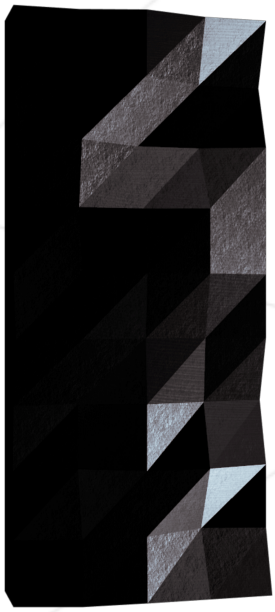


Total perspective vortex

Plug your brain to the shell

Jean Pirsch jpirsch@student.42.fr

Summary: *Brain computer interface with machine learning based on electoencephalographic data.*

Version: 1

Contents

Chapter I

Foreword

"The Total Perspective Vortex derives its picture of the whole Universe on the principle of extrapolated matter analyses. To explain — since every piece of matter in the Universe is in some way affected by every other piece of matter in the Universe, it is in theory possible to extrapolate the whole of creation — every sun, every planet, their orbits, their composition and their economic and social history from, say, one small piece of fairy cake. The man who invented the Total Perspective Vortex did so basically in order to annoy his wife. Trin Tragula — for that was his name — was a dreamer, a thinker, a speculative philosopher or, as his wife would have it, an idiot. And she would nag him incessantly about the utterly inordinate amount of time he spent staring out into space, or mulling over the mechanics of safety pins, or doing spectrographic analyses of pieces of fairy cake. “Have some sense of proportion!” she would say, sometimes as often as thirty-eight times in a single day. And so he built the Total Perspective Vortex — just to show her. And into one end he plugged the whole of reality as extrapolated from a piece of fairy cake, and into the other end he plugged his wife: so that when he turned it on she saw in one instant the whole infinity of creation and herself in relation to it. To Trin Tragula’s horror, the shock completely annihilated her brain; but to his satisfaction he realized that he had proved conclusively that if life is going to exist in a Universe of this size, then the one thing it cannot afford to have is a sense of proportion."

Douglas Adams, The Restaurant At The End Of The Universe

Chapter II

Introduction

This subject aims to create a brain computer interface based on electroencephalographic data (EEG data) with the help of machine learning algorithms. Using a subject's EEG reading, you'll have to infer what he or she is thinking about or doing - (motion) A or B in a t0 to tn timeframe.

Chapter III

Goals

- Process EEG datas (parsing and filtering)
- Implement a dimensionality reduction algorithm
- Use the pipeline object from scikit-learn
- Classify a data stream in "real time"

Chapter IV

General instructions

You'll have to process data coming from cerebral activity, with machine learning algorithms. The data was mesured during a motor imagery experiment, where people had to do or imagine a hand or feet movement. Those people were told to think or do a movement corresponding to a symbol displayed on screen. The results are cerebral signals with labels indicating moments where the subject had to perform a certain task.

You'll have to code in Python as it provides MNE, a library specialized in EEG data processing and, scikit-learn, a library specialized in machine learning.

The subject focuses on implementing the algorithm of dimensionality reduction, to further transform filtered data before classification. This algorithm will have to be integrated within sklearn so you'll be able to use sklearn tools for classification and score validation.

Chapter V

Mandatory part

V.1 Structure

You will have to write a python program implementing the three phases of data processing:

V.1.1 Preprocessing, parsing and formating

First you'll need to parse and explore EEG data with MNE, from physionet. You will have to write a script to visualize raw data then filter it to keep only useful frequency bands, and visualize again after this preprocessing.
This part is where you'll decide which features you'll extract from the signals to feed them to your algorithm. So you'll have to be thorough picking what matters for the desired output.
One example is to use the power of the signal by frequency and by channel to the pipeline's input.
Most of the algorithms linked to filtering and obtaining the signal's specter use fourier transform or wavelet transform (cf. bonus).

V.1.2 Treatment pipeline

Then the processing pipeline has to be setup :

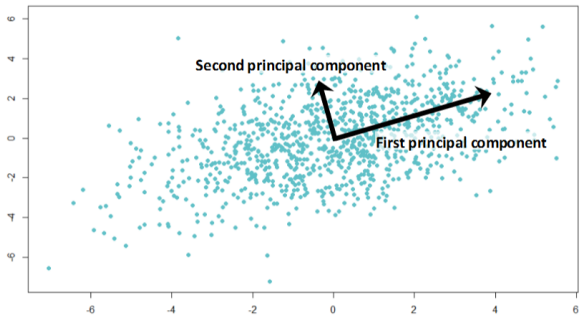
- Dimensionality reduction algorithm (ie : PCA, ICA, CSP, CSSP...).
- Classification algorithm, there is plenty of choice among those available in sklearn, to output the decision of what data chunk correspond to what kind of motion.
- "Playback" reading on the file to simulate a data stream.

It is advised to first test your program architecture with sklearn and MNE algorithms, before implementing your own CSP or whatever algorithm you chose.
The program will have to contain a script for training and a script for prediction.
The script predicting output will have to do it on a stream of data, and before a delay of 2s, after the data chunk was sent to the processing pipeline.

You will also have to use the pipeline object from sklearn, so you will be able to use cross_val_score on the whole processing pipeline, to evaluate your classification.

V.1.3 Implementation

The aim is to implement the dimensionality reduction algorithm. This means to express the data with the most meaningful features, by determining a projection matrix.
This matrix will project the data on a new set of axises that will express the most "important" variations. It is called a change of basis, and it is a transformation composed of rotations, translations and scaling operations.
As such the PCA considers your dataset and determine new basis components, sorted by how much those axises account for variations in the data.



The CSP or common spatial patterns, analyses the data depending on the output classes and try to maximize the variations between them.
PCA is a more general algorithm, but CSP is more used in EEG BCIs. Lets take the formal expression of an EEG signal :

$$\{E_n\}_{n=1}^{N} \in \mathbb{R}^{ch*time} \tag{V.1}$$

we have :

- N the number of event of every classes,
- ch number of channels (electrodes)
- $time$ the length of event recording

Considering the extracted signal matrix $X \in \mathbb{R}^{d*N}$, knowing that $d = ch * time$ is the dimension of a signal vector for an event record.

Your objective will be to find transformation matrix W such that :
$W^T X = X_{CSP}$ where X_{CSP} correspond to the transformed data by the CSP algorithm (or X_{PCA}, X_{ICA}, ... depending on your choice) .

Are also allowed Numpy or scipy functions to find eigenvalues, singular values, and covariance matrix estimation.

Chapter VI

Bonus part

The bonuses might be improvements on any step of the subject like :

- Improve preprocessing by working on signal specter variation (ie : use wavelets transform).
- Implement your own classifier or any other step of the pipeline
- Work on other datasets.

The implementation of another part of the pipeline allow to dig deeper into the parsing, the preprocessing or the classification. An harder bonus would be to coder your own functions for eigenvalues | singular values decompostion or covariance matrix estimation (this task is hard because the data are subject to noise and don't form a square matrix).

Chapter VII

Turn-in and peer-evaluation

Turn your work in using your `GiT` repository, as usual. Only work present on your repository will be graded in defense.

Only the python program needs to be present on your repository and not the dataset.