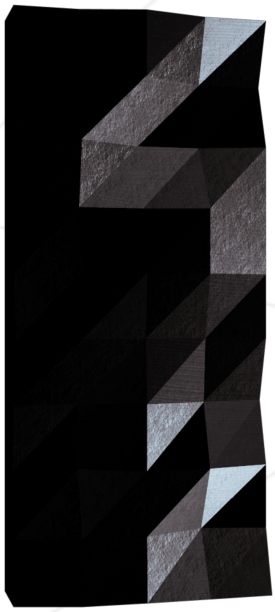


Total perspective vortex

Plug your brain to the shell

Jean Pirsch jpirsch@student.42.fr

Résumé: Interface cerveau-machine à base de machine learning et d'électroencéphalogramme.

Table des matières

I Préambule **2**

II Introduction **3**

III Objectifs **4**

IV Consignes générales **5**

V Partie obligatoire **6**

V.1 Structure du rendu 6

V.1.1 Prétraitement, parsing et formatage 6

V.1.2 Pipeline de traitement 6

V.1.3 Implementation 7

VI Partie bonus **8**

VII Rendu et peer-évaluation **9**

Chapitre I

Préambule

"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

Chapitre II

Introduction

Le but de ce sujet est de créer une interface cerveau machine basée sur des données d'électroencéphalogramme (EEG) à l'aide de machine learning. À partir d'un À partir d'un set de data vous devrez déterminer si la personne dont on a mesuré l'activité cérébrale pense ou effectue , indifferemment, un mouvement A ou un mouvement B durant la fenêtre de temps de t0 à tn.

Chapitre III

Objectifs

- Manipuler des données EEG (parsing et filtrage)
- Implementer un algorithme de réduction de dimension sur les données
- Maitriser l'objet pipeline de scikit-learn
- Classifier en temps réel un flux de data

Chapitre IV

Consignes générales

Pour ce sujet vous devrez manipuler des données provenant d'enregistrements de l'activité cérébrale, pour les traiter avec des algorithmes de machine learning. En particulier les datas à traiter ont été mesuré lors d'expériences où les sujets avaient à imaginer ou effectuer un mouvement des mains ou des pieds. Les sujets de l'expérience avaient pour consigne de penser ou d'effectuer (selon la phase de l'expérience) un mouvement correspondant à un symbole affiché sur un écran. On a donc un enregistrement du signal cérébral (provenant d'un casque d'électrodes) et des labels pour marquer les instants qui correspondent aux différentes actions.

Vous allez coder en python qui dispose de la librairie MNE spécialisée dans la manipulation de ces données. Celle-ci dispose notament les fonctions de parsing de fichiers EEG, et de filtrage du signal.

La partie centrale est d'implementer l'algorithme de réduction de dimension, qui transforme les données filtrées avant la classification. Cet algorithme devra s'intégrer au pipeline de traitement de la librairie sklearn pour pouvoir utiliser les outils de validation de score.
Ce pipeline sera donc constitué d'un algorithme de réduction de dimension des données, puis d'un classifieur de sklearn.

Chapitre V

Partie obligatoire

V.1 Structure du rendu

Pour répondre aux consignes il faudra donc rédiger un programme en python qui implémente les trois phases du traitement des données :

V.1.1 Prétraitement, parsing et formatage

Il faut commencer par s'approprier les données EEG, et donc utiliser MNE pour parser les données du dataset de physionet. Vous devrez avoir un script qui permet de visualiser les données brutes, puis filtrer ces données pour ne garder qu'une bande de fréquences utile dans le cadre d'une expérience de mouvement, et revisualiser après filtrage le signal. C'est la partie qui consiste à extraire les premières caractéristiques sur lesquelles se baser pour classifier le mouvement. C'est donc là que se pose la réflexion sur les aspects pertinents pour la décision de l'algorithme.
Par exemple vous pouvez utiliser la puissance du signal par fréquence et par channel comme donnée d'entrée du pipeline.
La plupart des algorithmes liés au filtrage et au spectre du signal utilisent les séries de fouriers, ou la transformée en ondelette (cf. bonus).

