Project report: An automated electroencephalography (EEG) pre-processing algorithm for artifact removal

Introduction to Computational Neuroscience (MTAT.03.291)

Student: Kristjan-Julius Laak

Date: May, 2014

Abstract

All the analysis of brain signals recorded from the scalp start with the most tedious part of any biological signal analysis – cleaning the recordings from non-brain-related activity. In this project, a novel algorithm was developed to automate some aspects of this pre-processing phase. The core of the algorithm developed by me uses density estimation accompanied with heuristics to determine channels that deviate the median of the EEG channels. I also apply semi-automated visual rejection protocols and an independent component analysis (ICA) method to reject the eye movement artifacts from the data. This method reliably repairs the bad channels by using interpolation and modern topographical approaches.

Table of contents

Introduction
Background information
    FieldTrip
    Data structure
    Batching
    Independent component analysis
Algorithm
    Overview
    Detecting bad channels
    Rejecting trials
    Interpolating
    Visual rejection
    ICA
The code
Introduction

A tedious phase of working with electroencephalography (EEG) involves cleaning the recording of different types of non-brain-related activity (i.e. measurement artifacts). This process requires choosing methods to correct or remove them as well as setting parameters for those methods that would assure a desirable mix of false positives and false negatives. In addition, different subjects may need slightly different treatment. The main goal of this project is to build a novel algorithm that could automate some aspects of this phase. But it is far from trivial to write an algorithm that would be canonical or generic for all possible types of data and artifacts. There are many ways to collect EEG data, starting from different experimental setups and ending with the recording apparatuses, electrode cap systems, and even the physiological gels used to keep the impedance low between the electrodes and the skin of a subject. Thus it is important to refine the goals of such automatic algorithm for dealing with EEG artifacts. In this project, I constraint the type of input data to be

a) Pre-processed to a specific data structure that of obtained from the preprocessing protocols of MATLAB toolbox Fieldtrip;

b) Separated into different observations i.e. trials as opposed to being a long continuous recording;

c) Band-pass and notch filtered (or not, if done after the artifact removal). That is, the algorithm does not apply any filters to the data.

In a regular artifact rejection protocol, slices of EEG data contaminated with artifacts can be simply thrown out. It is also relatively easy to cope with artifacts when the experimenter has conducted hundreds or thousands of observations. Indeed – a lot of trials means the trials containing artifacts can be removed as a whole and only the clean channels be kept. However, there are many cases where it is not reasonable to throw out the bad trials. For example, one might have a rare set of data from sleep experiments where subjects have been awakened 50 times over several days, and losing any of the awakenings might have a huge cost in time and money. It is not reasonable to throw the whole trial out just because only some channels are invalid. The aim of this project is refined to deal with the type of data that is more valuable to the experimenter and cannot be blankly rejected.

To match the constraints put on the data, there are two approaches that are used in this project:

- To identify all the bad channels (i.e. channels with too much noise or huge artifacts) and try to interpolate them with the nearest channels that are not contaminated;
- Use Independent Component Analysis (ICA) approach to reject the eye movement artifacts from the data.

To begin with, I introduce some background information relevant to EEG processing and this project. In the first part of the background section, I introduce the MATLAB toolbox Fieldtrip that is used to do much of the work involved in EEG data processing and analysis.
**Background information**

**Fieldtrip**

FieldTrip is the Matlab software toolbox for MEG and EEG analysis that is being developed at the Donders Institute for Brain, Cognition and Behaviour at the Radboud University Nijmegen, the Netherlands together with collaborating institutes. The development of FieldTrip is currently supported by funding from the BrainGain and the Human Connectome projects. The FieldTrip software is released as open source under the GNU general public license.

FieldTrip is a toolbox that contains a set of separate (high-level) functions, it is not a program with a graphical user interface. The toolbox functions can be combined into an analysis pipeline, i.e. a MATLAB script containing all steps of the analysis.

