Seizure Detection

April 20, 2018

1 Introduction

For individuals with drug-resistant epilepsy, responsive neurostimulation systems hold promise for augmenting current therapies and transforming epilepsy care.

Of the more than two million Americans who suffer from recurrent, spontaneous epileptic seizures, 500,000 continue to experience seizures despite multiple attempts to control the seizures with medication. For these patients responsive neurostimulation represents a possible therapy capable of aborting seizures before they affect a patient's normal activities. In order for a responsive neurostimulation device to successfully stop seizures, a seizure must be detected and electrical stimulation applied as early as possible. A seizure that builds and generalizes beyond its area of origin will be very difficult to abort via neurostimulation. Current seizure detection algorithms in commercial responsive neurostimulation devices are tuned to be hypersensitive, and their high false positive rate results in unnecessary stimulation.

In addition, physicians and researchers working in epilepsy must often review large quantities of continuous EEG data to identify seizures, which in some patients may be quite subtle. Automated algorithms to detect seizures in large EEG datasets with low false positive and false negative rates would greatly assist clinical care and basic research.

In this project you will be given datasets from patients with epilepsy undergoing intracranial EEG monitoring to identify a region of brain that can be resected to prevent future seizures are included in the contest. These datasets have varying numbers of electrodes and are sampled at 5000 Hz, with recorded voltages referenced to an electrode outside the brain.

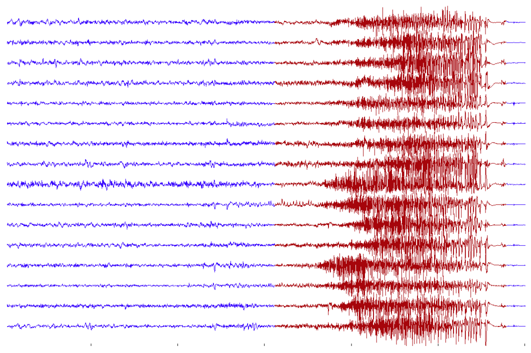

Figure 1: Intracranial EEG recorded from a dog with naturally occurring epilepsy using an ambulatory monitoring system. EEG was sampled from 16 electrodes at 400 Hz, and recorded voltages were referenced to the group average.

Since the number of electrodes and the recording conditions differ across patients, you will train a different classifier for each patient.

2 Data Description

Data are organized in folders for each human subject. The data is organized into 1-second EEG clips labelled "Ictal" for seizure data segments, or "Interictal" for non-seizure data segments. Ictal data segments are provided covering the entire seizure, while interictal data segments are provided covering approximately the mean seizure duration for each subject. Starting points for the interictal data segments were chosen randomly from the full data record, with the restriction that no interictal segment be less than one hour before or after a seizure. Within folders data segments are stored in matlab .mat files, arranged in a data structure with fields as follow:

- **data**: a matrix of EEG sample values arranged *row* $\times$ *column* as *electrode* $\times$ *time*.
- **data_length_sec**: the time duration of each data row (1 second for all data in this case).
- **latency**: the time in seconds between the expert-marked seizure onset and the first data point in the data segment (in ictal training segments

only).

- **sampling_frequency**: the number of data samples representing 1 second of EEG data. (Non-integer values represent an average across the full data record and may reflect missing EEG samples).
- **channels**: a list of electrode names corresponding to the rows in the data field.

The human data are from patients with temporal and extratemporal lobe epilepsy undergoing evaluation for epilepsy surgery. The iEEG recordings are from depth electrodes implanted along anterior-posterior axis of hippocampus, and from subdural electrode grids in various locations. Data sampling rate is 5,000 Hz.

- ictal_segment_N.mat - the Nth seizure training data segment
- interictal_segment_N.mat - the Nth non-seizure training data segment

Additional annotated intracranial EEG data is freely available at the International Epilepsy Electrophysiology Portal, jointly developed by the University of Pennsylvania and the Mayo Clinic.

