Supplementary Material

Epilepsyecosystem.org: Crowd-Sourcing Reproducible Seizure Prediction with Long-Term Human Intracranial EEG

Levin Kuhlmann, Philippa Karoly, Dean R. Freestone, Benjamin H. Brinkmann, Andriy Temko, Alexandre Barachant, Feng Li, Gilberto Titericz Jr., Brian W. Lang, Daniel Lavery, Kelly Roman, Derek Broadhead, Scott Dobson, Gareth Jones, Qingnan Tang, Irina Ivanenko, Oleg Panichev, Timothée Proix, Michal Náhlík, Daniel B. Grunberg, Chip Reuben, Gregory Worrell, Brian Litt, David T.J. Liley, David B. Grayden, and Mark J. Cook.

Contents

1. Team E-mail

Dr Kuhlmann (lkuhlmann@swin.edu.au), Ms Karoly (p.karoly@student.unimelb.edu.au), Dr Freestone (deanrf@unimelb.edu.au), Dr Temko (atemko@ucc.ie), Mr Titericz Jr (titericz@yahoo.com), Dr Lang (blang@arete.com), Dr Jones (gp.jones@ucl.ac.uk), Mr Tang (tqnsjtu@gmail.com), Ms Ivanenko (irinaai@protonmail.com), Mr Panichev (olegxpanichev@gmail.com), Dr Proix (timothee_proix@brown.edu), Mr Nahlik (Nahlik.Michal@seznam.cz), Dr Grunberg (dgrunberg@solverworld.com), Dr Liley (dliley@swin.edu.au), Dr Grayden (grayden@unimelb.edu.au), Dr Cook (markcook@unimelb.edu.au).

2. Seizure details

Seizures were classified as being either clinical (type 1) or clinically equivalent (type 2). Type 1 events were associated with clinical symptoms; type 2 events had no verified clinical symptoms but were electroencephalographically indistinguishable from clinical seizures. Based on the similarity of the ECoG, type 2 seizures were considered relevant for developing methods of seizure prediction and types 1 and 2 seizures were treated equivalently in this work.

3. Pseudo-prospective prediction evaluation details

The Snyder method (Snyder *et al.*, 2008) was used for pseudo-prospective evaluation of algorithms on the held-out data. This method was also used in the original NeuroVista trial (Cook *et al.*, 2013). Critical to the evaluation of the Snyder method is the definition of two key parameters warning duration, τ_w, and detection interval, τ_{w0}. Consistent with the definition of 'preictal' data clips in the contest training data, warning duration and detection interval were set to $\tau_w = 55.5$ min. and $\tau_{w0} = 5.5$ min., respectively, for the evaluation of sensitivity and proportion of time in warning. Consistent with 'interictal' data clips in the contest training data being at least 3 hours before seizure, warning duration and detection interval were set to $\tau_w = 185.5$ min. and $\tau_{w0} = 5.5$ min., respectively, for the evaluation of the proportion of seizures occurring during low-risk and the proportion of time in low-risk. In the evaluation, warning periods were set to override low-seizure-risk periods.

4. Algorithm summaries

Team A – 1st place – Notsorandomanymore

The first place solution was a combination of 11 models. All the models were created subject-specific. In all cases, the EEG was segmented into 20-30 second non-overlapping windows and the models differ in terms of features and classifiers employed. The models were ensembled using the average of ranked probabilities. No cross-subject probabilistic tuning was performed.

For the first 4 models, the raw unprocessed EEG was used to extract four different feature sets, starting with simple relative sub-band powers and cross-frequency coherence, and ending with complex Riemannian tangent space projections of autocorrelation matrices (Barachant *et al.*, 2013; Congedo *et al.*, 2017). The four feature sets were created and fed into a bagged XGBoost classifier. The maximum probability across the 10-minute segment was calculated.

The next 4 models were created on the band-pass filtered EEG. The features were based on statistics of spectral powers in EEG canonical frequency bands, and time/frequency domain cross channel correlations. Two models were built on univariate statistical features with an XGBoost classifier and k-nearest neighbors, and two models were built on a concatenation of univariate and multivariate features, using k-nearest neighbors and logistic regression classifiers.

The remaining 3 models were built on univariate and multi-variate features extracted from both time, frequency and information theory domain to capture energy, frequency, temporal and structural information to form a generic description of the EEG signal (Temko *et al.*, 2011; Temko and Lightbody, 2016). This pool of features was subjected to an XGBoost classifier from which an importance of each feature was computed and top-ranked features were used to create bagged XGBoost, linear support vector machine and logistic regression models.

Team B – 2nd place – Arete Associates

The second place algorithm utilized extremely randomized trees (Geurts *et al.*, 2006) on both long-term (10-minute) and short-term (1-minute) derived features to generate a single model per patient. Long-term features (Temko *et al.*, 2011) involved correlation coefficients between channels at full temporal resolution and spectral and temporal statistics on a per channel basis, down sampled by a factor of five. The short-term features, at full temporal resolution, consisted of normalized summed energy in the standard spectral bands (delta, theta, alpha, low and high gamma), derived on a minute-by-minute basis with 30 seconds of overlap. Algorithm implementation was done in Python using the scikit-learn toolkit.

Team C – 3rd place – Gareth Jones

For each subject, 10 minute segments of data were concatenated into hour segments (where possible) then epoched into shorter time windows. Three window lengths were used; 80, 160 and 240 s. For each window length, features were extracted from non-overlapping sequential windows, then compiled into a structured table containing the features from all three window lengths. Features included frequency band powers (delta-gamma, and 50Hz bands up to 200Hz), cross-channel correlations in the temporal and frequency domains, and temporal summary statistics (e.g. skewness, kurtosis). Overall, features extracted from the longer window lengths had the most predictive power. A random undersampling boosted tree ensemble and a quadratic support vector machine were trained on the extracted features from all subjects in MATLAB 2016b (MathWorks Inc, Natick MA) using grouped K-fold cross validation, with data group by original hour segment. The final predictions were an unweighted mean of the normalised (z-score) individual model predictions.

Team D – 4th place – qingnantang

The iEEG data were split into 75s non-overlapping windows and resampled to 100Hz. The real fast Fourier transform was applied to the resampled data to get a frequency spectrum from 0 to 50Hz. Features include the time correlation matrix (and eigenvalues) between channels of resampled data, edge frequency at 50% power, power and entropy of the spectrum bands split by two methods (see Supplementary Material), correlation matrix (and eigenvalues) of the bands, and square of all the previous features. Gradient Boosting machine (Friedman, 2001) with multiple estimators was trained to classify the features from each window, and the finalized preictal probability is an average of prediction probabilities from all windows within the same patch. All the code was written in Python and used scikit-learn toolkit (http://scikit-learn.org/stable/).

Team E – 5th place – nullset

The fifth place team final model is an ensemble of seven different models. AdaBoost, Gradient Boosting (Friedman, 2001), Random Forest, XGBoost, GridSearch and Voting classifier were applied for a different combination of feature sets in a patient-specific approach. Each 10 minute EEG segment was divided into epochs with 30 seconds duration. These epochs were considered as separate objects that all belong to the same class as the whole 10-minute segment. Features were extracted separately for each epoch and included: features extracted from power spectral density (e.g. values of power in EEG rhythms and their relations), correlation matrices and their eigenvalues calculated for EEG channels and spectrums of channels, spectral entropy for dyadic bands, Shannon entropy, spectral edge frequency, Hjorth parameters, fractal dimensions. All algorithms were implemented in Python and used libraries: scikit-learn, XGBoost, and pandas.

Team F – 6th place – tralala boum boum pouêt pouêt

Data were sampled in sequential 1-min windows, in which spectral features were computed at eight frequency bands (0-4 Hz, 4-8 Hz, 8-12 Hz, 12-18 Hz, 18-25 Hz, 25-50 Hz, 50-80 Hz, 80-150 Hz), including spectral power, coherence matrix and its eigenvalues, and spectral entropy. Temporal features included the correlation matrix and its eigenvalues, Petrosian fractal dimension (Petrosian, 1995), Hjorth mobility and complexity parameters (Hjorth, 1970), variance, skewness and kurtosis. Additional features were obtained by taking the output of the penultimate layer of either a convolutional neural network (LeCun *et al.*, 1998) trained on the above features, or a long-short-term-memory network (Hochreiter and Schmidhuber, 1997) trained on the spectral power only. An ensemble of nine models was used to obtain the final prediction, using three types of classifiers: (i) a support vector machine (Cortes and Vapnik, 1995); (ii) a random forest algorithm (Breiman, 2001); (iii) a XGBoost algorithm (Chen and Guestrin, 2016).

Team G – 8th place – michaln

Features were calculated for each channel on the whole 10 minute data files: mean value, standard deviation, spectral edge at 50% power below 40 Hz, skewness, kurtosis, Hjorth parameters, Shannon's entropy across energies in each frequency band (0.1-4 Hz,4-8 Hz,8-14 Hz,14-32 Hz,32-70 Hz,70-180 Hz) and at each dyadic level, maximum correlation between channels in the interval ±0.5 seconds, correlation between channels in frequency domain and between channel power spectrums at dyadic levels, Brownian, Petrosian and Katz fractal dimensions, singular values of 10 scale wavelet transformation using Morlet wave (Brinkmann *et al.*, 2016) (code available at https://www.kaggle.com/treina/feature-extractor-matlab2python-translated). These features were used to train 10 classification decision trees for each channel and patient in 10-fold cross validation using the exact search training algorithm. The final classification of a file was calculated as the mean output across channel models for the patient. Algorithms were coded in MATLAB (MathWorks Inc, Natick MA) using the statistical and machine learning toolbox.

Team H – Special case: best team for patient 3 – Chipicito+SolverWorld

The approach consisted of two main algorithms. In algorithm 1 intracranial EEG data were windowed into 60 sec non-overlapping segments. Features were the total energy in the fast Fourier transform in bands demarcated at 1,2,4,8,16,32 and 64 Hz, plus the ratio of the total energy in band 10-25 Hz to 0.1-10Hz. Two models were used: (1A) an XGBoost boosted-tree model with 160 rounds, and (1B) a 600-tree random forest. Each model used a 7-fold validation and prediction strategy, with models generated separately for each patient. Results from (1A) and (1B) were combined in the ratio (3:1). In algorithm 2 intracranial EEG data were divided into 15-second sub-segments. Short-time-Fourier-transform without windowing was performed, and absolute values selected at approximately 2 to 50 Hz based on previously-reported findings (Mormann and Jefferys, 2013) and averaged down to 126 frequency groups. The 16 channels by 126 frequency groups were fed into a 3-layer convolutional neural network (LeCun *et al.*, 1998). The preictal probability was the sub-segment preictal prediction frequency. The two algorithms were rank averaged at a 4:1 ratio (algorithms 1:2).

5. Algorithm details

Details of the algorithms of each team considered in the held-out evaluation are included in this document after the list of references. These descriptions are in the format requested by Kaggle.com for post-contest surveys.

6. ROC classification curves for contest and held-out data

Figure S1 shows the ROC classification curves for contest and held-out data, and compares them against the results from the 2014 'American Epilepsy Society Seizure Prediction Challenge' (Brinkmann *et al.*, 2016).

Comparing the contest private leaderboard and held-out data ROC curves, we see the held-out set performance is reasonably stable given that the held-out data set contains much more data. Interestingly, circadian prediction performs well relative to the other algorithms for the held-out data. Comparing the held-out data ROC curves resulting from the current competition and the 2014 competition it can be seen that the true positive rates for the 2014 results appear higher for false positive rates below 0.2, while the true positive rates for the 2016 results appear higher for false positive rates between 0.4 and 0.6. It can be noted that the held-out data set corresponding to the current contest involves the most challenging long-term human data available (based on seizure prediction performance in the original NeuroVista trial (Cook *et al.*, 2013)), while the 2014 contest held-out data involved long-term dog data only. Moreover, the current held-out dataset involves 12 times more data clips than used for the 2014 contest follow-up. When it comes to clinical application, this gives us more confidence in the results we see for the current held-out dataset.

