team S-2 in the original competition held by Schalk[3]. We extract features across windows in parallel by first reformatting the filtered data into a 3d array of dimensions [samples x channels x windows]. We then compute the stft along the first dimension of the array and use the result to calculate the specified bandpowers, which we then reformat into a 2d array of size [samples x channels * # bands]. The other features are similarly calculated in parallel along the first dimension of the time windows array and reformatted into a 2d array of size [samples x channels * # other features]. These arrays are finally concatenated into our final feature matrix.

We then use the Matlab function `fscmrmr()` to perform forward selection for each pairing of patient and finger with the minimum redundancy maximum relevance (MRMR) algorithm. The MRMR algorithm computes relevance and redundancy based on mutual information and ranks the features accordingly. We save the sorted indices of features for later use and take the first 200 sorted features for each patient-finger pairing. We then take a moving mean along the first dimension (the samples) of these downsized feature matrices with a window size of 40 samples to arrive at our final processed feature matrices. We then use these feature matrices to incrementally train random forests of 512 regression trees for each patient-finger pair using Matlab's `TreeBagger()` function.

Finally, when predicting, we first extract features the same as aforementioned, then use the saved feature selections to select the first 200 sorted features for each patient-finger. We then smooth the feature matrices across samples with a 40 sample moving mean and use the predict method of the TreeBagger class to make predictions with the appropriate model for each patient-finger.

# Developing the Model

When working on RIMS we were developing 2 separate algorithms in parallel with the intent of using the better of the two. Both algorithms used mostly the same set of features; the main difference was that one performed regression with a support vector machine (SVM) while the other used this random forest approach. The random forest regression model was the winner mostly because it took less time to train and was thus easier to prototype with. We also generally got better results with the random forest models when testing on the leaderboard. This likely had mostly to do with the larger window sizes we were using when training the SVM (window size/overlap) and the excess of features in the training data. The major

[3] Tangermann et al, 2012

issue was that the time complexity of training a nonlinear SVM is between quadratic and cubic with respect to the number of training examples[4], such that increasing the number of samples in the training set by decreasing the window size would have made training take an unreasonable amount of time. Linear SVMs, though easier to train, were unable to achieve results that were significantly better than the linear regression baseline. Furthermore, the ensemble structure of the random forest made it less prone to overfitting than the SVM, leading to more consistency between cross-validation performance and test set performance.