Since you do not have different validation and test data you should divide your data into training, validation and test data.

Since the data are from intra-cranial recordings there is no need for data cleaning and data preprocessing.

Download all project files in: `http://www.telecom.tuc.gr/patreco/res/projects/07_Epilepsy/`

2.1 Features

People who have worked in this problem have used the following features:

- fractal dimension
- mobility
- complexity
- skewness
- kurotsis

- variance
- frequency energy at following bands: delta(0.5-4Hz), theta(4-7Hz), alpha(7-14Hz), beta(14-30Hz), gamma(30-100Hz)

An alternative is the following set of features [2]:

- FFT 1-47Hz
- correlation coefficients between channels, or
- non-linear interdependence between channels [1]. The code of this measure is provided to you.

Alternative features are described in [6]

3 Classifiers

A good choice is the Random Forest classifier (3000 trees), which exists in scikit-learn. Another choice is the support vector machines.

4 Bonus

The following are optional and count for extra grade in the project.

4.1 Spike Sorting

Implement the Self Organizing Maps and try to collect the ictal data into clusters. Alternatively you may implement the Superparamagnetic Clustering [4]. Note that due to high dimensionality of the feature vector, clustering methods such as the k-means do not work.

4.2 CNN classifier

You will try to do the classification with a deep convolutional neural network (CNN). The input to CNN will be spectrograms calculated from short time Fourier transform. A Matlab/Octave script, *createTimeFrequency.m*, is provided to help you calculate the spectrograms from the signal files. You will need to use the Time Frequency Toolbox[1]. This library, *tftb-0.2.tar.gz*, together with the manual is included in the project support software. The spectrograms are images of size $freq\_bins \times time\_bins \times num\_channels = 128 \times 128 \times num\_channels$. The frequency range is from 0 to $\frac{num\_freq\_bins}{NFFT} \times \frac{Fs}{2} = \frac{128}{1024} \times \frac{5000}{2} = 312.5Hz$. The time range is from 0 to 1. The images are created with Matlab/Octave and are used in Python. Note that python reads the raw data as the transposed matrix $num\_channels \times time\_bins \times freq\_bins$. Together with the images you should have label lists where the label of *Interictal* is 0 and the label of *Ictal* is 1.

In order to improve the classification score the images should be presented to the classifier in randomized order. You may do this by collecting all images of all train utterances in one huge list L. Then at the start of each new epoch you must shuffle (randomly permute the elements of) the list. Note that the corresponding label list should also be shuffled the same way so that to preserve the labelling of images. There is no need to shuffle the validation images and the test images (and the corresponding labels).

You may implement the CNN in any language and library you want. However, in order to help you, part of a working TensorFlow code is provided. You may complete this code according to the following instructions:

- Read each *train* image of shape $128 \times 128 \times num\_channels$ and append it to a *train_params* list.
- Read each *valid* (validation) image of shape $128 \times 128 \times num\_channels$ and append it to a *valid_params* list.
- The input to the CNN should be a batch of $batch\_size = 64$ images and the corresponding labels.
- Let $n\_elements$ be the number of images in train list L, then $n\_batches = \lfloor n\_element/batch\_size \rfloor$ is the number of batches. We do not consider possible remaining images ($remainder = n\_elements \mod batch\_size$).
- Similarly for the validation list.
- When you define the computation graph the image input should be a *tf.placeholder* of type *ft.float32* and shape ($batch\_size \times height \times width \times channels$), where $batch\_size = 64$, $height = 128$, $width =$

[1]http://tftb.nongnu.org/

128 and $channels = num\_channels$. The input to labels should be a *tf.placeholder* of type *ft.int32* and shape (*batch_size*,).

- There should be separate place-holders for train and validation data.
- When you evaluate the computation graph through a session you should iterate through all train batches and feed each one (together with the batch of corresponding labels) into the network using a feeding dictionary (*feed_dict*). An iteration through all data is called an epoch.
- The same when you evaluate the validation part of the graph.
- Use the validation score to decide if the and you should stop the iterations.