V.1.2 Pipeline de traitement

Ensuite il faudra mettre en place le pipeline de traitement des données :

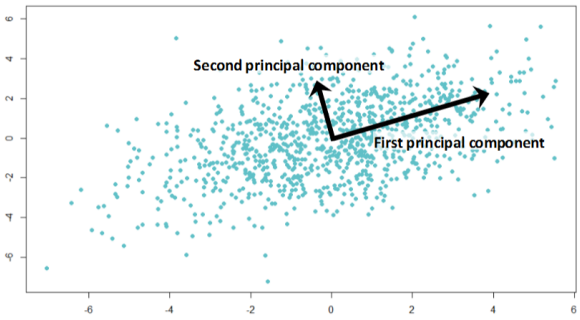
- Algorithme de transformation permettant de réduire la dimension des datas (ie : PCA, ICA, CSP, CSSP...).
- Algorithme de classification, à choisir parmis ceux de sklearn, pour faire la correspondance entre un segment de data et un mouvement.
- Mode de lecture "playback" des données pour simuler un stream de datas.

Il est conseillé de faire l'implémentation de votre propre algo de transformation après, pour d'abord vérifier que l'architecture de votre programme fonctionne sur le principe avec les implémentation disponibles dans sklearn et MNE.
Le programme devra avoir un script pour l'entrainement et un script pour la prédiction. Le script faisant la prediction devra lire les données comme un stream et aura un délai de 2 secondes pour afficher à quelle classe correspond le dernier évènement traité.

Un autre aspect important de la structure du programme est l'utilisation de l'objet pipeline de la librairie sklearn. L'intérêt est de pouvoir utiliser cross_val_score sur tout le pipeline, pour évaluer le score de la classification.

V.1.3 Implementation

Ici le but est d'implémenter l'algorithme de réduction de dimension. Il s'agit d'aller extraire dans les données les valeurs pertinentes pour la classification en déterminant une matrice de projection.
Celle-ci servira à projetter les données dans un nouveau repère où les variations pertinentes des données seront maximisées. Un changement de repére est une transformation qui peut être composée de rotations, translations et "scaling" sur les vecteurs de la base de votre espace.
Par exemple l'algorithme PCA considère l'ensemble de vos données et détermine de nouvelles composantes de bases, dont la première exprime l'axe déterminant le maximum de variance.



Quant à lui, CSP ou common spatial patterns, considère vos données en fonction des classes à déterminer et cherche à maximiser la variance entre elles.
PCA est un algorithme plus général, tandis que CSP est utilisé majoritairement dans le domaine des interfaces EEG. Prenons la formalisation d'un signal EEG :

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

on à :

- N le nombre d'évènements toutes classes confondues,
- ch le nombre de channels (électrodes)
- $time$ la durée de l'enregistrement des évènements

On considère la matrice du signal extrait $X \in \mathbb{R}^{d*N}$, sachant que $d = ch * time$ est la dimension d'un vecteur correspondant au signal en un point du temps.

Votre objectif sera de déterminer la matrice de transformation W telle que :
$W^T X = X_{CSP}$ où X_{CSP} correspond aux données transformées à l'aide de l'algorithme CSP (ou X_{PCA}, X_{ICA}, ... en fonction de votre choix) .

Dans ce but vous pourrez utiliser les fonctions de numpy ou scipy pour la détermination des valeurs propres, des valeurs singulières, et l'estimation de la matrice de covariance.

Chapitre VI

Partie bonus

Les bonus possibles sont des améliorations d'une ou des étapes du sujet, comme :

- Améliorer le prétraitement en travaillant sur la variation du spectre du signal (ex : utiliser la transformée en ondelettes).
- Implémenter vous même un classifieur ou une autre étape du pipeline.
- Traiter d'autres datasets que celui proposé par le sujet.

Concernant l'implementation d'une autre partie du pipeline, cela permet d'appronfondir le parsing, la transformation ou la classification. Un bonus plus dur serait de recoder les fonctions de recherche de valeurs propres, de valeurs singulières ou de matrice de covariance (c'est parceque les données sont soumises à du bruit et ne forment pas une matrice carrée que la tâche est compliquée, dans le cas de matrices "simples" c'est beaucoup plus trivial).

Chapitre VII

Rendu et peer-évaluation

Rendez-votre travail sur votre dépot `GiT` comme d'habitude. Seul le travail présent sur votre dépot sera évalué en soutenance.

Ne devra être présent sur le dépot que le programme python et pas le dataset.