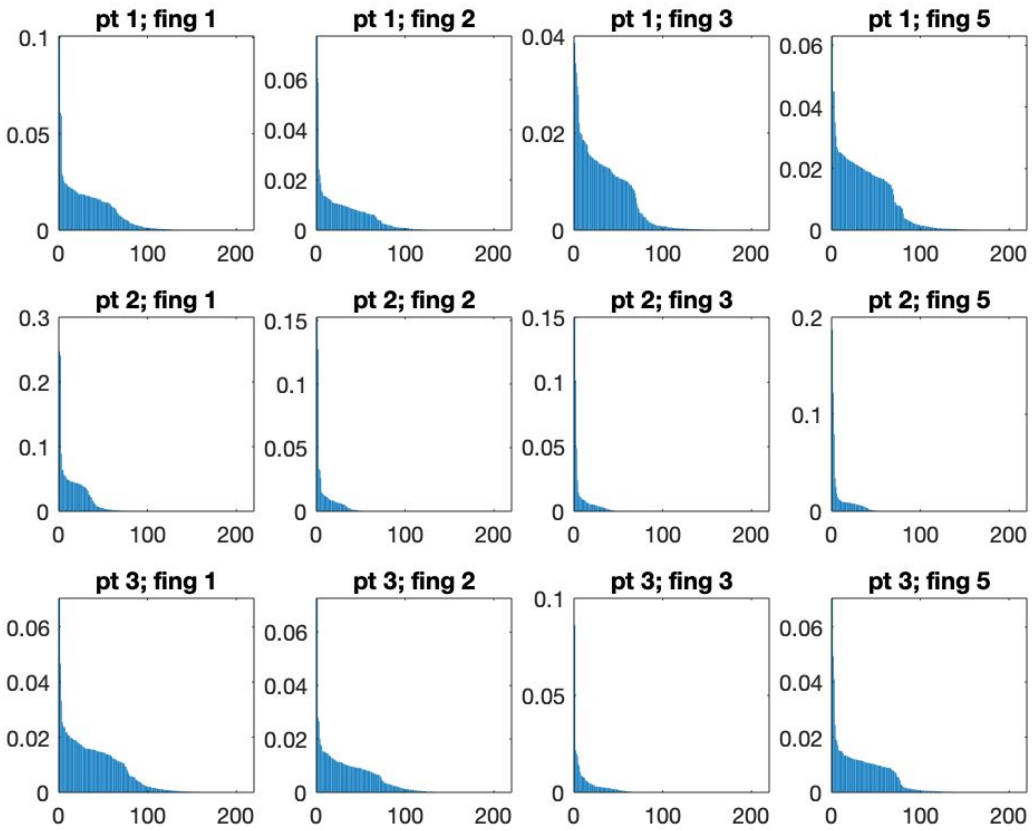


**Figure 2** Bar plots of the first 200 sorted feature prediction scores for each patient finger pairing excluding finger 4. As can be seen, all pairings a majority of the predictive power lies within these first 200 features.

The random forest on the other hand showed promising results from the first round of testing. We began with the same features described in the optimal linear decoder algorithm and trained a random forest of 64 trees with the R matrix computed with N-3 time windows and were already at a correlation of 0.42 with the leaderboard data. From here we mostly focused our efforts on feature engineering. After further

[4] List & Simon, 2009

separating the training data into a training set and a testing set and implementing the same approach except with 8 trees, one thing we noticed in the predictions on the testing data was that there appeared to be excess activity in the regions that should have corresponded to non-movement. Furthermore, there was generally quite a lot of noise in the output signal. We figured there was a possibility that including the N-3 time steps in our feature matrix was influencing both of these observations and thus decided to remove them. Furthermore we figured to clean up the noise, and also further help with training time we could do some feature selection, which we did with the MRMR algorithm. Since we were already doing feature selection we also added some potentially informative features; namely we added energy, line length, and kurtosis (a measure of the shape of the probability distribution of the ECoG values).