4.2.1 Network Architecture

You will define the network architecture in *inference* function. Useful references are the work of Qian et al. [3] and the work of Schirrmeister et al. [5].

- Define a few blocks consisting of one or more convolutional layers, followed by a max pooling layer.
- All convolutions will use 3×3 filters and $padding =' SAME'$.
- The input image has one channels. The convolution layers will progressively increase the number of output channels up to value 256.
- All max pooling should have $padding =' VALID'$.
- The max pooling layers may have $ksize = [1, 2, 2, 1]$ and $strides = [1, 2, 2, 1]$. These layers will reduce both the height and the width of the layer's input by 2.
- When the width and the height of the output of the above blocks are both ≤ 2 and the number of channels is 256, insert a flatten layer.
- Then insert one or more dense layers.
- Use the ReLU non-linearity for all the layer but the flatten and the last one.

- Note that the last layer will not use a non-linearity function. Instead, its output will pass through the softmax function in other parts of the code.

4.2.2 Apply a Trained Model to Test Samples

When a network is trained its weights are saved to disk. In evaluation time the weights are reloaded from the disk.

- Iterate through all test images.
- In a session run the inference to calculate the output of the network, which will be an array $n\_classes$, where $n\_classes = 2$.
- Apply the softmax to this array to convert it to an array of probabilities.
- Decide *Interictal* if the first element of the array is larger than the second layer, else decide *Ictal*.
- Calculate the accuracy and F_1 measures.
- Repeat the above calculations by taking into account the prior probabilities of the two classes. A prior probability can be approximated as the number of files of a class divided by the total number of files.
- Repeat the classification by applying the logarithm log to spectrograms. Be careful when the values of the spectrogram are 0 or very close to 0.

5 Project deliverables

You must provide your own implementation of every method and algorithm of the project apart from advanced classifiers such as the random forests which you may use code from matlab or python libraries. You must also write a report where you provide all your results (plots, tables, etc.) and comments on the project questions. Include an introduction where you explain the mathematical background of your implementations. Your report should provide sufficient details that are needed to future readers in case they want to repeat your experiments. Provide explanations about your empirical design decisions (if any), the insights you learn in intermediate analysis steps, as well as your final results.

References

[1] J. Arnhold, P. Grassberger, K. Lehnertz, and C.E. Elger. A robust method for detecting interdependences: application to intracranially recorded eeg. *Physica D: Nonlinear Phenomena*, 134(4):419 – 430, 1999.

[2] P. W. Mirowski, Y. LeCun, D. Madhavan, and R. Kuzniecky. Comparing svm and convolutional networks for epileptic seizure prediction from intracranial eeg. In *2008 IEEE Workshop on Machine Learning for Signal Processing*, pages 244–249, 2008.

[3] Y. Qian, M. Bi, T. Tan, and K. Yu. Very deep convolutional neural networks for noise robust speech recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24(12):2263–2276, 2016.

[4] R Quian Quiroga, Z Nadasdy, and Y Ben-Shaul. Unsupervised spike detection and sorting with wavelets and superparamagnetic clustering. *Neural computation*, 16(8):1661–1687, 2004.

[5] Robin Tibor Schirrmeister, Jost Tobias Springenberg, Lukas Dominique Josef Fiederer, Martin Glasstetter, Katharina Eggensperger, Michael Tangermann, Frank Hutter, Wolfram Burgard, and Tonio Ball. Deep learning with convolutional neural networks for eeg decoding and visualization. *Human Brain Mapping*, 38(11):5391–5420, 2017.

[6] Nhan Duy Truong, Levin Kuhlmann, Mohammad Reza Bonyadi, Jiawei Yang, Andrew Faulks, and Omid Kavehei. Supervised learning in automatic channel selection for epileptic seizure detection. *Expert Systems with Applications*, 86:199 – 207, 2017.