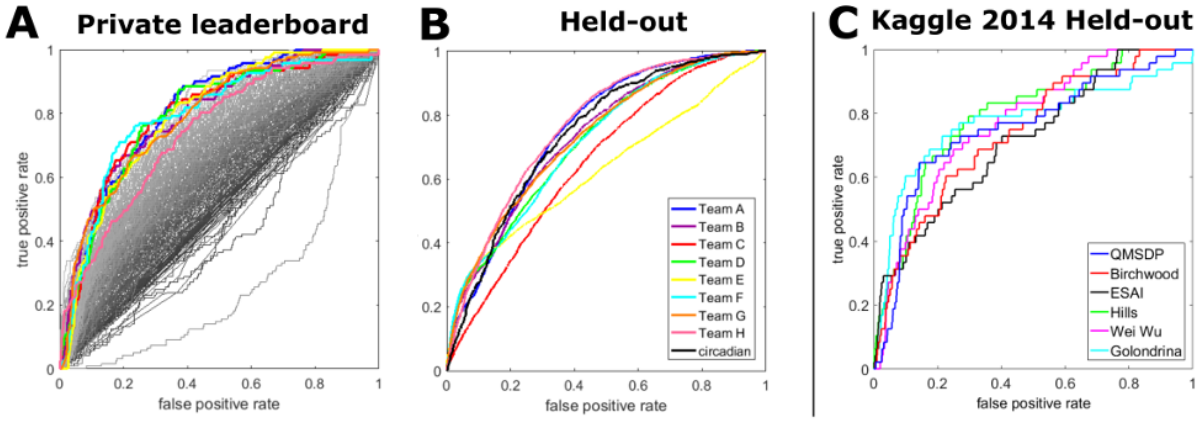


Figure S1. ROC classification curves for (A) the private leaderboard results for the 478 teams that entered the contest, (B) the eight teams (plus the circadian predictor) considered in the held-out evaluation, and (C) the results from the held-out evaluation for the 2014 Kaggle.com 'American Epilepsy Society Seizure Prediction Challenge' (Brinkmann *et al.*, 2016). The legend in (B) applies to (A) and (B) and lists teams in decreasing ranking on the private leaderboard as well as the circadian predictor (note the circadian predictor is not considered in (A)). In each subplot the y- and x-axes correspond to true and false positive rate, respectively. In (A) the gray ROC curves correspond to the teams not included in the held-out evaluation and the gray tone becomes lighter as the corresponding AUC value increases.

7. AUC performance for individuals as a function of time

Figure S2 plots the AUC performance for each individual patient for all of the held-out data, as well as the held-out data broken into consecutive quarterly (3 month) periods. Quarterly period durations were used as they make it more likely that the period will contain an adequate number of preictal periods to properly calculate AUC. Quarterly periods also represent a practical clinical interval for periodic evaluation of a patient's seizure prediction performance. From Figures S2A-C it can be seen that different algorithms achieved the best AUC performance for each patient when considering all of the held-out data. It can also be noted that there are no clear trends in AUC performance as a function of time, except for a possible increase in AUC for patient 3, for example for Team B. To assess if any AUC trends were in part dependent on the proportion of preictal data clips, Pearson's correlation coefficient between AUC and the proportion of preictal clips for quarterly data blocks was computed. Statistical significance of the correlation coefficient values was assessed with a t-test (Fisher, 1925) under the assumption that quarterly blocks were statistically independent. For Teams C and D, p-values below 0.05 were obtained for patient 3 but these would not survive correction for multiple comparisons for all of the teams considered. Therefore, there is no strong evidence that AUC shows a dependence on the proportion of preictal clips for the time scale considered, but it cannot be ruled out as a possibility.

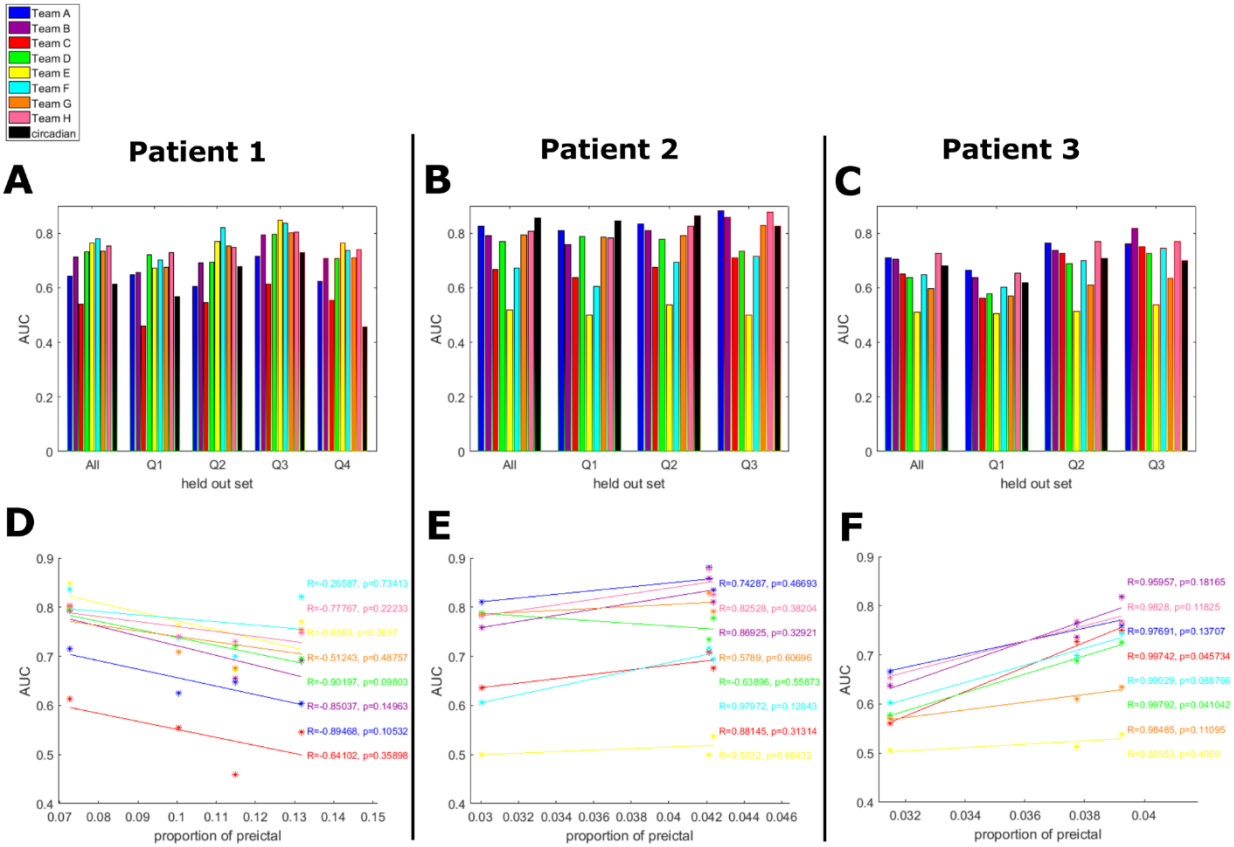


Figure S2. AUC performance as a function of time for the held-out dataset for patients (A) 1, (B) 2, and (C) 3. Associated tests of the dependence of AUC on the proportion of preictal clips in quarterly (3-month) blocks of held-out data for patients (D) 1, (E) 2, and (F) 3. The legend at the top applies to (A)-(F) and lists the teams in the held-out evaluation in order of decreasing rank on the private leaderboard, as well as the circadian predictor (Note the circadian predictor was not considered in (D)-(F)). In (A)-(C), 'All' corresponds to all of the held-out data, whereas 'Q1', 'Q2', 'Q3' and 'Q4' refer to consecutive 3-month quarterly blocks of held-out data. In (D)-(F), the y- and x-axes correspond to AUC and the proportion of preictal clips for quarterly data blocks, respectively. Solid lines indicating lines of best fit, correlation coefficient values, *R*, and p-values are provided for each team.

8. AUC performance of sub-models for Team A

As described in the algorithm summary for Team A, their system was composed of 11 sub-models. The AUC performance of these models is described in Table S1 for the contest and held-out data.

Table S1. AUC scores for the 11 sub-models for Team A for the held-out data experiment and the public and private leaderboards

Window (overlap)	Features	Machine learning algorithm	Public leader-board	Private leader-board	Held-out data	Percent change	Sensitivity at 75% specificity
30s (0s)	standard deviation, spectral power	extreme gradient boosting	0.75909	0.77598	0.72769	-6.2227	0.52555
30s (0s)	1965 features	extreme gradient boosting	0.7731	0.77237	0.73619	-4.6845	0.51992
20s (0s)	spectral power	extreme gradient boosting	0.8072	0.76908	0.73336	-4.6441	0.53962
20s (0s)	cross-frequency coherence	extreme gradient boosting	0.75004	0.76703	0.73316	-4.4158	0.579
30s (0s)	200 features	generalized linear model	0.71313	0.75856	0.76151	0.38845	0.61697
30s (0s)	300 features	linear support vector machine	0.7358	0.75369	0.71628	-4.9633	0.50352
20s (0s)	spectral power, distribution statistics, AR error, fractal dimensions, Hurst exponent	extreme gradient boosting	0.77081	0.74481	0.74665	0.24743	0.58228

30s (0s)	standard deviation, spectral power	k-nearest neighbors	0.79631	0.74006	0.62821	-15.1138	0.39194
50s (0s)	standard deviation, spectral power, correlation	generalized linear model	0.67852	0.73308	0.72471	-1.1418	0.55274
30s (0s)	standard deviation, spectral power, correlation	k-nearest neighbors	0.75947	0.7267	0.60121	-17.2692	0.38068
20s (0s)	Riemannian autocorrelation	extreme gradient boosting	0.62951	0.69535	0.73705	5.9963	0.55087

Sub-models are listed in order of performance on the private leaderboard data.

9. True prediction times relative to seizure onset

Figure S3 captures the distribution of predictions occurring in the window of -65 to -5 minutes before seizure onset computed with the held-out data for each patient. The distributions show a strong dependence on time of warning and algorithm.

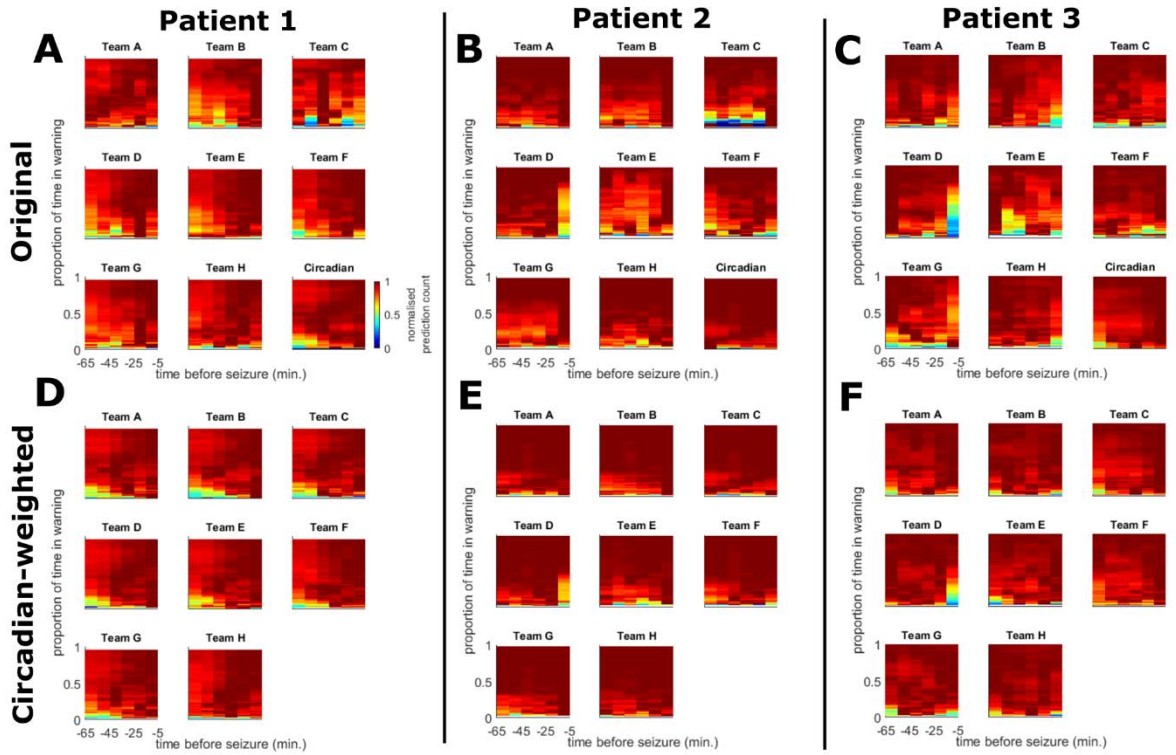


Figure S3. Distribution of true predictions relative to seizure onset as a function of proportion of time in warning for each team considered in the held out evaluation, as well as the circadian predictor. (A), (B) and (C) correspond to patients 1, 2 and 3, respectively, for the original versions of the algorithms, as well as the circadian predictor. (D), (E) and (F) correspond to patients 1, 2 and 3, respectively, for the circadian-weighted versions of the algorithms. In each sub-figure for a given algorithm the y-axis represents proportion of time in warning and the x-axis represents time before seizure (in minutes). The colorbar in (A) applies to (A)-(F) and indicates the normalised true prediction count, calculated by dividing each prediction count for a preseizure bin for a given proportion of time in warning by the maximum prediction count across the preseizure bins for the same time in warning. This normalisation facilitates viewing of the distributions across different proportions of time in warning and algorithms.

References

Barachant A, Bonnet S, Congedo M, Jutten C. Classification of covariance matrices using a Riemannian-based kernel for BCI applications. Neurocomputing 2013; 112: 172-8.