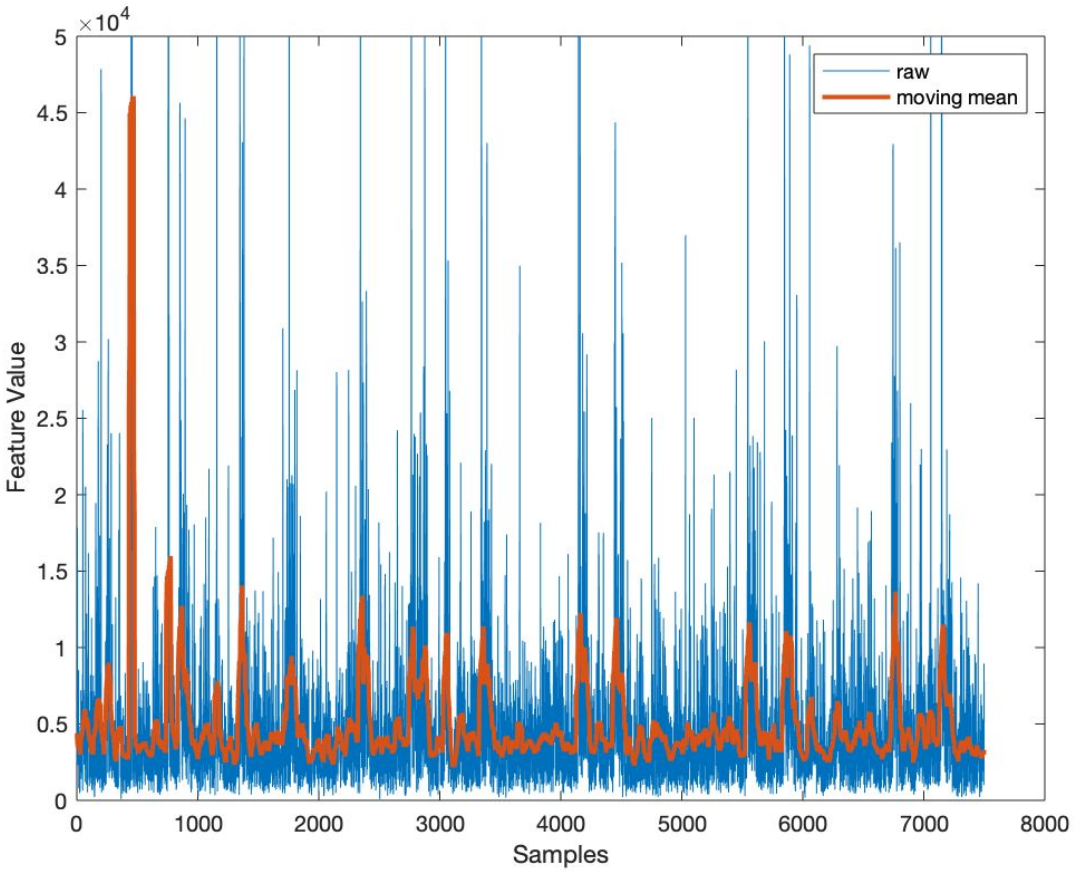


**Figure 3** The value of the most predictive feature for patient 1 finger 2 is plotted here both pre and post moving mean. The moving mean evidently removes most of the noise in the signal making it better resemble the frequency content of the expected predictions.

After making these changes we decided to plot some of the selected features over time and noticed the same noise we observed in the prior predictions. This made us realize that what was more likely to be

influencing the noise in our output signal was the noise in our features. As a result, we decided to smooth the features by taking a moving mean within features and across samples. We tested a few window sizes and arrived at 40 samples. These changes brought our correlation with the leaderboard data up to the 0.5 range with 64 trees trained with all training data. From this point on we just kept adding trees until the correlation of the out of bag predictions with the actual finger angles in the training set began to plateau, which occurred at 94.9% correlation with 512 trees.

