Seizure Prediction Using Short-Term and Long-Term Features

Brian W. Lang, Daniel Lavery, Kelly Roman,
Scott Dobson, Derek Broadhead

December 2016

Location: Areté Associates, Arlington, Virginia, USA
Email: {blang, dlavery, kroman, sdobson, dbroadhead}@arete.com
Competition: Melbourne University AES-MathWorks-NIH Seizure Prediction Challenge

1 Summary

The final model utilized extremely randomized trees[1] on both long (entire 10-minute file) and short (minute based) term generated features, about a total of 2200 in all. It was trained and tuned, not at all exhaustively but rather crudely, for each patient. It should be noted that deep-learning techniques (both CNNs and LSTMs) were investigated but quickly abandoned. Due to the limited data-set, the best result using such techniques was only $\sim$0.62.

2 Feature Selection

Long term features included those easily implemented from the literature[2], in particular correlation coefficients between channels and spectral and temporal statistics on a down-sampled (by a factor of 5) per channel basis.

To be complete, the long terms features included: Correlation coefficients and eigenvalues for all 16 channels at full temporal resolution. Next, on a per-channel factor of 5 reduced temporal resolution the following features were obtained: standard deviation, kurtosis, skew, number of zero crossings, complexity and mobility as well at the 1^{st} and 2^{nd} derivative standard deviations and number of zero crossing. The features, on a per channel basis, in the spectral domain included: Maximum frequency, total summed energy, entropy and normalized summed energy in bands [0.1, 4, 6, 12, 30, 40]Hz. Total number of features were: 120 for correlation coefficients; 16 for eigenvalues and 16*(19), where 19 is the number of features per channel, for a total of 440. Named variables for long-term features are show in Table 1:

The short term, at full temporal resolution, features were normalized summed energy in the standard spectral bands derived on a minute-by-minute basis with 30s of overlap.

To be complete, the summed energy bands consisted of [0.1, 4, 8, 12, 30, 70, 180] Hz. Named variables for short-term features took the form: BandEnergy_i_j_k where i is temporal segment; j is channel; k is band number.Total number of features were (19*16*6) = 1824 where 19 is the number of 1 minute segments per file, 16 is the number of channels and 6 is the number of bands tabulated.

For both long and short term feature extraction the data was whitened (i.e. mean-zero and standard deviation of unity) per channel. Additionally, all power spectrum features used the Welch method so as to help beat down the noise. Feature breakdown and top 20 importances are showing in Figures 1,2 and 3.

Var. Name	Descrpition
coef{i}	Correlation Coefficients
coef_timeEig{i}	Eigenvalue of the Correlation Matrix
sigma{j}	Standard Deviation
kurt{j}	Kurtosis
skew{j}	Skewness
zero{j}	Number of Zero crossings
sigmad1{j}	Standard Deviation of the 1^{st} derivative
sigmad2{j}	Standard Deviation of the 2^{nd} derivative
zerod1{j}	Number of Zero crossings of the 1^{st} derivative
zerod2{j}	Number of Zero crossings of the 2^{nd} derivative
maxF{j}	Maximum Frequency
RMS{j}	Root-Mean-Square
SumEnergy{j}	Total Energy in the spectrum
entropy{j}	Entropy
Mobility{j}	Mobility
Complexity{j}	Complexity
BandEnergy{j}{c}	Total normalized energy per band

Table 1: Table of Long-term derived features, where {i} is the coef number, {j} is the channel number and {c} the band number