Breiman L. Random forests. Machine learning 2001; 45(1): 5-32.

Brinkmann BH, Wagenaar J, Abbot D, Adkins P, Bosshard SC, Chen M, *et al.* Crowdsourcing reproducible seizure forecasting in human and canine epilepsy. Brain 2016; 139(Pt 6): 1713-22.

Chen T, Guestrin C. Xgboost: A scalable tree boosting system. Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining; 2016: ACM; 2016. p. 785-94.
Congedo M, Barachant A, Bhatia R. Riemannian geometry for EEG-based brain-computer interfaces; a primer and a review. Brain-Computer Interfaces 2017: 1-20.
Cook MJ, O'Brien TJ, Berkovic SF, Murphy M, Morokoff A, Fabinyi G, *et al*. Prediction of seizure likelihood with a long-term, implanted seizure advisory system in patients with drug-resistant epilepsy: a first-in-man study. Lancet Neurol 2013; 12(6): 563-71.
Cortes C, Vapnik V. Support-vector networks. Machine learning 1995; 20(3): 273-97.
Fisher RA. Statistical methods for research workers: Genesis Publishing Pvt Ltd; 1925.
Friedman JH. Greedy function approximation: a gradient boosting machine. Annals of statistics 2001: 1189-232.
Geurts P, Ernst D, Wehenkel L. Extremely randomized trees. Machine learning 2006; 63(1): 3-42.
Hjorth B. EEG analysis based on time domain properties. Electroencephalography and clinical neurophysiology 1970; 29(3): 306-10.
Hochreiter S, Schmidhuber J. Long short-term memory. Neural computation 1997; 9(8): 1735-80.
LeCun Y, Bottou L, Bengio Y, Haffner P. Gradient-based learning applied to document recognition. Proceedings of the IEEE 1998; 86(11): 2278-324.
Mormann F, Jefferys JG. Neuronal firing in human epileptic cortex: The ins and outs of synchrony during seizures. Epilepsy Currents 2013; 13(2): 100-2.
Petrosian A. Kolmogorov complexity of finite sequences and recognition of different preictal EEG patterns. Computer-Based Medical Systems, 1995, Proceedings of the Eighth IEEE Symposium on; 1995: IEEE; 1995. p. 212-7.
Snyder DE, Echauz J, Grimes DB, Litt B. The statistics of a practical seizure warning system. Journal of neural engineering 2008; 5(4): 392.
Temko A, Lightbody G. Detecting neonatal seizures with computer algorithms. Journal of Clinical Neurophysiology 2016; 33(5): 394-402.
Temko A, Thomas E, Marnane W, Lightbody G, Boylan G. EEG-based neonatal seizure detection with support vector machines. Clinical Neurophysiology 2011; 122(3): 464-73.

Kaggle Melbourne University AES/MathWorks/NIH Seizure Prediction challenge - Winning Solution

Authors: * Alexandre Barachant * Andriy Temko * Feng Li * Gilberto Titericz Junior

Contents :

Overview of the winning solution

The winning solution is a blend of 11 models created by the team members before they teamed up. All models were created subject-specific. No usage of test data and no cross-subject probabilistic tuning was performed. The blend is done using an average of ranked predictions of each individual models. The blend has been designed to reduce overfitting and improve robustness of the solution. To this end, we limited ourselves to the minimum of weight tuning, choosing a weight of 1 for all models.

Each model will be described in details below

Models

Alex and Gilberto Models

A total of 4 models were selected for the final ensemble (see table 1).

For all models, preprocessing consisted in segmentation of the 10 minutes segment into 30 non-overlapping 20 seconds segment. No filtering or artifact rejection was applied. After modeling, the maximum of the prediction of this 30 smaller time window was affected to the 10 minute segment.

Total training time (including feature extraction) is estimated to less than half a day for these 4 models. We used python, scikit-learn, pyRiemann, xgboost, mne-python and pandas.

Alex_Gilberto_relative_log_power_XGB.csv

Features : This dataset consist in the normalized log power in 6 different frequency band (0.1 - 4 ; 4- 8 ; 8 - 15 ; 15 - 30 ; 30 - 90 ; 90 - 170 Hz) and for each channel. Power spectral density was estimated using Welch's method (window of 512 sample, 25% overlap). PSD was averaged in each band, normalized by the total power before applying a logarithm. Total size of this dataset is 6 x 16 = 96 features.

Model : XGB, 10 bags

Alex_Gilberto_all_flat_dataset_XGB.csv

Features : This dataset include the relative log power dataset mentioned before with the addition of various measures including signal statistics (mean, min, max, variance, 90th and 10th percentiles), AR error coefficient (order 5), Petrosian and Higuchi fractal dimension and Hurst exponent. Total size of this dataset is 21 x 16 = 336 features

Model : XGB, 5 bags

Alex_Gilberto_autocorrmat_TS_XGB.csv

Features : For each channel, an auto-correlation matrix was estimated by concatenating time-delayed single channel signal before estimation of correlation matrix. Signal was downsample by 2 and 8 logarithmically spaced delays were used (0, 1, 2, 4, 8, 16, 32, 64). Each of the autocorrelation matrices were projected into their respective riemannian tangent space (see [1], this operation can be seen as a kernel operation that unfold the natural structure of symmetric and positive define matrices) and vectorized to produce a single feature vector of 36 item. Total size of this dataset was 36 x 16 = 576.

Model : XGB, 4 bags.

Alex_Gilberto_coherences_transposed_TS_XGB.csv

Features : This feature set is composed by cross-frequency coherence (in the same 6 sub-band as in the relative log power features) of each channels, i.e. the estimation of coherence is achieved between pairs of frequency of the same channel instead to be between pairs of channels for each frequency band. This produce set of 6x6 coherence matrices, that are then projected in their tangent space and vectorized. Total size of this dataset is 21 x 16 = 336.

Model : XGB, 10 bags.

Table 1.

Model name	Public Score	Private Score
Alex_Gilberto_all_flat_dataset_XGB.csv	0.77081	0.74481
Alex_Gilberto_relative_log_power_XGB.csv	0.80720	0.76908
Alex_Gilberto_autocorrmat_TS_XGB.csv	0.62951	0.69535

Model name	Public Score	Private Score
Alex_Gilberto_coherences_transposed_TS_XGB.csv	0.75004	0.76703
Ensemble	**0.77276**	**0.77439**

Models that did not make it into the ensemble

The difficulty (or impossibility) to build a reliable cross-validation procedure was making very difficult to select best performing feature set properly. Other features developed during this challenge include coherence, correlation, spectral edge frequency, cumulative power and peak frequency covariance. In addition to XGB modeling, logistic regression was used but led to slightly lower performance (depending on the feature set).

Remark

When slicing larger 10 min segment with a 20s time window, comes the question of how to optimally combine those predictions. We chose here to use the maximum probability to represent the probability of the 10 min segment. The rationale was that patterns predictive of seizures were likely not to be stationary during the whole 10 minutes. Post-competition analysis revealed that using mean of probability led to a significant decrease in performances (0.674 Public / 0.722 Private). Very interestingly, using standard deviation of probability led to an increase in performances (0.805 Public / 0.776 Private). This finding seems to validate our initial hypothesis. However, this observation only holds true for the 4 models described above.

Feng models

A total of 4 models were selected for the final ensemble (see table 2)

Preprocessing : The Butterworth filter (5th order with 0.1-180 HZ cutoff) was firstly applied to the raw data and then Ipartitioned the raw data into non-overlapping 30s windows(for GLM model, I used non-overlapping 50s windows). A combination of arithmetic mean of individual analysis windows was used to aggregate them into a single probability score for each 10-minute segment.

Total training time (including feature extraction) is estimated to less than 5 hours for these 4 models on my 8 GB RAM MacBook Pro .

Features :

- The standard deviation and average spectral power in delta (0.1–4 Hz), theta (4–8 Hz), alpha (8–12 Hz), beta (12–30 Hz), low gamma (30–70 Hz) and high gamma (70–180Hz)
- The correlation in time domain and frequency domain (upper triangle values of correlation matrices) with their eigenvalues.

Models :

1. Standard deviation and average spectral power were used in the XGB classifier and KNN classifier.
2. All features were used in Logistic Regression with L2 penalty(Ridge).

3. All features were used in the second KNN classifier.

Remark

I read relevant papers at the beginning of this competition and found one could generate thousands of features from the raw EEG. Because I don't have background of signal digital processing, I generated those features based on common features used in some important papers and my intuition. Too many noise features and correlated features would damage the performance of most classifiers.

Table 2.

Model name	Public Score	Private Score
Feng_xgb.csv	0.75909	0.77598
Feng_knn.csv	0.79631	0.74006
Feng_knnmorefeature.csv	0.75947	0.72670
Feng_glmmorefeature.csv	0.67852	0.73308
Ensemble	**0.80165**	**0.79044**

Andriy models

A total of 3 models were selected for the final ensemble (see table 3).

Preprocessing: for all models preprocessing consisted in a) demeaning the EEG signal, b) filtering of the EEG signal between 0.5 and 128 Hz with a notch filter set at 60Hz, c) downsampling to 256 Hz, d) segmentation of the 10 minutes segment into non-overlapping 30 seconds segment. After modeling, the maximum probability was taken to represent the probability of preictal for the whole 10m window.

Feature extraction: the features can be divided into two groups, per-channel feature (sometimes called univariate) and cross-channel features (multivariate). From each EEG channel, 111 feature were extracted from both time, frequency and information theory domain to capture energy, frequency, temporal and structural information and to form a generic description of the EEG signal. These features have been previously used in several EEG applications, including seizure detection in newborns and adults [2-3]. These include: peak frequency of spectrum, spectral edge frequency (80%, 90%, 95%), fine spectral log-filterbank energies in 2Hz width sub-bands (0-2Hz, 1-3Hz, ...30-32Hz), coarse log filterbank energies in delta, theta, alpha, beta, gamma frequency bands, normalised FBE in those sub-bands, wavelet energy, curve length, Number of maxima and minima, RMS amplitude, Hjorth parameters, Zero crossings (raw epoch, Δ, ΔΔ), Skewness, Kurtosis, Nonlinear energy, Variance (Δ, ΔΔ), Mean frequency, band-width, Shannon entropy, Singular value decomposition entropy, Fisher information, Spectral entropy, Autoregressive modelling error (model order 1-9). These led to 111*16=1776 features in a concatenated feature vector.

Apart from univariate measures, autoregressive modelling error (model order 1-9) was extracted from a single channel following common spatial filtering. The cross-channel features consisted of the following characteristics: lag of maximum cross correlation, correlation, brain asymmetry, brain

synchrony index, coherence, and frequency of maximum coherence. These 6 features were extracted for the five conventional EEG subbands (delta, theta, alpha, beta, gamma) for 6 different montages (horizontal, vertical, diagonal, etc) leading to 180 features.

Both univariate and multivariate features form a pool of 1965 features.

Feature selection: for feature selection a pool of features was subjected to an XGB classifier from which an importance was computed and top N features were used for some models.

Modelling:

1) All features were used in a bagged XGB classifier (XGB).

2) Linear SVM was trained with top 300 features (SVM)

3) GLM was trained with top 200 features (glmnet)

Table 3.

Model name	Public Score	Private Score
Andriy_submission5_7_SVM.csv	0.73580	0.75369
Andriy_submissionLR5_3_glmnet.csv	0.71313	0.75856
Andriy_submissionXGB7_5mean.csv	0.77310	0.77237
Ensemble	**0.75774**	**0.78247**

Whole team Ensemble

Table 4.

Model name	Public Score	Private Score
Ensemble	**0.80630**	**0.80856**

comment: Due to some reproducibility issues discovered during code release, the score of the ensemble obtained from this code is different from the one submitted durring the competition. New score shows a slight increase in private score and slight decrease in public score.

CV method

Building a representative CV was one of the most challenging tasks in this competition, especially after the leak. We tried many CV approaches and most of them gave us too optimistic AUC scores.

Finally, our CV score is mainly composed of observing the score of the 2 approaches. Both of them were not perfect and served as an indicator whether to trust the LB or not. Both CV approaches are based on the integrity of 1-hour segments and only "safe" data were used.

2-Fold CV: The first fold is the first half interictal and the train preictal. The second fold is the second half of the interictal and the old test preictal. The problem of this approach is that we have

time-based split of preictal data and no such split for interictal data. Also, as only 2 folds are used the results possess natural measurement noise.

26-Fold CV: Because we have 25 1-hour preictal segments for train per patient, we put each 1 hour to form first 25 folds, whereas as no sequence was provided for old test data, all of it was placed to the 26th fold. Interictal are split into 26 folds based similarly preserving the 1-hour sequence integrity and keeping the proportion of the preictal to interictal roughly equal in all folds.

References

[1] Barachant, A., et al., Classification of covariance matrices using a Riemannian-based kernel for BCI applications, Neurocomputing 112 (2013): 172-178.