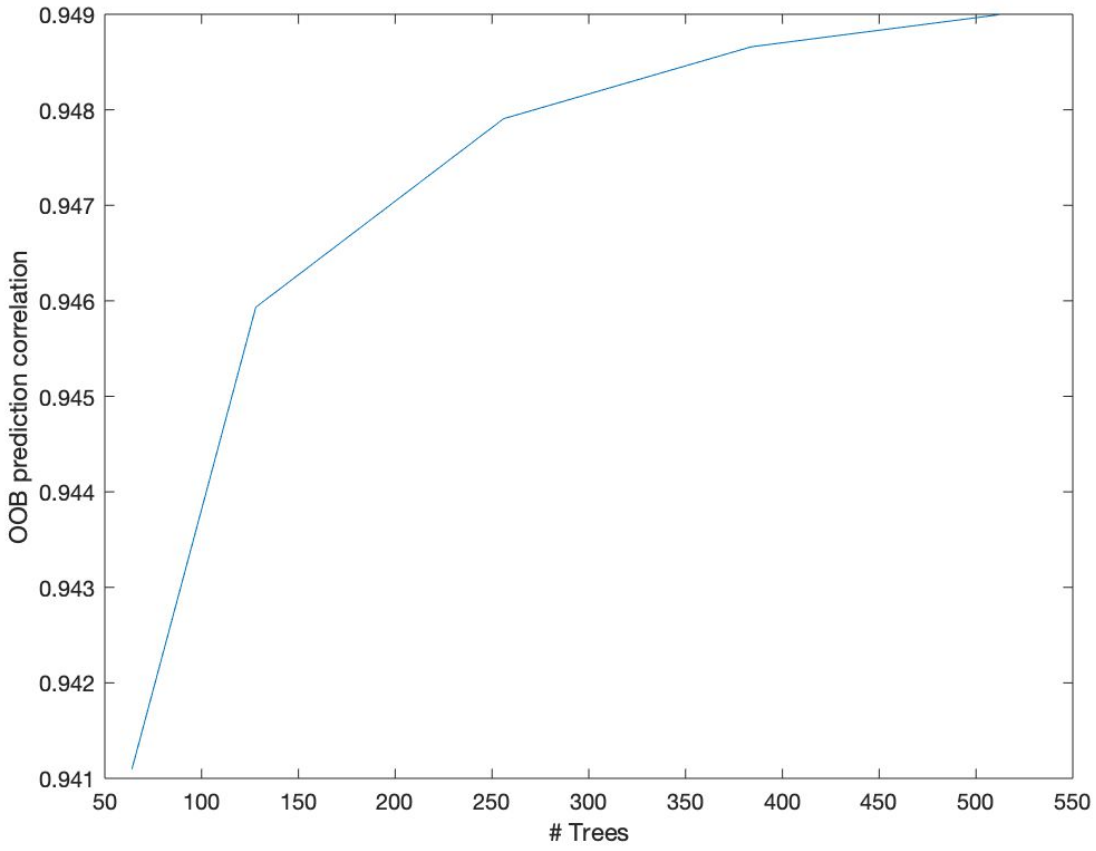


**Figure 4** A plot of the out of bag predictions' correlation with the finger movements in the training data. Adding trees increases the correlation (admittedly moderately) until it begins to plateau. Theoretically more trees will continue to improve performance but beyond this point these improvements would be marginal.

One thing worth commenting on is the fact that throughout this process we never used finger 4 for any patient. This is because as stated in the project description, finger 4 angles were found to correlate with neighboring finger movements. Intuitively, this makes a lot of sense as finger 4 corresponds to a person's ring finger which is in fact quite difficult for most people to move independently of neighboring fingers.

From research we found that this is because, as we understand it, the ring finger is enervated by both the radial and ulnar nerves, the first of which is primarily dedicated to controlling fingers 1 through 3 and the latter finger 4-5[5]. As a result one may intend to move solely their ring finger but the neural circuitry on the level of the hand restricts the ability to do this without pulling other fingers along.

# Conclusion

The open-ended nature of this project allowed us to use a wide array of theory and techniques that we had learned throughout the semester as well as research new strategies. Overall we were pleased with the performance of the model given the relatively small amount of training and testing data. However, we are interested to see if, given enough data, we could utilize more advanced methods in deep learning to improve performance. Such architectures have been successfully applied in similar signal processing applications such as audio processing[6] and could likely match or exceed state-of-the-art performance in ECoG problems such as this.

[5] Rapp et al, 2020
[6] Stöter et al, 2018

# References


1. M. Kamrunnahar, N. S. Dias and S. J. Schiff, "Optimization of electrode channels in brain computer interfaces," *2009 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, Minneapolis, MN, 2009, pp. 6477-6480, doi: 10.1109/IEMBS.2009.5333585.

2. G. Schalk, J. Kubanek, K. J. Miller, N. R. Anderson, E.C. Leuthardt, J.G. Ojemann, D. Limbrick, D.W. Moran, L.A. Gerhardt, and J.R. Wolpaw, “Decoding Two-Dimensional Movement Trajectories Using Electrocorticographic Signals in Humans”, *J Neural Eng*, vol. 4, pp. 264-275, 2007.

3. Tangermann, Michael, et al. “Review of the BCI Competition IV.” *Frontiers in Neuroscience*, vol. 6, 2012, doi:10.3389/fnins.2012.00055.

4. N. List, and H. U. Simon, "SVM-optimization and steepest-descent line search," in *Proceedings of the 22nd Annual Conference on Computational Learning Theory*, 2009.

5. Rapp FA, Soos MP. Anatomy, Shoulder and Upper Limb, Hand Cutaneous Innervation. [Updated 2019 Jun 21]. In: StatPearls [Internet]. Treasure Island (FL): StatPearls Publishing; 2020 Jan-.

6. F. Stöter, A. Liutkus, and N. Ito, “The 2018 signal separation evaluation campaign,” in *Latent Variable Analysis and Signal Separation: 14thInternational Conference, LVA/ICA 2018, Surrey, UK*, 2018, pp. 293–305.

# Appendix

## Contents

- train_model.m: script for training
- make_predictions.m : script for making predictions
- getWindowedFeats.m : Split raw ECoG into windows and compute features
    - filter_data.m : filters the raw ECoG
    - get_features.m : computes time-domain features
    - get_bandpowers.m : computes time-frequency features
- zointerp.m : upsamples output to match the raw ECoG