**Data structure**

EEG data is composed of many channels and many time points. Therefore it contains a sample, a single number representing electrovols, for every Channel x Time point:

![Data structure diagram](image)

Note that the exact timing cannot be derived from the data itself. For that we need to know the sample rate (in samples per second, or hertz). The representation of the data in terms of time is relative. In most cases only after defining trials will we be saying something like “at 0.5 seconds after stimulus”. The data used by my algorithm have to be obtained by calling the Fieldtrip function `ft_preprocessing`. The function processes the data to a Fieldtrip specific data structure with separate fields for each trial, sample info, etc., as illustrated in the following figure.
Batching

The analysis of an experiment typically involves a lot of repetition as similar analysis steps are taken for every condition and for every subject. Also, the same steps are often repeated with only slightly different settings (e.g. filters, timings). Because of this we should program our own functions around the FieldTrip functions. Batching is the ultimate aim of any analysis pipeline in Fieldtrip. It means that in the end most of your analysis steps can be repeated over all subjects and/or conditions with a single command. Batching increases the efficiency of the analysis done by increasing the speed and memory usage and will allow us to automate most of the steps that do not require manual labour. In this project, the algorithm is developed with the best practices of batching.

Independent component analysis

Severe contamination of EEG activity by eye movements, blinks, muscle, heart and line noise is a serious problem for its interpretation and analysis. Many methods exist to remove eye movement and blink artifacts. Simply rejecting contaminated epochs results in a considerable loss of collected information.

Independent Component Analysis (ICA) can effectively detect, separate and remove activity in EEG records from a wide variety of artifactual sources, with results comparing favourably to those obtained using regression- or PCA-based methods. First we need to decompose the data into independent components. The only thing we have to be sure of is that we only use the actual EEG channels and don’t use reference sensors or electrooculogram (EOG).

The ICA will return as many components as there are channels put in. Each component consists of a component time course for every trial together with a single topography:
Components are automatically sorted based upon on the sum of the weighting factors, commonly resulting in the most interesting components appearing on top.
Algorithm

Overview

In the end we would like to have certain standardized approach to our bad channel removal algorithm that will give the best results possible. I ended up with the following procedure.

1. Per each trial, detect and mark channels that have a high rate for containing artifacts or noise.
2. Interpolate the detected channels with the clean neighbouring channels.
3. Reject the trials that would cause too many channels to be rejected from the dataset.
4. Go through the data visually and manually select trials that still any noise or artifacts.
5. Reject the remaining channels that contain artifacts.
6. Decompose the data using ICA.
7. Find components clearly corresponding to eye blinks and saccades.
8. Recompose data without those components.

Next, I describe the methods used to identify channels with artifacts (i.e. bad channels) for the interpolation.

Detecting bad channels

The data is processing trial by trial. In order to detect the bad channels, the following algorithm was developed for each trial.

First, if an absolute value of any time point of a channel is over 250 uV, the channel is marked as contaminated. The amplitude estimation is based on the fact that the intrinsic amplitude of EEG signal measured from the scalp for a typical adult human is about 10 uV to 100 uV.

Second, each channel is standardized to a unit variance with the formula

$$X_s = \frac{X - \mu}{\sigma}$$

where $X_s$ is the result vector, $X$ is the original channel vector, $\mu$ is the median of the channel, and $\sigma$ the standard deviation of the channel. Therefore, each sample in the channel vector is represented by its standard score.

Third, for each normalized channel, the Euclidean distance from the median of the normalized trial matrix is calculated. This results in a number describing the overall distance of the channel from the median of the whole trial matrix.

Fourth, for each trial, a probability density estimate of the samples in the channel norms vector is computed with the MATLAB function ksdensity (See Figure 1 below). The function ksdensity evaluates the density estimate at 100 points covering the range of the data in the input vector. The estimate is based on a normal kernel function, using a window parameter that is a function of the number of points in the input vector.
Figure 1. A. Probability density estimate plots for each channel per trial. B. The red line on is a reflection of the left side curve from the maximum, illustrating a hypothetical Gauss curve.