[2] Temko, A., et al., EEG-based neonatal seizure detection with support vector machines, Clin. Neurophysiol. 122 (2011): 464-473.

[3] Temko, A., et al., Performance assessment for EEG-based neonatal seizure detectors, Clin. Neurophysiol. 122 (2011): 474-82.

Reproduce the solution

The code corresponding to each group of model is available in separate folders. The solution can be reproduced in three steps :

- 1 : place the data in the `data` folder
- 2 : Go in each of the 3 following folders and follows instruction given in the `README` of each folder:
  - `Alex_Gilberto`
  - `Andriy`
  - `Feng`
- 3 : run `python make_blend.py` to blend the 11 different models.

Instruction for Hold-out evaluation

The best solution is to replace new_test data with hold out data using the same file s naming convention, and re-generate the solution from scratch following the three steps above.

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

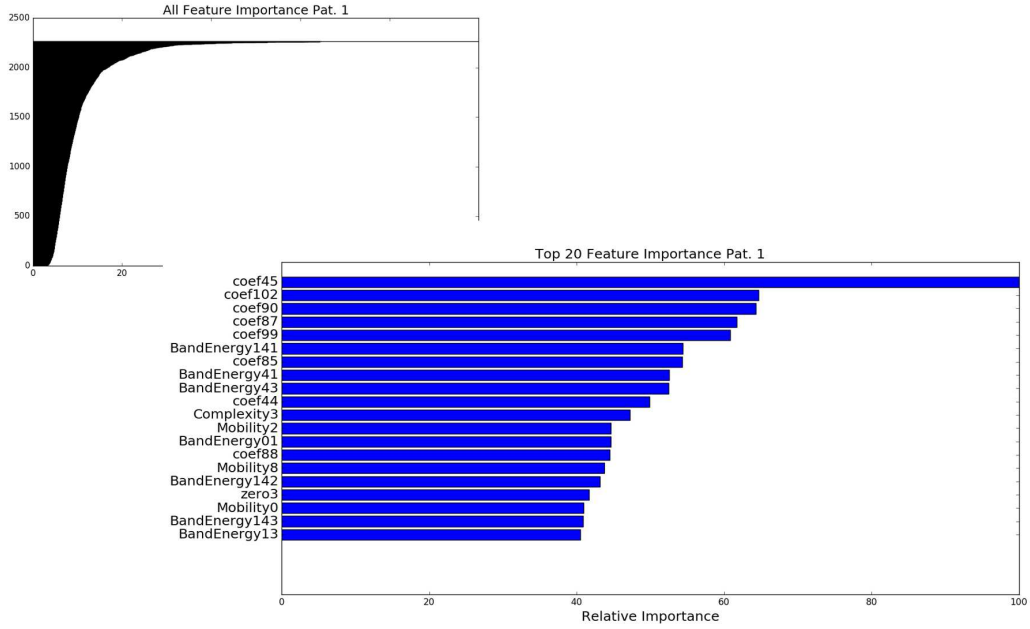


Figure 1: Feature space for patient 1 including the top twenty feature breakdown.

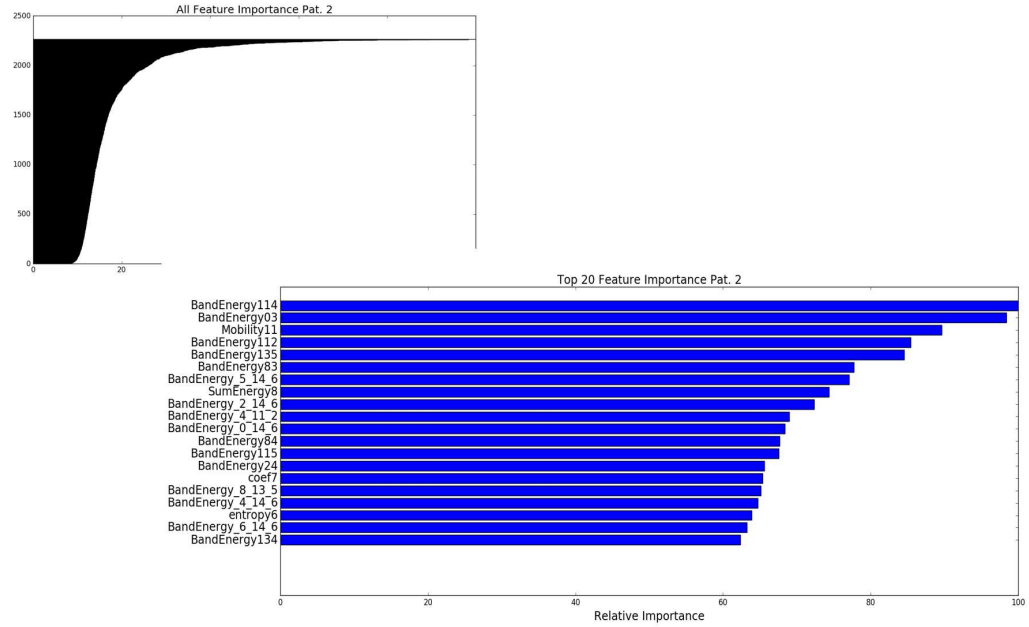


Figure 2: Feature space for patient 2 including the top twenty feature breakdown.

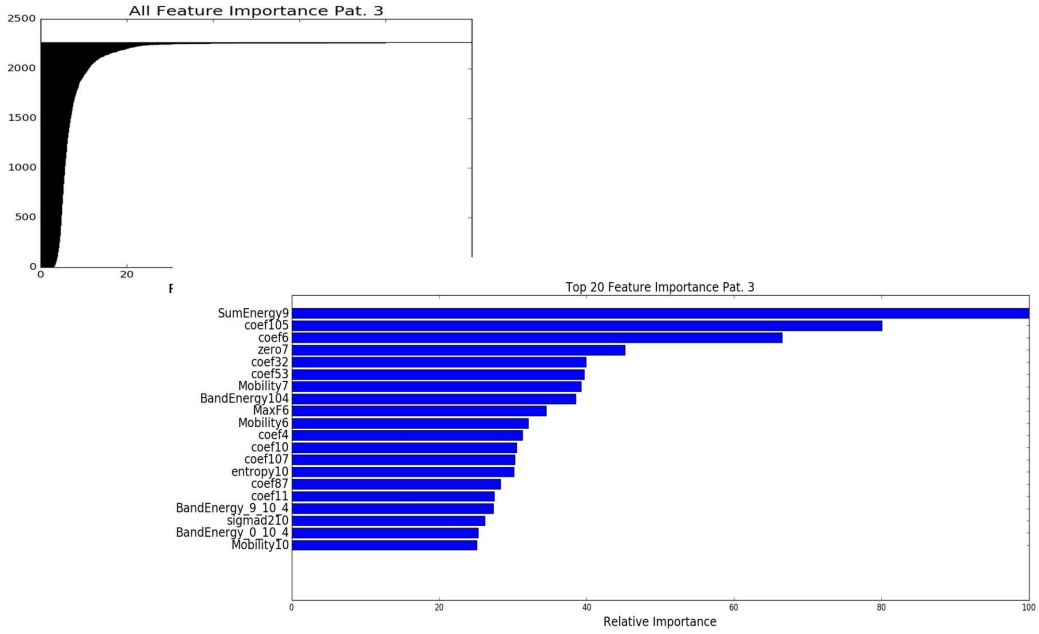


Figure 3: Feature space for patient 3 including the top twenty feature breakdown.

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).

Winning Model Documentation

Name: Gareth Jones

Location: London, UK

Email: garethjns4@gmail.com

Competition: Melbourne University AES/MathWorks/NIH Seizure Prediction

1. Background on you/your team

If part of a team, please answer these questions for each team member. For larger teams (3+), please give shorter responses.

- What your academic/professional background?

  I have a PhD in neuroscience and currently work as an experimental/computational neuroscientist at the UCL Ear Institute, London, UK. I work on evidence accumulation and multisensory (auditory and visual) decision making.
- Did you have any prior experience that helped you succeed in this competition?

  I have experience dealing with single unit neural data, although not EEG data specifically or in seizure prediction.
- What made you decide to enter this competition?

  It was a nice crossover between my area of expertise and machine learning, which I have a lot of interest in but don't often get to deploy in my own work.
- How much time did you spend on the competition?

  A significant amount of my spare time!
- If part of a team, how did you decide to team up?
- If you competed as part of a team, who did what?

2. Summary

4-6 sentences summarizing the most important aspects of your model and analysis, such as:

- The training method(s) you used (Convolutional Neural Network, XGBoost)

  I used a combination of a polynomial SVM (quadratic) and a RUS boosted tree ensemble (RBT). I focused on training a general model for all of the subjects, rather

than create single-subject models and combining the predication across subjects. My best solution came from an ensemble of the general SVM and general RBT.

- The most important features

  I haven't rigorously quantified this, but I think power in low frequency bands was the most significant group of features, followed by summary statistics and correlation between channels.

- The tool(s) you used

  MATLAB 2016b and Classifier Learner App.

- How long it takes to train your model

  Feature generation could take between 2-12 hours. Training 1-4 hours depending on number of CV folds.

3. Features Selection / Engineering

- What were the most important features?

  All included features appeared to have some value, although I have not rigorously quantified this. There were a number of types of features generated, which were as follows, roughly in order of value. I didn't apply any PCA or feature selection.

  - **bandsLin2D** and **bandsLinAv**
    - **bandsLin2D**: Linear power (FFT) in the frequency bands: 1-3, 4-7, 8-9, 10-12, 13-17, 18-30, 31-40, 41-50, 51-70, 71-150, 151-250 Hz, for each channel (176 features).
    - **bandsLinAv**: A version of bandsLin2D averaged across channels (11 features)
  - **summ32D and summ3Av**
    - **summ32D:** Summary statistics for each channel. Mean of absolute data, mean, standard deviation, RMS, RMS of first derivative, RMS of second derivative, kurtosis, skewness (128 features).
    - **summ3Av:** summ32D averaged across channels (16 features).
  - **mCorrsT, mCorrsF**
    - **mCorrsT**: Summary statistics of the correlation matrix of all channels in the temporal domain. Mean, standard deviation, sum, and sum of abs (64 features).
    - **mCorrsF**: Summary statistics of correlation matrix of all channels in frequency domain. Mean, standard deviation, and sum (48 features).
  - **maxBands2D and maxBandsAv**

    - **maxBands2D**: The bands in bandsLin2D were labelled 1 to 11 and maxBands2D contained the label for the most powerful band for each channel.
    - **maxBandsLin**: A version of maxBands2D averaged across channels.
- **hillsBandsLog2D and hillsBandsLogAv**
    - **hillsBandsLog2D:** Based on https://github.com/MichaelHills/seizure-detection**.** Similar to bandsLin2D but returning the real log power in 1 Hz frequency bins between 1-47 Hz. (0 features, this group of features was not used in the best submission).
    - **hillsBandsLogAv**: A version of hillsBandsLog2D averaged across channels (47 features – this averaged version was used).
- **maxHills2D and maxHillsAv:**
    - Similar to maxBands2D and maxBandsAv, but using the values from hillsBandsLog2D.

- How did you select features?

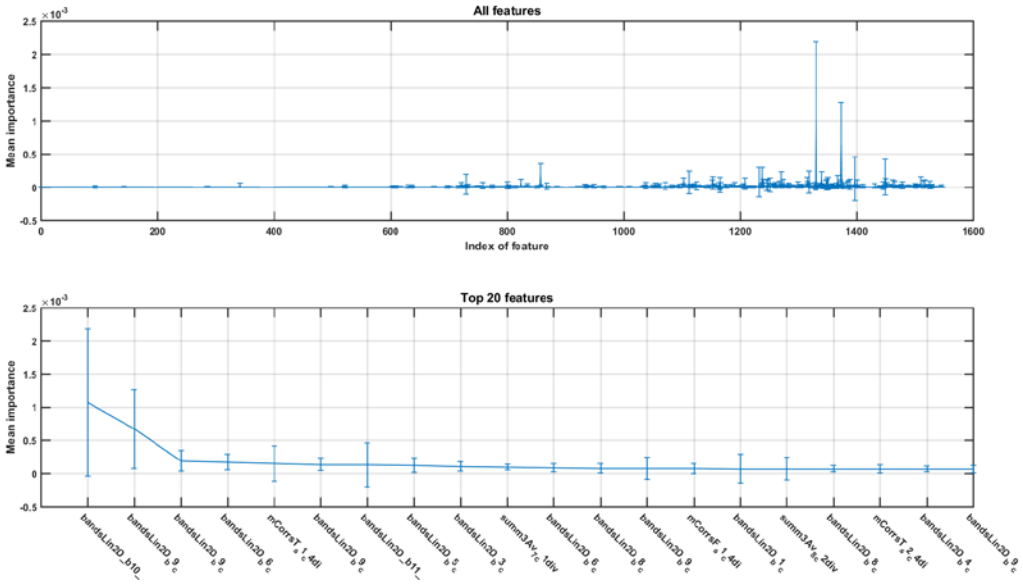
  Groups of features were generated and iteratively added to the training set. If they harmed performance, they were removed. Only features improving performance were kept. I didn't use PCA or any feature selection, I'm not sure if PCA would work well with the across-subject variation.
- Did you make any important feature transformations?

