Seizure detection in EEG time series

Eben Olson and Damien Mingle

September 1, 2014

1 Introduction

We describe here the methods used in preparing our submission to the UPenn and Mayo Clinic Seizure Detection Challenge, which obtained second place on the private leaderboard. We present in detail our final and most sucessful approach, as well as an overview of less sucessful experiments which also contributed to our final ensemble or provided some insight. It is assumed that the reader is familiar with the structure of the challenge and the data, described at http://www.kaggle.com/c/seizure-detection.

2 Early approaches

2.1 Spectrograms

Our initial feature extraction method calculated spectrograms of each EEG trace, in an attempt to capture both frequency content and temporal dynamics. Each clip was first resampled to 500Hz, and the short-time Fourier transform was applied, discaring phase information. Spectrograms were flattened into vectors, and mean subtraction and normalization was applied on a per subject and per feature basis. Features from each channel were concatenated, and logistic regression or random forests were used for classification. Our best (ensembled) submission with these features scored 0.94081 on the public leaderboard.

2.2 Scattering coefficients

As an alternative to spectrograms, we attempted to use scattering coefficients[2], a framework for time-frequency analysis which has been shown to give good results in audio classification tasks[1]. We used the ScatNet MATLAB toolbox[1] to compute scattering coefficients for each clip, after resampling to 500Hz. Coefficients for each channel were concatenated and logistic regression was used for classification. Only a marginal improvement (0.94212 public leaderboard) was seen over spectrogram features.

2.3 Maximal cross-correlation

We next considered maximal cross-correlation, which has been reported to produce useful features for detection of epileptic EEG activity[3]. This method

[1]`http://www.di.ens.fr/data/software/`

attempts to compensate for propagation delays of brain activity by computing cross-correlation between channels at various lag times and taking only the maximum value, normalized by the channel autocorrelation. We obtained a substantially worse score (0.86761 public leaderboard) with this method. However, review of the code indicated that this may have been due to a bug in the feature calculation, and further investigation of this method may be valuable.

3 Final approach

3.1 Feature extraction

Our final approach to feature extraction calculated the covariance matrix of the EEG data, in order to capture correlations between channels. Since seizure activity is characterized by increased long-range synchronization of neuronal activity, this was expected to produce informative features. Matrices were individually normalized to zero mean and unit variance.

As frequency analysis had been shown to be valuable, rather than compute a single covariance matrix we first filtered each trace with several bandpass filters. We initially applied four filters covering the range 1-200Hz. Filter choice presents a complicated trade-off between frequency selectivity, signal to noise ratio, and output dimensionality. Performance was evaluated by cross validation of logistic regression predictors.

While attempting to manually optimize the filter parameters, we found that filters chosen for one subject could perform extremely poorly on others. We therefore performed an automated filter selection step, in which combinations of up to four filters were evaluated on each subject. These filters were chosen from a bank of 10 partially overlapping, approximately log-spaced bandpass filters covering the range 5-200Hz. The three combinations which gave the highest CV values were retained.

3.2 Neural network classification

As an alternative classification strategy, we experimented with the use of multilayered neural networks. Our initial motivation was the possibility of learning a cross-subject mapping which would allow our model to use the full training set to improve its predictions. While this goal was not realized, we did find that the NN models provided a boost over logistic regression. Our software was based on dnn.py[2], a recently released demonstration of a deep neural network written in Python. This provided an excellent

We tested a number of network architectures, but found that a network with two hidden layers of 200 and 100 units respectively gave good results while being reasonably quick to train. Rectified linear units were used in the hidden layers and logistic regession in the output layer. Dropout of 0.5 was used in the hidden layers for regularization. All networks were trained with the adadelta method for 100 epochs. Multiple networks were trained for each subject and filter combination. In an attempt both to increase diversity and to reduce the impact of dissimilar electrode patterns across subjects, each network was trained on a 12-channel subset of the full covariance matrix. We found that depending

[2] `https://gist.github.com/SnippyHolloW/8a0f820261926e2f41cc`

on network architecture, predictions would become extremely compressed into the neighborhoods of zero and one. To avoid potential issues with numerical precision, we applied a logarithmic rescaling to predictions in the $(0, 0.1]$ and $[0.9, 1)$ ranges.

3.3 Early seizure prediction

Our best scores were obtained by submitting the same values for p_{early} and $p_{seizure}$, rather than trying to train separate classifiers for early ictal events. This phenomenon was reported early in the competition by the user Alexandre[3]. We observed a similar trend in our cross-validation testing, and believe it is explained by the combination of the AUC metric and the imbalanced classes of the data set, which leads to a much larger penalty for false negatives than false positives. At least for the classification strategies we employed, the error due to training on the "wrong" labels was outweighed by the benefits of a larger training set.

However, post-deadline testing showed that our error on the early detection task was several times higher than on the seizure detection task. Improvements could potentially be obtained by fine-tuning of the trained networks with early-ictal data alone, or by applying other techniques for learning from noisy labels.

4 Ensembling

5 Technical details

5.1 Software

All of our experiments, with the exception of scattering coefficient extraction, were carried out using Python and IPython notebooks. The `numpy`, `scipy`, and `scikit-learn` packages were used extensively. Theano[4] is required by the neural network module. Our repository, available at `https://github.com/ebenolson/seizure-detection`, contains a full list of required packages and instructions for running our example code. We will also give a brief description here:

- `Data Preparation.ipynb`
  This notebook loads all clips for each subject, applies bandpass filters and calculates covariance matrices, then performs the filter selection step. The filtered covariance matrices are saved, along with the clip labels and filenames, to a pickle file in the `data` subdirectory.

- `Train Classifiers and Predict.ipynb`
  This notebook loads the preprocessed data and trains multiple networks on each input file. The predictions of each network are saved to the `output` subdirectory.

[3]`http://www.kaggle.com/c/seizure-detection/forums/t/9382/max-achieveable-auc/48642#post48642`

[4]`https://github.com/Theano/Theano`

- `Postprocessing.ipynb`
  This notebook loads predictions of the trained networks and combines them to produce `submission.csv`, a submission in the required format.

- `Miscellaneous.ipynb`
  This notebook contains code snippets implementing some of our earlier approaches.

- `simplednn/`
  This subdirectory contains the neural network module.

5.2 Hardware and runtime

Most computations were done using Amazon EC2 m3.2xlarge instances with 8 virtual cores and 30GB RAM. Using spot requests, EC2 provides an affordable platform which can be easily scaled and parallelized depending on the memory and processing power required. Some experiments were also done using a Core i5 quad core desktop with 16GB of RAM. On the desktop, preprocessing the full data set requires approximately one hour and training one iteration of networks (3 per subject) requires 13 minutes. The time required to generate predictions is negligable.

References

[1] Joakim Andén and Stéphane Mallat. Multiscale scattering for audio classification. 2011.

[2] J. Bruna and S. Mallat. Classification with scattering operators. In *2011 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1561–1566, June 2011.

[3] P.W. Mirowski, Yann LeCun, D. Madhavan, and R. Kuzniecky. Comparing SVM and convolutional networks for epileptic seizure prediction from intracranial EEG. In *IEEE Workshop on Machine Learning for Signal Processing, 2008. MLSP 2008*, pages 244–249, October 2008.