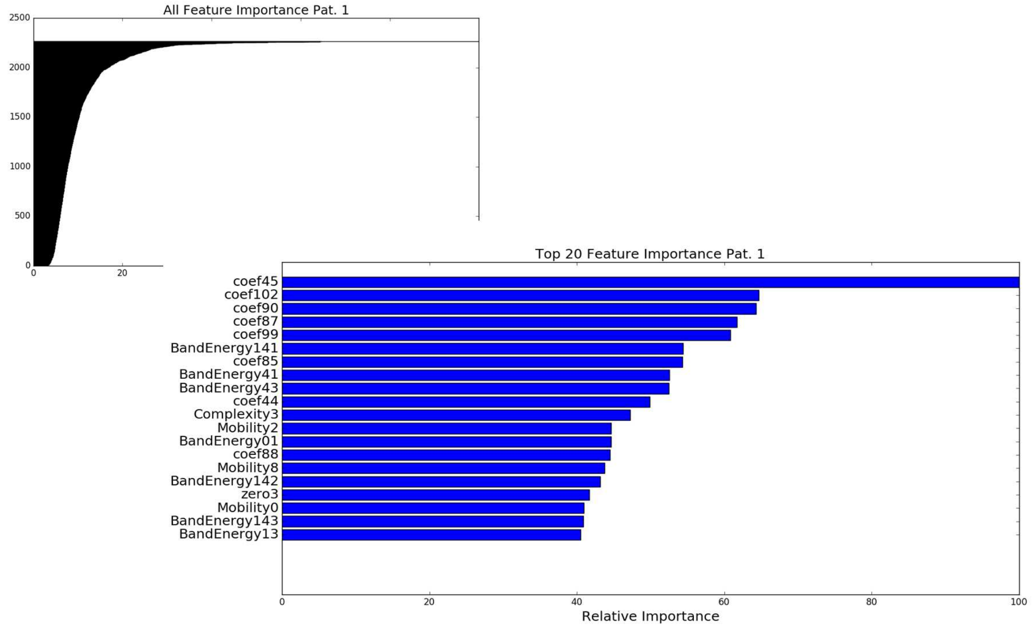

Figure 1: Feature space for patient 1 including the top twenty feature breakdown.

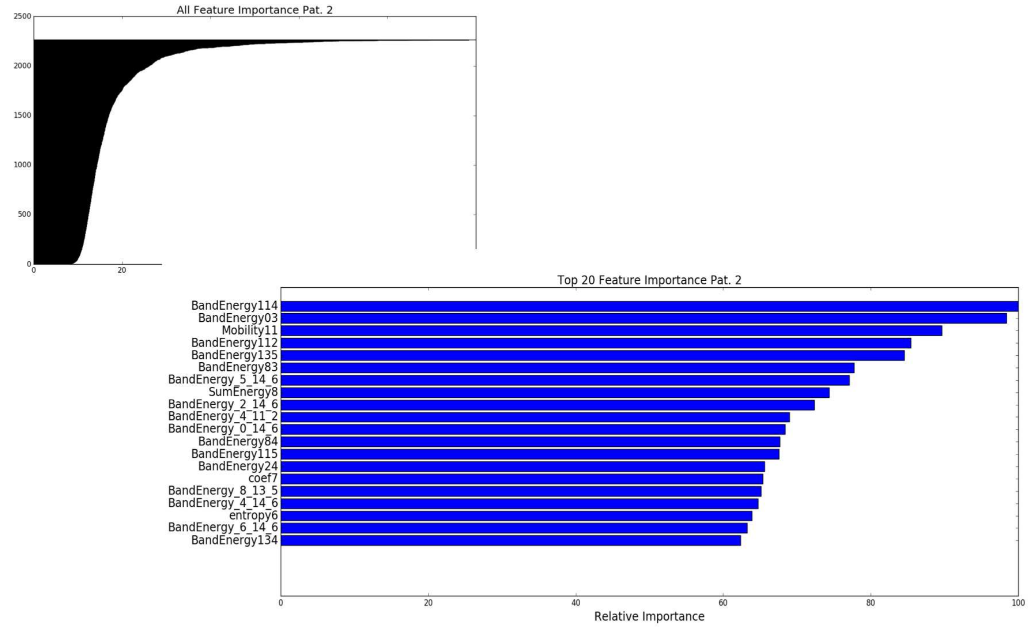

Figure 2: Feature space for patient 2 including the top twenty feature breakdown.