  The above features were calculated on the raw data divided in to windows of varying lengths. For the best submission, epoch lengths of 240, 160, and 80s were used. This data was joined together to create the training/test sets. Predictive values of different window lengths appeared roughly equal, but improved when multiple window lengths were used.
- Did you find any interesting interactions between features?
- Did you use external data?

  No external data was used.

  *We suggest you provide:

- a variable importance plot showing the 10-20 most important features and



Larger version included in docs\RBTFeaturePlot.png.

For the RBT the most important features were generally those that were generated from shorter time windows. In the top plot here, the order is features from: 240, 160, 80s time windows. There are approx. 550 features for each window, so features with index >~1000 are from the 80s time window.

The bottom subplot shows the mean best features (mean across CV folds). The large STD indicates large variation in feature importance across CV fold. BandsLin features appear to be the most important.

- partial plots for the 3-5 most important features

  If this is not possible, you should provide a list of the most important features.

4. Training Method(s)

- What training methods did you use?

  Polynomial SVM (quadratic) and RUS boosted tree ensemble (RBT).
- Did you ensemble the models?

  Yes
- If you did ensemble, how did you weight the different models?

  Simple mean. In some cases (where there was data drop out) the SVM produced NaN predations. For these, just the tree ensemble's predications alone were used.

5. Interesting findings

- What was the most important trick you used?
    1. My general model outperformed my subject-specific models.
    2. Combining the two models produced consistently significantly better performance than either alone.
- What do you think set you apart from others in the competition?

  I don't know what the approaches used by the top two teams were yet, but compared to the approaches I've seen posted on the forums so far:
    1. I used a general model trained on all subjects, rather than a combination single-subject models. I think this protected against overfitting and will hopefully turn out to be generally more useful.
    2. Most of the other approaches don't seem to have ensemble different model types.
    3. SVMs appear to be commonly used, but not tree ensembles. Tree ensembles appear to have a bad reputation for this sort of application, but the RUS boosted tree ensembles (designed to handle class imbalance) performed well throughout the completion.
    4. Most other approaches don't seem to have combined features extracted from different epoch window lengths, rather than used single window lengths.
- Did you find any interesting relationships in the data that don't fit in the sections above?

6. Simple Features and Methods

Many customers are happy to trade off model performance for simplicity. With this in mind:

- Is there a subset of features that would get 90-95% of your final performance? Which features? *

  I suspect the **bandsLin2D** and **bandsLinAv, summ32D** and **summ3Av**, and **mCorrsT** and **mCorrsF** feature groups would do a pretty good job without the others. The FFT is the time consuming process in producing the features, but only needs to be done once with some forward planning during feature creation.
- What model that was most important? *

  The tree ensemble scored best alone (see below). However, the SVM was quicker to train and to predict new data.
- What would the simplified model score?

  Resubmitting the best submission and non-ensembled equivalents scores as follows:

SVM alone (Master50BasicSVMGen.csv): 0.648 (public LB) and 0.679 (private LB)

RBT alone (Master50BasicRBTGen.csv): 0.715 and 0.722

SVM + RBT (Master50BasicSVMgRBTg.csv): 0.790 and 0.797.

*Try and restrict your simple model to fewer than 10 features and one training method.

Appendix

This section is for a technical audience who are trying to run your solution. Please make sure your code is well commented.

A1. Model Execution Time

Many customers care about how long the winning models take to train and generate predictions:

- What software did you use for training and prediction?

  MATLAB 2016b
- What hardware (CPUS spec, number of CPU cores, memory)?

  Core i7 4790k 4 cores running at 4.3Ghz, 32GB RAM.
- How long does it take to train your model?

  A few minutes per cv fold for the general SVM, 10-15 minutes per cv fold for the general RBT.
- How long does it take to generate predictions using your model?

  Negligible amount of time for the SVM, a few minutes for the RBT.
- How long does it take to train the simplified model (referenced in section 4)?

  With 6 cv folds, less than an hour (*not* including feature processing)
- How long does it take to generate predictions from the simplified model?

  <10 minutes

A2. Dependencies

List of all dependencies including:

- programming language/statistical tool

    - MATLAB 2016b
        - 2016b required for new string class
- libraries or packages
    - MATLAB Statistics and machine learning toolbox
    - Parallel computing toolbox
- operating system

  Windows 7 64 bit
- another other software used to generate your solution.

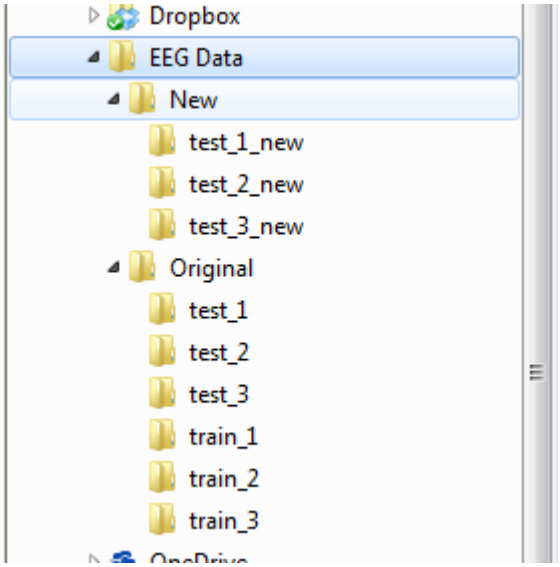
  Please include version numbers.

A3. How To Generate the Solution (aka README file)

Provide step-by-step instructions for how to create the solutions file from the code provided. Include that description here and in a separate README file to accompany the code.

Basic instructions:

1. Prepare raw data directories – should look like this



2. Open master50.m and set working directory to Master50 folder
3. Set path to raw data in master50.m on line 10 – eg S:\EEG Data
4. Click run

   See readme.md for more detailed instructions

Alternatively the trained models used to generate the 3rd place submission can be downloaded here: https://1drv.ms/u/s!AnPB0aAsVNjEk7ZpyG0M_FHimYEk9g (160 MB). Note that the seizureModel.m class needs to either be in the current directory or in the MATLAB path to load this file correctly.

A4. References

Citations to references, websites, blog posts, and external sources of information where appropriate.

Model Description

4th Place

Qingnan Tang

The frequency spectrum was: 1) divided into 50 equally spaced bands of 0.67-46.67 Hz, the power of these bands as well as their correlation matrix between channels and eigenvalues were extracted features; 2) divided into five frequency bands: delta (0.1-4Hz), theta (4-8Hz), alpha (8-12Hz), beta (12-30Hz), low-gamma (30-50Hz) (High gamma was not included as they exceed max frequency of the spectrum) , whose power and entropy were calculated as features. Gradient Boosting machine of parameter set (n_estimators=6000,max_depth=10,min_samples_leaf=5,min_samples_split=2,learning_rate=0.001,max_features=40, subsample=0.65) was used in optimal prediction.

Cross-validation was applied by a self-written algorithm. So that sequence from same hour are kept in same fold, the class labels are also stratified in each fold. One interesting finding from late-stage submission is that if data were resampled to 160Hz, and equally sized bands were extracted from 0.67~33.33 Hz, better LB score can be achieved.

Model Documentation

Name: **team Nullset (Irina Ivanenko, Oleg Panichev)**

Location: **Kyiv, Ukraine**

Email: **irinaai@protonmail.com olegxpanichev@gmail.com**

Competition: Melbourne University AES/MathWorks/NIH Seizure Prediction

1. Summary

4-6 sentences summarizing the most important aspects of your model and analysis, such as:

- The training method(s) you used (Convolutional Neural Network, XGBoost)
- The most important features
- The tool(s) you used
- How long it takes to train your model

The final model is an ensemble of seven different models. AdaBoost, Gradient Boosting, Random Forest, XGBoost, GridSearch and Voting classifier were applied for a different combination of feature sets in patient-specific approach. Each 10 minute EEG segment was divided into epochs with 30 seconds duration. These epochs were considered as separate objects that all belong to the same class as the whole 10-minute segment. Features were extracted separately for each epoch. It took from a few hours to 5 days depending on a features set to extract features for both train and test datasets. Train and prediction takes around 1.5 hours.

2. Features Selection / Engineering

- What were the most important features?
- How did you select features?
- Did you make any important feature transformations?
- Did you find any interesting interactions between features?
- Did you use external data?
- We suggest you provide:a variable importance plot showing the 10-20 most important features and partial plots for the 3-5 most important features

  If this is not possible, you should provide a list of the most important features.

The signal from each file was divided on epochs 30 seconds length without any filtration. From each epoch features were extracted. We have tried also 15 and 60 seconds epoch length but the results were worse.

We tried many features in different combinations during this competition, but not all of them were used in final models. Feature sets we've tried:

1. **Deep's kernel for features extraction (Shannon entropy, spectral edge frequency, eigenvalues of correlation matrices calculated for EEG channels and spectrums, spectral entropy for dyadic bands, Hjorth parameters).**
2. **Tony Reina's kernel for features extraction (Shannon entropy, spectral edge frequency, eigenvalues of correlation matrices calculated for EEG channels and spectrums, spectral entropy for dyadic bands, Hjorth parameters, Fractal Dimensions, Hjorth Fractal Dimension).**
3. **Correlation between all channels (120 features).**
4. **Correlation between spectras of all channels (120 features).**
5. **Spectral features version 1: total energy (sum of all elements in range 0-30 Hz), energy in delta (0-3 Hz), theta (3-8 Hz), alpha (8-14 Hz) and beta (14-30 Hz) bands, energy in delta, theta, alpha and beta bands divided by total energy, ratios between energies of all bands.**
6. **Spectral features version 2: the same as Spectral features set 1 plus low and high gamma band were used in calculation of total energy, energy in bands and ratios between energies in bands. In addition, mean energy in bands was extracted.**
7. **Spectral features version 3: power spectral density was calculated for the whole epoch. Then it was divided on 1 Hz ranges and in each range energy was calculated (30 features).**

3. Training Method(s)

- What training methods did you use?
- Did you ensemble the models?
- If you did ensemble, how did you weight the different models?

Dividing signals on epochs allowed to increase training dataset size, so total number of observations *No* was equal to

$$No = Nf * Ne,$$

where *Nf* - number of 10-minute signals, *Ne* - number of epochs per one 10-minute signal.

For cross-validation stratified K-folds with 6 folds was used. It was extremely important to use K-fold without shuffling the data, otherwise the leakage was very high and estimated performance was much higher. The leakage during

shuffling was present because two neighboring epochs with very similar parameters were often present both in train and test sets.

Each model predicted probability of epoch belongs to *preictal* class. The final probability for 10-minute signal was calculated as mean of all probabilities for epochs in this signal.

We tried both patient-specific and non-patient-specific approaches on the same model but performance was higher when patient-specific approach was used.

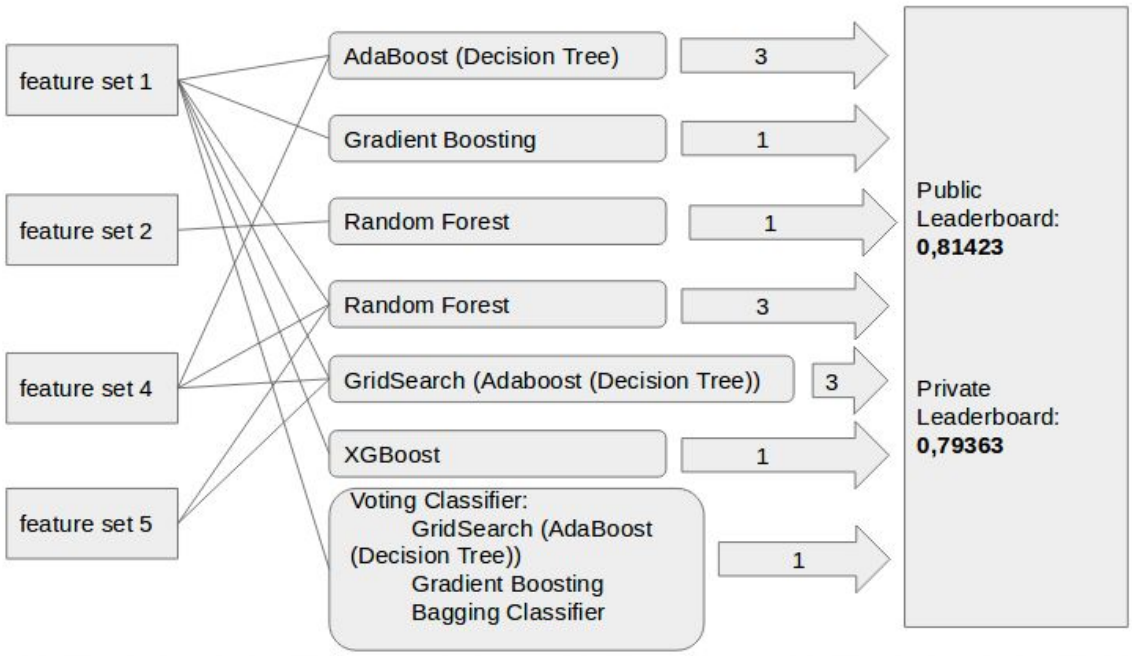
The final solution was an ensemble of best performing models (the first one is the best performing and the last one - is the worst):

1. **AdaBoost with Decision Tree base estimator with combined feature sets 1, 4 and 5 .**
2. **Gradient Boosting Classifier with feature set 1.**
3. **Random Forest Classifier with feature set 2.**
4. **Random Forest Classifier with combined feature sets 1, 4 and 5.**
5. **GridSearch for "number of estimators" parameter for AdaBoost with Decision Tree base estimator with combined feature sets 1, 4 and 5.**
6. **Voting classifier with feature set 1. Voting was performed for 3 classifiers: GridSearch for "number of estimators" parameter for AdaBoost with Decision Tree base estimator; Gradient Boosting Classifier and Bagging Classifier.**
7. **XGBoost Classifier with feature set 1.**

AdaBoost with Decision Tree base estimator with combined feature sets 1, 4 and 5 showed the highest performance among the models.