It is intuitively presumed that the Euclidean distances of the clean channels’ norms follow a Gauss curve, i.e. the distances of the clean channels from the median follow a normal distribution. Alternatively, it is presumed the distance from median of bad channels deviate from the Gauss distribution of the clean channels. This is based on the fact that the data should be collected carefully with respect to electrodes’ impedance.

Fifth, to simulate a Gauss curve, the maximum of the density estimate is found, and the curve from the left side of the maximum is mirrored to the right side of the maximum. To determine the contaminated channels by the density estimate plot seen above, the channels farther than the end of the mirrored curve are marked as contaminated (See Figure 1 above).

It might happen, that there are so many bad channels in the data that the above estimate fails to provide a desired mix of false positive and false negatives. Thus, for these trials, the “cut off” point is marked at the third quartile of the Gauss curve. This is a compromise between marking too many clean channels as bad and still marking some of the channels as invalid. After the procedure, some channels are marked as invalid. The head model for electrode positions and red electrodes indicating a possible resultant bad channels detected by the algorithm is illustrated in the Figure 2.
**Interpolating**

The interpolation of previously marked channels is done using FieldTrip function `ft_channelrepair`. The function uses nearest-neighbour approach to repair bad channels in EEG data by replacing them with the average of the their neighbours weighted by distance. The method uses a neighbourhood structure, i.e. predefined structure defining the neighbouring channels for each channel. Unfortunately, the nearest-neighbour approach cannot be used reliably to repair multiple bad channels that lie next to each other. Thus, the neighbourhood structure has to be modified for each trial separately so that no channel that needs to be repaired has neighbours that need to be repaired themselves. If a channel has less than two neighbours after altering the latter structure, the channel is not interpolated. After interpolating, the final decision of which channel to keep in the dataset is done visually.

**Visual rejection**

The final decision of which trials/channels to keep is done visually using FieldTrip function `ft_rejectvisual`. The function guides the user through all the data to visually select the trials/channels.
that should be rejected, i.e. thrown out of the dataset. This manual approach was chosen after a thorough study of the automatic data rejection methods available. It is important to notice that per one dataset, a bad channel in one trial denotes removing the channel from all of the trials. Thus, first the trials containing an excessive amount of bad channels compared to the other trials can be rejected visually from the dataset. Visual inspection for the channels was chosen because there is a possibility that some of the channels that the algorithm did not detect were still bad channels or that there were less than two neighbouring channels so that interpolation could not be done. In any case, removing too many or not enough channels per dataset would result in an undesired mix of false positives and false negatives.

ICA

If during an EEG acquisition procedure the subject moves eyes suddenly, the signal will be distorted heavily. A common way for removing eye movement artifacts from EEG data is to use independent component analysis (ICA).

ICA is a an elegant and practical computational blind source separation method to recover a set of underlying components which are statistically maximally independent from each other. The resultant components are automatically sorted based upon on the sum of the weighting factors. Determination of the components with eye movement artifacts is aided by spatial topography of the components. An illustration of the components can be found in Figure 3.

Figure 3. The spatial topography of the ICA components were used to determine the eye movement artifacts. In this illustration, the second head model from the left represents such artifact.

A trained eye can relatively easily spot the components that represent artifacts from the eye movements. After performing ICA the user is prompted with a dialog to inject the component numbers to be rejected. Then, the data back-projected to its original form. The difference between the raw and pre-processed data is illustrated in the Figure 4 below.
Figure 4. The difference between raw data (A) and pre-processed data (B) for the same trial. Different colours mark different channels.

The code

To run the code, the user has to have already a pre-processed dataset which is in Fieldtrip specific format (use function `ft_preprocessing`). For each subject, a specific .m file has to be written (by simply copying the example file `sub1.m`) containing all the information necessary to run the script. To run the code, open the coreEXE.m file and run it. In some points of the algorithm, the user is prompted with dialog boxes for input arguments such as the name of the subject file and the components to reject after ICA. Note that in this project MATLAB R2013a was used.