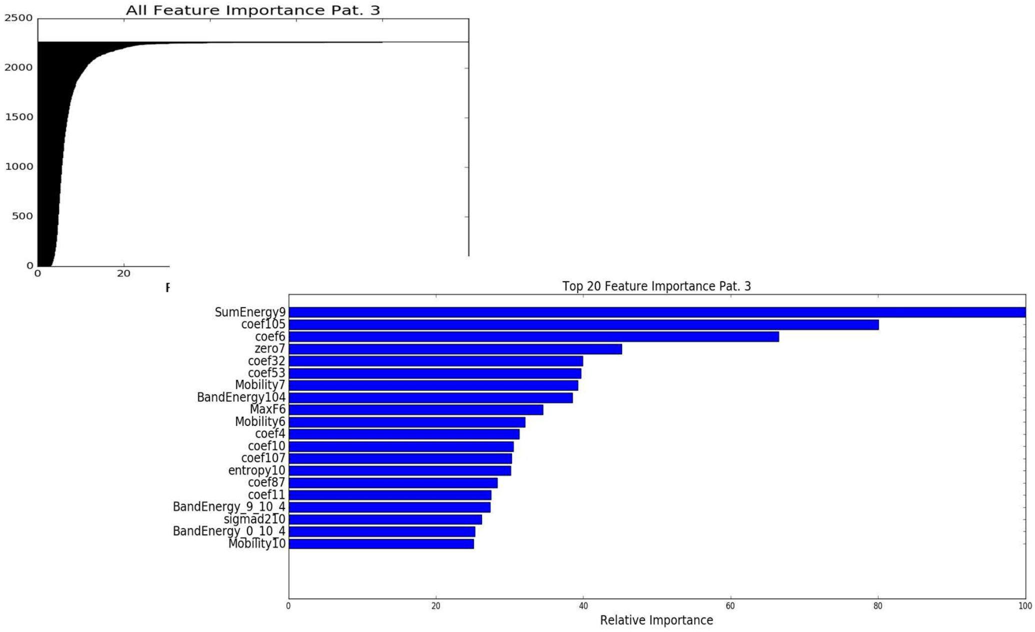

Figure 3: Feature space for patient 3 including the top twenty feature breakdown.

3 Training Methods

As mentioned in the summary, we originally looked to deep learning techniques to identify the features important for classification. However, these techniques were quickly abandoned because of the lack of data. Concerned about over-fitting, we isolated ourselves to a single model and thus did not apply any ensemble schemes in the competition. Feature optimization was briefly explored but not in-depth. Additionally, other classification methods were investigated (logistic regression, gradient boosted trees, etc.) however extra trees consistently out-performed the other models in our validation method. Our validation and testing method was rather simple. A randomly selected hold-out set was used to optimize the various hyper-parameters on a per patient basis —10% for both ambient and target per patient sequences.

4 Interesting Finding

The most important finding was to remain simple. That is, use a single model and a handful of derived features; do not ensemble or over-tune your parameters. This methodology worked rather well as can be seen in the nice agreement between both private and public leader-board scores.

5 Simple Features and Methods

Looking at some earlier submissions, it is worth noting that the short term features alone would have achieved a top 10 result while the long term features alone would have achieved a top 20 result. This was done with similar hyper-parameters and extremely random trees.

6 Dependencies and Execution Time

This analysis was done in the Python/Anaconda suite. Feature generation took the longest (roughly a hour to gather all features for all patients both training and testing) but needed to only occur once. Training took on the order of 1-2 minutes. In short, it was rather speedy.

6.1 Required

- Python 2.7
- numpy-1.11.1
- pandas-0.19.1
- scipy-0.18.0
- scikit-learn-0.18.1

7 How to Generate the Solution

There are three main codes that need to be run in order to reproduce the results. One for generating the features, one to train and one to generate the submission file. Additionally, we removed all the unsafe files from the training sets and added in the 'old' test data for train. In order to incorporate the 'old' test data into our scheme the target label was appended to the filename (e.g. 1_665.mat became 1_665.mat_1.mat)

Generated feature files for all patients for training, hold-out testing and submission file generation are included. Additionally, since a subset of the available training data was held-out for testing, a list of training and hold-out testing files are included. These files are: pat_3_train_files.txt, pat_3_hold_out_test_files.txt, pat_2_train_files.txt, pat_2_hold_out_test_files.txt, pat_1_train_files.txt and pat_1_hold_out_test_files.txt. Recall that 10% of the available training data was used for hold-out testing therefore as a result we only used 90% of the available training data for model generation. If done again, this hold-out data would be included in the final model generation so as to improve predictions.

- get_all_features.py: Generates both the long-term and short-term features and outputs two *.csv files for model generation per patient for both test and train.
- train.py: Trains the model and applies to a hold-out set for hyper-parameter adjustments on a per patient basis.
- predict.py: Generates the a submission file on a per patient basis. The final submission file is obtained by appending together the individual patients results.

References

[1] P. Geurts, D.Ernst, and L. Wehenkel, *Extremely randomized trees*, Machine Learning, 63(1), 3-42 (2006).

[2] A. Temko, et.al., *EEG-based neonatal seizure detection with Support Vector Machines*, Clinical Neurophysiology 120, 464-473 (2011).