Final result *P* was calculated as follows:

$$P = 1/13 * (3*\text{Model 1} + \text{Model 2} + \text{Model 3} + 3*\text{Model 4} + 3*\text{Model 5} + \text{Model 6} + \text{Model 7})$$



4. Interesting findings

- What was the most important trick you used?

  Ensembling.
- What do you think set you apart from others in the competition?

  Ensembling of different models with different features sets.
- Did you find any interesting relationships in the data that don't fit in the sections above?

  No.

5. Simple Features and Methods

- Is there a subset of features that would get 90-95% of your final performance? Which features? *

  Maybe, but it need to be checked.
- What model that was most important? *

  All models showed approximately the same performance.
- What would the simplified model score?

  *Try and restrict your simple model to fewer than 10 features and one training method.

Appendix

A1. Model Execution Time

- What software did you use for training and prediction?

  All data analysis and models were built using Python. Libraries used: scikit-learn, pandas, xgboost.
- What hardware (CPUS spec, number of CPU cores, memory)?

  MacBook Pro mid-2015, 2.5 GHz Intel Core i7, 16GB RAM

  Asus K56, 1.7 GHz Intel Core i5-3317U CPU, 8GB RAM
- How long does it take to train your model?

  The longest part is the feature extraction. The most of feature sets may be extracted within a couple of hours for the whole dataset. Features set 2 needed around 5 days to be extracted. Train and prediction takes around 1.5 hours.
- How long does it take to generate predictions using your model?

  Haven't evaluated as well, but around 10-15 minutes.
- How long does it take to train the simplified model (referenced in section 5)?

  -
- How long does it take to generate predictions from the simplified model?

  -

A2. Dependencies

List of all dependencies including:

- programming language/statistical tool

  Python
- libraries or packages

  scikit-learn, pandas, xgboost
- operating system

  Ubuntu 16.04, MacOS Sierra
- another other software used to generate your solution.

Please include version numbers.

https://github.com/oleg-panichev/Melbourne-University-Seizure-Prediction-2016/blob/master/pip_freeze.txt

A3. How To Generate the Solution (aka README file)

Provide step-by-step instructions for how to create the solutions file from the code provided. Include that description here and in a separate README file to accompany the code.

All our code is available on github, instruction how to run it here: README

1. **First you need to extract features. Use scripts from 'features_extraction.py' and 'features_extraction_reina.py'. To run this scripts on your machine you need to change config variables (lines 639-650 in 'features_extraction.py' and lines 695-702 in 'features_extraction_reina.py'). 'features_extraction_reina.py' extracts fractal dimension features and it may take a lot of time. Extracted features are available here: https://drive.google.com/drive/folders/0B4orCuBRwwdYa1Y3YlpwcW5VWEE**
2. **After features extracted run script 'train_models.py'. This script reads train data for each patient, train all needed models, saves them into files and makes prediction on test data. Config it changing appropriate variables on lines 290-298. Fitted models are stored in 'models/' folder, generated submission files are stored in 'submissions/' folder.**
3. **After that script from 'submission_ensembling.py' should be ran to combine all submissions from models into ensemble and generate final submission file.**

A4. References

Citations to references, websites, blog posts, and external sources of information where appropriate.

https://github.com/drewabbot/kaggle-seizure-prediction/blob/master/report.md

http://www.slideshare.net/markpeng/general-tips-for-participating-kaggle-competitions

http://mlwave.com/kaggle-ensembling-guide/

https://www.youtube.com/playlist?list=PL1Vv3oB800jsEs1Lmm9beYC3McvloMm7C

Deep's kernel

Tony Reina's kernel

Model Documentation

Name:Timothée Proix

Location: Department of Neuroscience, Brown University, Providence, Rhode Island, United States of America

Email: timothee_proix@brown.edu

Competition: Melbourne University AES/MathWorks/NIH Seizure Prediction

1. Summary

In our approach, we only used features previously examined in the 2014 Kaggle seizure prediction competition (Brinkmann et al., 2016), and focused on developing different classification methods. Feature selection included spectral and temporal features, with the most important being the spectral power in different frequency bands. Classification was obtained by averaging nine different models, using different classifiers, which included Convolutional Neural Networks (CNN, LeCun et al., 1988), random forests (Breiman, 2001), Support Vector Machines (SVM, Cortes and Vapnik, 1995), Xgboost (Chen and Guestrin, 2016), and Long Short-Term Memories (LSTM, Hochreiter and Schmidhuber, 1997). Implementation was done in Python with scikit-learn (Pedregosa et al., 2011), xgboost (Chen and Guestrin, 2016) and Keras (Keras) libraries. The training of the full model typically required a few hours.

2. Features Selection / Engineering

As a first preprocessing step, we removed the drop-out parts in each iEEG segment. This step improved significantly our leaderboard (LB) score. Data were then sampled in sequential 1-min windows.

For feature selection, we used features selected by top competitors of the previous Kaggle seizure prediction competition (Brinkmann et al., 2016), including spectral and temporal features. The most important feature was the spectral power computed in eight frequency bands (0-4 Hz, 4-8 Hz, 8-12 Hz, 12-18 Hz, 18-25 Hz, 25-50 Hz, 50-80 Hz, 80-150 Hz). Other spectral features included the coherence matrix and its eigenvalues, and the spectral entropy. Temporal features included the correlation matrix and its eigenvalues, Petrosian fractal dimension (Petrosian, 1995), Hjorth mobility and complexity parameters (Hjorth, 1970), and the variance, skewness and kurtosis. In the following, we refer to these features as the “raw” features.

We obtained additional features by taking the output of a CNN and a LSTM network. The original idea was to use them directly as classifiers, however they did not perform so well directly on the classification task. Instead, we first trained them on

the raw features described above in a classification task, and then the outputs of the penultimate layer (a dense layer for the CNN network and the LSTM layer for the LSTM network) were used as additional features for the other classifiers, i.e. SVM, random forest and xgboost algorithms. These features lead to significant improvement in the LB score. Specifically:

1. the CNN network contained the following layers:

   a. a 2D convolutional layers with 16 filters and a kernel size of 3, followed by a rectified linear unit (relu) activation layer.

   b. a 2D convolutional layers with 32 filters and a kernel size of 3, followed by a relu activation layer, and a dropout layer with a rate of 0.25

   c. a dense layer with 128 units, followed by a relu activation layer, and a dropout layer with a rate of 0.5

   c. a dense layer with 2 units, followed by a sigmoid activation layer.

2. the LSTM network contained the following layers:

   a. a LSTM layer with 100 unit and a drop-out of 0.2.

   b. a dense layer with 2 units, followed by a sigmoid activation layer.

Both the CNN and the LSTM networks were trained with an Adadelta optimizer (learning rate = 0.01, rho = 0.95, Fuzz factor epsilon = 10^{-6}), over 50 epochs.

3. Training Method(s)

The prediction was given by the arithmetic average of an ensemble of nine models:

1. Two models were obtained by training a SVM with a radial basis function (rbf) kernel on the above features, CNN and LSTM outputs excepted. For the second SVM, we additionally balanced the weights (by changing the penalty term C in the SVM) according to the number of samples in each class.
2. Six models were obtained by training a random forest with 80 trees. The difference between the models is in the CNN and LSTM outputs used as additional features for training the model. Specifically, the models all used the raw features, and the CNN output, with the CNN trained on power spectrum (all bands) and:

   a. nothing else.

   b. on the correlation.

   c. on the spectral correlation (first frequency band (0-4 Hz)).

   d. on the spectral entropy (all bands).

   e. on the eigenvalues of the spectral correlation (all bands).

   f. the LSTM output, with the LSTM trained on power spectrum (all bands).

To chose those features, we trained the CNN and LSTM networks on the classification task using each of the raw features, and ranked their performances as classifiers on the LB. The raw features that performed less well when used as input in the CNN or LSTM network were discarded.

3. One model was obtained by training a xgboost algorithm. Specifically, we performed a parameter search over several xgboost parameters (maximum depth of a tree, learning rate, L1 regularization term on weights, subsample ratio of columns when constructing each tree, subsample ratio of the training instance, and the number of estimators) with 200 samplings. The 10 best models were chosen and the arithmetic average of their predictions was computed.

4. Interesting findings

Three different tricks may have help us to obtain a good score in this competition:

1. Preprocessing the data to remove drop-out parts.
2. Using LSTM and CNN outputs as additional features for our classifiers.
3. Using very different methods for the final ensemble that prevented overfitting. The ranking order changed significantly between the private and the public leaderboard, as it seemed that many teams largely overfitted the public dataset (top score going from 0.854 on the public LB to 0.807 on the private LB). However, our public and private LB score were almost equal (0.804 to 0.791). We believe using ensembling of completely different classifiers help us a lot in preventing overfitting.

5. Simple Features and Methods

The most important feature was the power spectrum in different frequency bands, as well as the output of the CNN using the power spectrum as an input feature. Our best models in isolation (before ensembling) were the random forest models, with the best model performing 0.778 on the private LB.

Appendix

A1. Model Execution Time

Training was performed using scikit-learn for random forest and SVM, xgboost, and Keras on top of Tensorflow for CNN and LSTM networks. We used a computer with 32 cores, 190 GB of RAM, and a K40 NVIDIA GPU. To train the model, xgboost and

random forests were typically completing rapidly (around 2 minutes for all patients, multithreading with 30 CPUs for Xgboost), SVMs would take up to two hours, while CNNs and LSTMs could require a few hours to train correctly (on K40 GPUs). Generating predictions, however, is fast, taking only a few minutes to complete. A simplified model using only an ensemble of random forests would be fast to train and to generate predictions (few minutes). However, if the CNN features are included, a few additional hours are necessary to train the CNN.

A2. Dependencies

python 3.5.2; h5py 2.6.0; numpy 1.11.1; scipy 0.18.0; scikit-learn 0.18.1; tensorflow 0.9.0; keras 1.1.2.

OS: Ubuntu 14.04.

A3. How To Generate the Solution (aka README file)

- put data in respectively test_1_new, test_2_new or test_3_new directory according to patient ID.

- run preprocessing.py; removes the artifacts. The output is saved in “processed” directory. Each test matrix generates a processed file. Two output files containing the indices and the name of the test matrices that could not be read are also computed.

- run feature_extraction.py; extracts features from the time series. Features are saved in “features” directory. One file by feature and by patient is computed.

- run final_cnn.py; obtains CNN features as the output of the penultimate layer of a CNN. Features are saved in features directory. One file by feature and by patient is computed.

- run final_lstm.py; obtains LSTM features as the output of the penultimate layer of a LSTM. Features are saved in features directory. One file by feature and by patient is computed.

- run final_pred.py; uses all these features to make different predictions using SVM, random forest and xgboost. The output of each prediction is saved in avg_sub directory.

- run bag_and_add_unread.py; performs the averaging and adds arbitrary decision for the unread files.

subs.csv is the submission file.

A4. References

Breiman, L. (2001). Random forests. Machine Learning *45*, 5–32.

Brinkmann, B.H., Wagenaar, J., Abbot, D., Adkins, P., Bosshard, S.C., Chen, M., Tieng, Q.M., He, J., Muñoz-Almaraz, F.J., Botella-Rocamora, P., et al. (2016). Crowdsourcing reproducible seizure forecasting in human and canine epilepsy. Brain *139*, 1713–1722.

Chen, T., and Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, (ACM), pp. 785–794.

Cortes, C., and Vapnik, V. (1995). Support-vector networks. Machine Learning *20*, 273–297.

Hochreiter, S., and Schmidhuber, J. (1997). Long Short-Term Memory. Neural Computation *9*, 1735–1780.

Keras https://github.com/fchollet/keras.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1988). Gradient-based learning applied to document recognition. In Proceedings of the IEEE, pp. 2278–2324.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research *12*, 2825–2830.

Model Documentation

Name: Michal Nahlik

Location: Prague, Czech Republic

Email: nahlik.michal@seznam.cz

Competition: Melbourne University AES/MathWorks/NIH Seizure Prediction

1. Summary

The solution is based on classification decision tree models created for each channel and patient using various statistical variables calculated on the whole 10 minute data files. The most important features included correlation of power spectrums at dyadic levels, maximum correlation between channels in interval ±0.5 seconds, correlation between channels in frequency domain, spectral edge and parameter estimates of fractional Brownian motion. Training models for all patients on provided data set took less than a minute. Solution was coded in MATLAB 2014a (MathWorks Inc, Natick MA) using Statistical and Machine learning toolbox.

2. Features Selection / Engineering

Most of the features come from the sample solution based upon team Drew Abbot's winning solution from the previous Kaggle seizure prediction competition and Tony Reina's feature extractor kernel published at this competition.

As the wavelet decomposition is often used when working with EEG recordings I tested different types of waves and scales of transformations. 10 scale wavelet transformation using Morlet wave seemed to provide the best improvement of my solution. Singular value decomposition was then used to extract information from the wavelet transformation output.

For the final model following features were calculated for each channel on the whole 10 minute data files with data dropouts removed: mean value, standard deviation, spectral edge at 50% power below 40 Hz, skewness, kurtosis, Hjorth parameters, Shannon's entropy across energies in each frequency band (0.1-4 Hz,4-8 Hz,8-14 Hz,14-32 Hz,32-70 Hz,70-180 Hz) and at each dyadic level, maximum correlation between channels in interval ±0.5 seconds, correlation between channels in frequency domain and between channel power spectrums at dyadic levels, Petrosian

and Katz fractal dimension, parameter estimation of fractional Brownian motion and singular values of 10 scale wavelet transformation using Morlet wave.

No external data or further feature transformations were used.

The most important features included spectral edge, all types of correlations, Shannon's entropy of dyadic levels and parameter estimates of fractional Brownian motion.

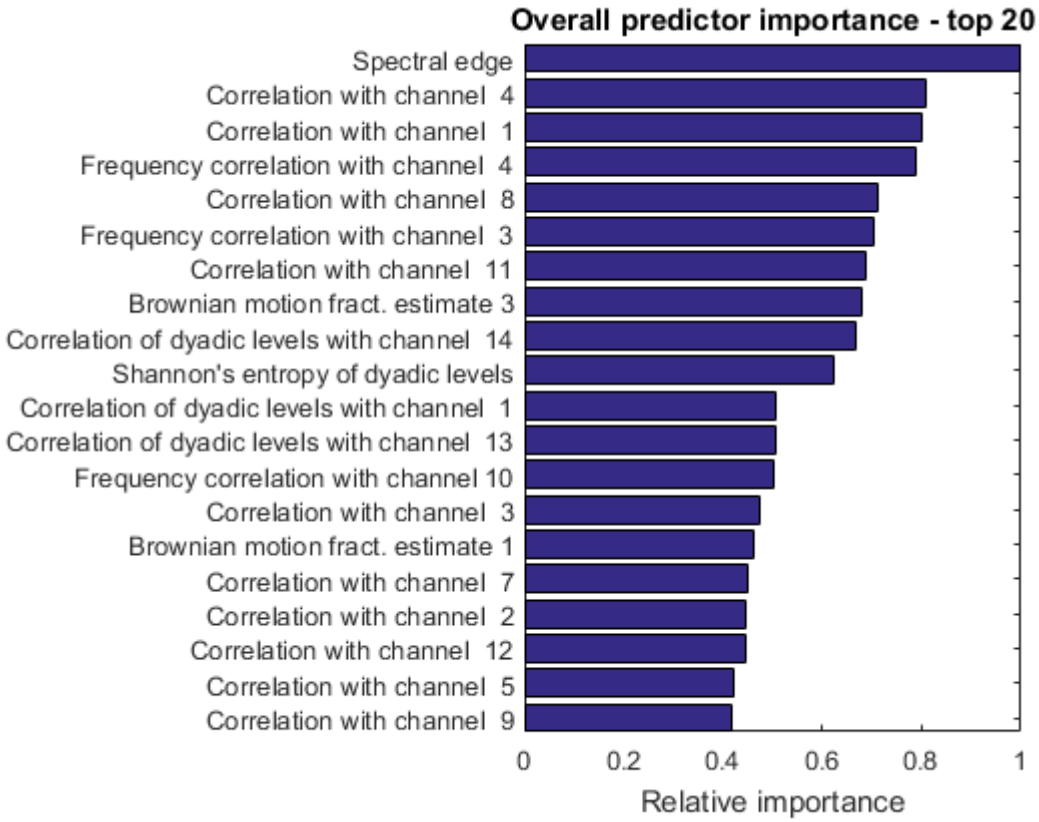


Figure 1 20 most important predictors across all models

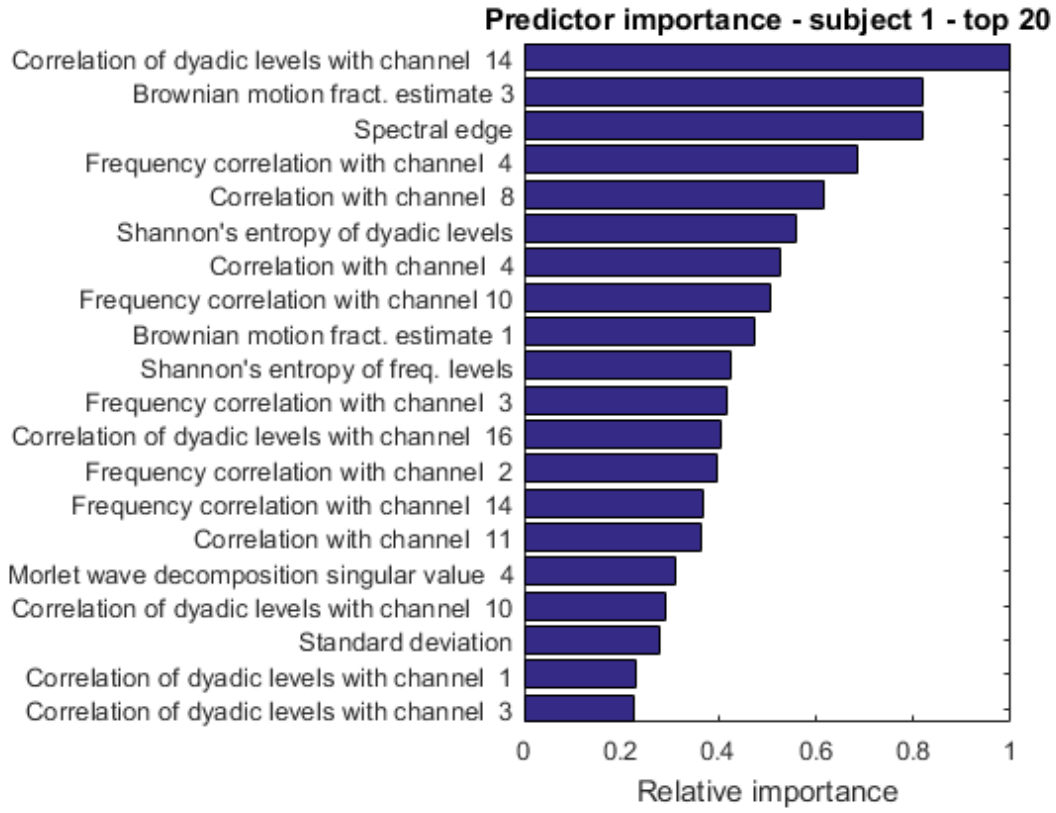


Figure 2 20 most important predictors for subject 1 models

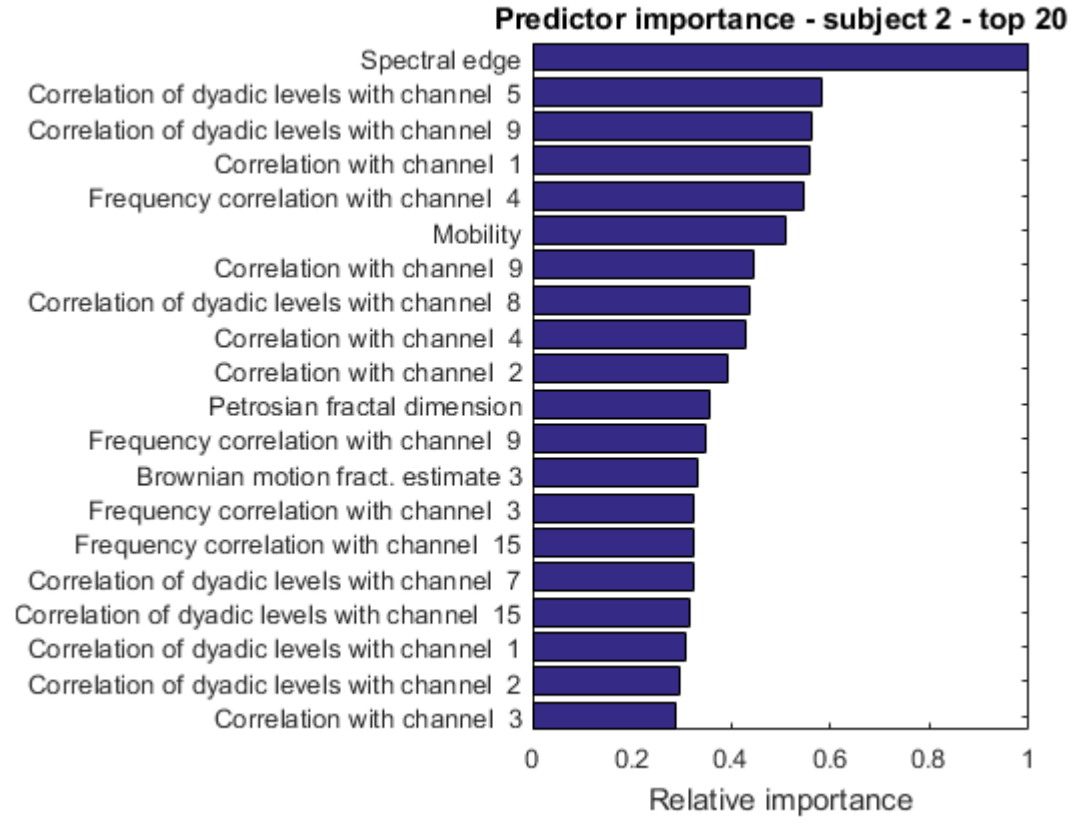


Figure 3 20 most imporant predictors for subject 2 models

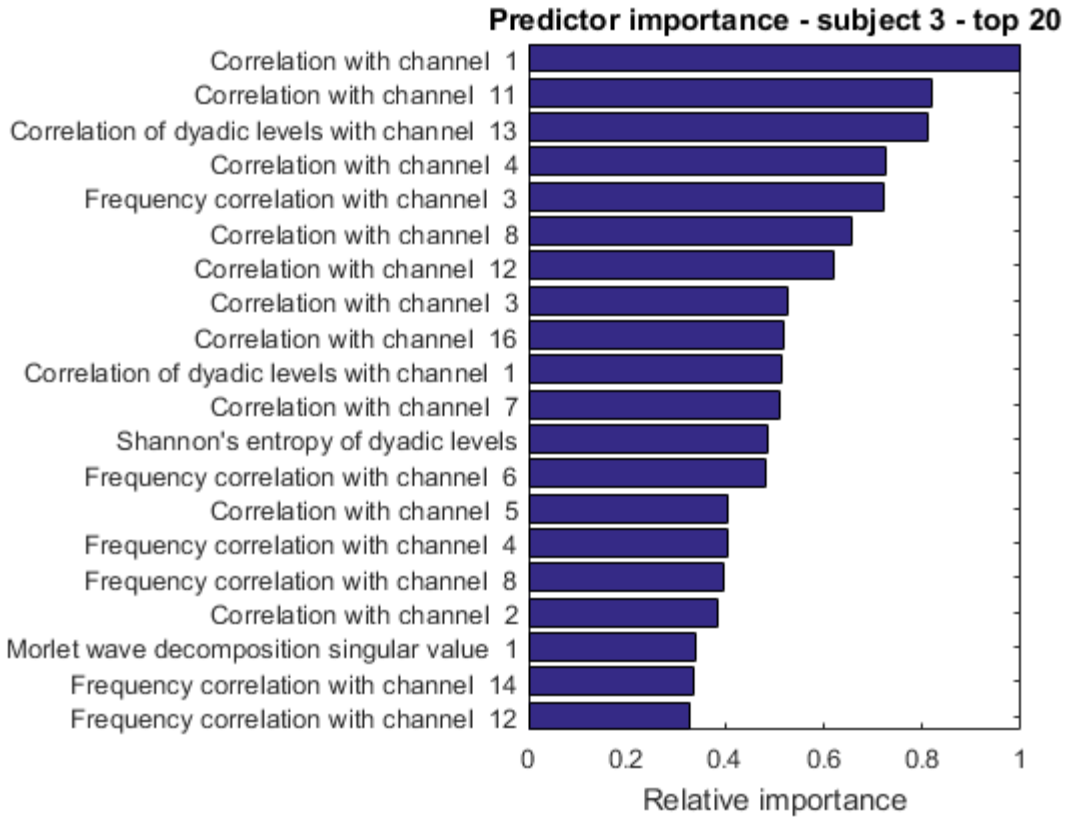


Figure 4 20 most imporant predictors for subject 3 models

3. Training Method(s)

For each channel and patient 10 classification decision trees were trained in 10-fold cross validation using the Exact search training algorithm (see Matlab documentation of *fitctree* for more details). The final classification of a file was calculated as the mean output across channel models for the patient

4. Interesting findings

I tried to make both subject specific models and general model as the general model might be more useful. Both approaches actually reached the same scores on the leaderboard, which is probably caused by the fact that the general model was trained on all subjects on which it was then tested. That might allow decision trees to grow different branches for each subject at some level and thus reaching the same score as subject specific models. When I tried to cross validate general model by subject (2 subjects used for training, 1 subject for testing) the results got much worse, around 0.6 AUC. I guess signal/feature normalization would have to be done to make the general model work for unseen subjects.

5. Simple Features and Methods

I tried two simplifications: first using only 10 most important features of each subject specific model and second using only overall top 10 features.

Model using 10 most important features of each subject specific model (see figures 2-4) reached 0.77225 on private leaderboard and 0.79406 on public leaderboard.

Model using only overall top 10 features (see figure 1) scored 0.77467 on private leaderboard and 0.73941 on public leaderboard.

Appendix

A1. Model Execution Time

Software: Windows 7 Professional 64-bit, MATLAB 2014a (MathWorks Inc, Natick MA) with Statistical and Machine learning toolbox

Hardware: Dell Vostro 3460 (Intel i7-3612QM 2.1GHz (4 cores), 8GB RAM)

Training of all models on the given data set takes around 50 seconds.

Generating predictions for all test files takes less than 2 seconds.

Generating features for a single file takes around 1.2 seconds.

Training of the simplified models takes around 20 seconds.

Generating predictions with the simplified model takes the same time as with the full feature model (less than 2 seconds).

The main run time issue is reading data from HDD and generating features.

A2. Dependencies

Matlab 2014a (tested also with Matlab 2015b) with Statistics and Machine Learning Toolbox

A3. How To Generate the Solution (aka README file)

Preparation

- Download the source code from: https://github.com/nahlik-michal/kaggle-melbourne-seizure-prediction
- Download and unpack data, train_and_test_data_labels_safe.csv and sample_submission.csv
- Change file/folder paths in *settings.m*

Feature extraction

- execute */features/run_generate_features.m*
- copy features from the old (leaked) test files into the training feature folders (test_1 to train_1, ...)

Running the model

- Execute */main/run_dt.m* to load the features, cross validate the solution and create submission. Model for each subject and channel will be created and if *opts.save_model* in *settings.m* is set to true, the models used for creating submission will be saved to *opts.modelDir*.
- */main/run_trained_dt.m* can be used to load the saved models and create submission for the testing data.
- */main/run_dt_general_model.m* loads data for all subjects and creates a general model (across subjects) for each channel, cross validates the solution and creates submission. This solution is cross validated the standart way and then by subject (2 subjects for training, 1 subject for testing).

A4. References

Brinkmann, B. H., Wagenaar, J., Abbot, D., Adkins, P., Bosshard, S. C., Chen, M., ... & Pardo, J. (2016). Crowdsourcing reproducible seizure forecasting in human and canine epilepsy. Brain, 139(6), 1713-1722.

Reina, Tony. "Feature Extractor Matlab2python Translated." Melbourne University AES/MathWorks/NIH Seizure Prediction Kernel, Feature Extractor Matlab2python Translated | Kaggle., 30 Nov. 2016. Web. 23 May 2017. <https://www.kaggle.com/treina/feature-extractor-matlab2python-translated>.

Model Documentation

Team: Chipicito+Solverworld
Solverworld:
Name: Daniel Grunberg
Location: Massachusetts
Email: dgrunberg@solverworld.com
Competition: Melbourne University AES/MathWorks/NIH Seizure Prediction

1. Summary

Our solution was created by merging 2 solutions (Algorithm 1 and Algorithm 2) through rank averaging with specified weights. Rank averaging mean converting the predicted probabilities (0 to 1) to a rank (i.e. 1st, 2nd, etc.), then averaging those ranks with the specified weights. The weighted ranks were then converted back to evenly spaced probabilities, i.e. rank r becomes probability r/N, where N is the total number of predictions. Because the figure of merit for scoring is area-under-curve (AUC), the specific probabilities do not matter, only their order.
The 2 algorithms were combined through rank averaging with weights 4 (Algorithm 1) and 1 (Algorithm 2)

Algorithm 1 (Solverworld): iEEG data was windowed into 60 sec non-overlapping segments. Features were the total energy in the FFT in bands demarcated at 1,2,4,8,16,32 and 64 Hz, plus the ratio of the total energy in band 10-25 Hz to 0.1-10Hz. Two models were used: (1A) an XGBOOST boosted-tree model with 160 rounds, and (1B) a 600-tree random forest. Each model used a 7-fold validation and prediction strategy, with models generated separately for each patient. Results from (1A) and (1B) were combined in the ratio (3:1).
The data has a number of dropouts, which are times when all 16 channels are identically zero. The data was prepped by first removing all short sequences of valid data (where less than 400 valid samples were surrounded by invalid data). Then all the valid data was moved to the start of the sequence, with the zeros placed at the end. This is to keep all sequences the same length, which helps with the windowing. The data was then resampled to 80 Hz, or 16x48000 samples per iEEG.
The most time consuming part of training was the feature generation. This took several hours on a desktop i7, without parallelization. The training/model generation for XGBoost and Random Forest runs fairly quickly – just a few minutes. The prediction runs even faster, less than 60 seconds.

Algorithm 2 (Chipicito): iEEG data from the samples were divided into 15-second sub-segments. STFT without windowing was performed, and absolute values were selected at approximately 2 to 50 Hz based on previously-reported findings [3] and averaged down to 126 frequency groups. Dropouts were removed, and the 10-minute segments were represented by <=40 sub-segments for training. The 16 channels X 126 frequency groups were the data inputs into a 3-Layer convolutional

net (ConvNet) adapted from [4] that included a second convolutional layer. The probability of preictal was the frequency of sub-segments predicted to be preictal.

2. Features Selection / Engineering

Algorithm 1:
A large number of features were tried before landing on the ones used in the solution. These included

- Additional frequency bands
- Standard deviation of each channel
- 3 Hjorth energy values (h1,h2,h3)
- Wavelet coefficients (db4)
- Autoregressive model errors (AR models with orders 7,8 and 9)
- Correlation coefficients (off-diagonal terms)
- Absolute value of eigenvalues of the Correlation coefficient matrix
- Correlation coefficients of FFT of the samples

In addition, a variety of window and overlap sizes were considered, from 2 seconds to 60 seconds.

In addition, because the windowed data was a multiplication of training samples (10x for 60s windows, for example), a method had to be used to go from the predictions on each window to the prediction for each original sample. We tried a variety of Lp normal “collapse” functions. For example, L1 (average of the predictions across samples), L2 (L2 norm), L4, L-infinity (max value), etc.

Within each set of features and windows, there are a few hyper-parameters that needed to be searched for each training method. For example, for XGBoost, one needs to choose rounds (trees), learning rate, tree depth, and so on.

Algorithm 2:
The features used were derived from 15-second sub-segments, although 5- and 30-second sub-segments were also attempted. STFT without windowing was performed, and absolute values were selected at approximately 2 to 50 Hz based on previously-reported findings [3] and averaged down to 126 frequency groups.

Choosing best features, windows, and collapse method:

Algorithm 1:
We standardized on 7-Fold cross validation strategy in order to compare methods. We tried to avoid using public leaderboard scores to pick features or methods as much as possible, as that leads to overfitting to the leaderboard.

In addition, to avoid over-fitting, we did not want to just pick the “best” features out of a large number, as random features could bubble to the top as best – per Section

7.10 in [Hastie]. So we tried combinations of groups of related features, not individual features.

We also compared per-patient models to a model that was uniform across all 3 patients.

Using these ideas, we landed on just energy in a set of frequency bands and the ratio of energy in 2 ranges, as specified in the summary. Features were the total energy in the FFT in bands demarcated at 1,2,4,8,16,32 and 64 Hz, plus the ratio of the total energy in band 10-25 Hz to 0.1-10Hz.

Algorithm 2:
Cross validation was achieved by randomly selecting a 1:1 ratio of the 60-minute groups. Other combinations of the data were attempted, ie. 2D data structures of numChannels x numTimeChunks and numFrequencyBuckets X numTimeChunks, with the remaining dimension being used to calculate the probability of preictal.

In the ConvNet architecture used first layer of convolutional filters were only 1x1 (so essentially just feature weights) and the second layer of filters as approximately one half the size of the input layer. Sets of 10 filters were used per layer.

Other machine learning techniques were attempted, including self-taught learning and support vector machines. Also different hyperparameters of the ConvNet were attempted, as were different numbers of convolutional layers, and different sizes of the intermediate convolutional layer.

In neither algorithm did not use any external data, although algorithm 1 did attempt to use dog data for the unsupervised portion of self-taught learning (STL).

3. Training Method(s)

Algorithm 1:
We finalized on just XGBoost and Random Forest models, which were combined in a ratio of 3:1. The ratios were determined by the relative performance of the 2 models in cross-validation and the public leaderboard.
Algorithm 2:
Stochastic gradient descent was used to train the ConvNet. Training was performed either per patient or across all 3 patients. In the end, training was performed across all three patients.

4. Interesting findings

Algorithms 1 and 2:
We tried hard to not overfit to the leaderboard, although we are not sure how successful that was, as we did drop a number of places from public leaderboard to private leaderboard (although less than others).

For **Algoritm 1,** I tried retraining several times the exact architecture I used to arrive at my highest public leaderboard score, but was not able to improve or even approximate the highest score, leading me to conclude that the random combination of initialized parameters and selection of training subset lead to an element of "luck" in the results I achieved.

5. Simple Features and Methods

Algorithm 1:
I think we got our features down to a very small number already. The XGBoost model was the best performing of the various ones we tried, including Logistic Regression and Random Forest.
Algorithm 2:
Various subsets of the 2 to 50 Hz range were attempted, as were both lower and higher frequency ranges.

Appendix

A1. Model Implementation and Execution Time

Algorithm 1:
We used Python with standard packages numpy and scipy. The only additional package installed was xgboost. The CPU used was a i7 8-core processor with 64 GB of RAM, although the models involved did not use that much memory (probably less than 4GB total needed). XGBoost is very good at using multiple cores during training, so that helped speed things up.

It took approximately 8 hours to generate features on the training and test sets, but the training (model generation) and prediction were much faster, on the order of minutes. This was due to the conversion of the EEG data into npz (native numpy matrix format, the downsampling to 80 Hz, and the slowness of some of the feature calculations).

Anaconda 4.2.0 was used to set up the development environment. This included:

- Scipy 0.18.1
- Numpy 1.11.1
- Pandas 0.18.1
- Sklearn 0.17.1 (later upgraded to 0.18.1)

XGBoost was installed from source (v0.6)

Algorithm 2:
Initially Octave 4.0.0 was used, and later the algorithm was migrated over to Matlab R2016b. Preprocessing took about 30 to 60 minutes, and training typically took

about 20 to 30 minutes when using SGD with the ConvNet. The CPU was an i7 (Sandybridge), using 16 GB of RAM on a Dell N5110 Laptop.

A2. References

[1] Mark J Cook, et al, *Prediction of seizure likelihood with a long-term, implanted seizure advisory system in patients with drug-resistant epilepsy: a first-in-man study*, The Lancet/neurology Vol 12, June 2013.

[2] Trevor Hastie, et al, *The Elements of Statistical Learning*, Springer 2009.

[3] Florian Mormann and John G. R. Jefferys. Neuronal firing in human epileptic cortex: the ins and outs of synchrony during seizures. *Epilepsy Curr.* 2013; 13: 100–102.

[4] Convolutional Neural Network. UFLDL Tutorial. Available at: http://ufldl.stanford.edu/tutorial/supervised/ConvolutionalNeuralNetwork/. Accessed: 15 May 2017.