Entraîner des modèles d'apprentissage automatique sur des données MEG avec MNE-Python

Yann Harel, Vanessa Hadid
Hamza Abdelhedi et Annalisa Pascarella

École d'été NeuroQAM - 13 mai 2025

Plan

Section 1 : Introduction à l'analyse des données EEG/MEG avec MNE-Python

Section 2 : Explorer les données brutes, les prétraiter et générer des réponses évoquées – script 1

Section 3 : Explorer les statistiques multivariées pour décoder les informations issues des données M/EEG – script 2

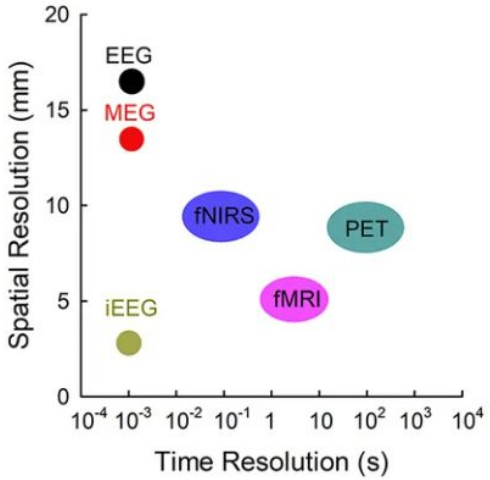
Section 4 : Appliquer le Common Spatial Pattern (CSP) pour décoder l'imagerie motrice à partir de données EEG – script 3

Section 5 : Activité pratique

Section 1

Introduction à l'analyse des données EEG/MEG avec MNE-Python

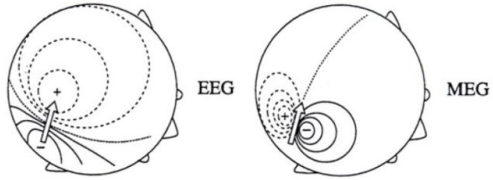
Figure 1

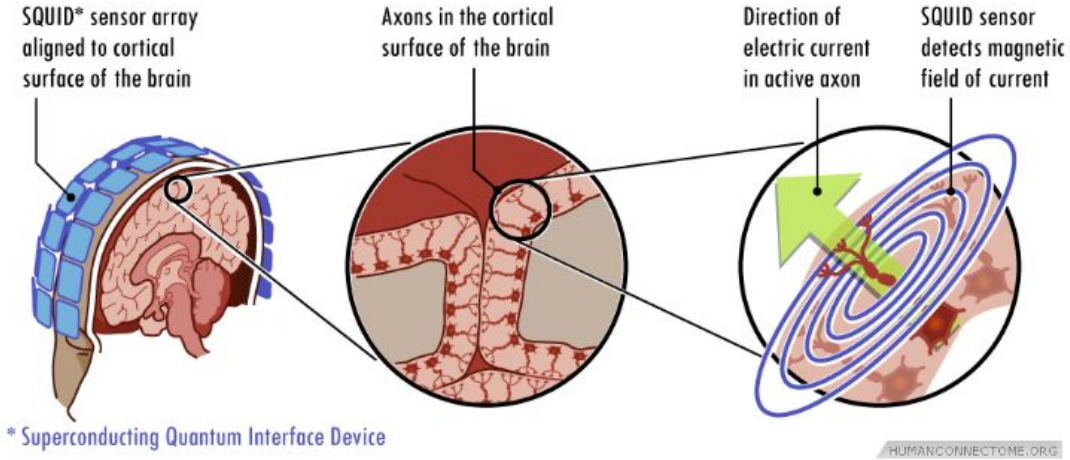


Difference between Electric Field vs Magnetic Field	
Electric Field	**Magnetic Field**
Measured as newton per coulomb, volt per metre.	Measured as gauss or tesla.
Proportional to the electric charge.	Proportional to the speed of electric charge.
It is perpendicular to the magnetic field.	It is perpendicular to the electric field.
An electric field is measured using an electrometer.	The magnetic field is measured using the magnetometer.

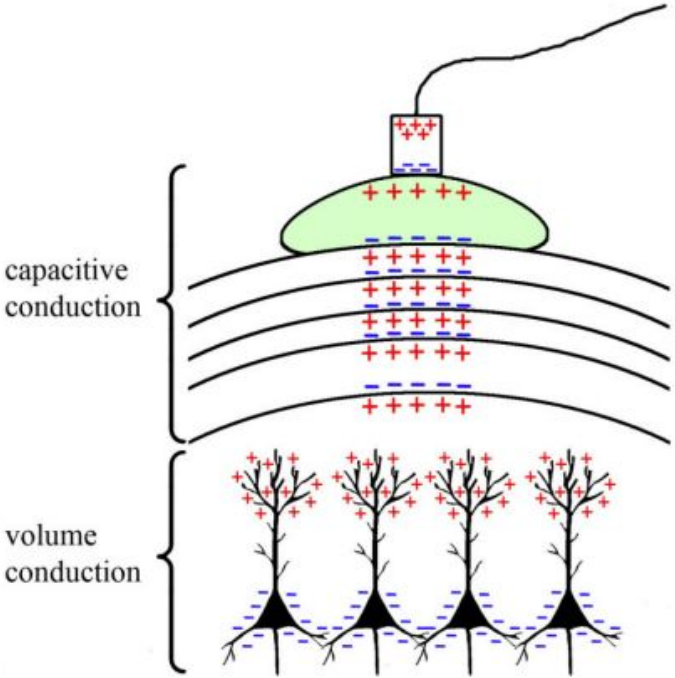
Xu et al., 2021, Frontiers in Human Neuroscience

Introduction à l'analyse des donné
avec MNE-Python



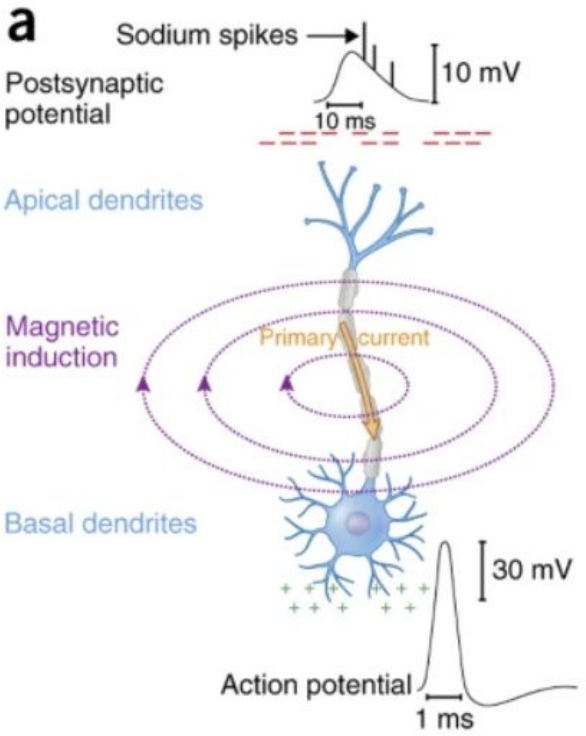


EEG : Champ électrique



Jackson & Bolger 2014, Psychophysiology

MEG : Champ magnétique

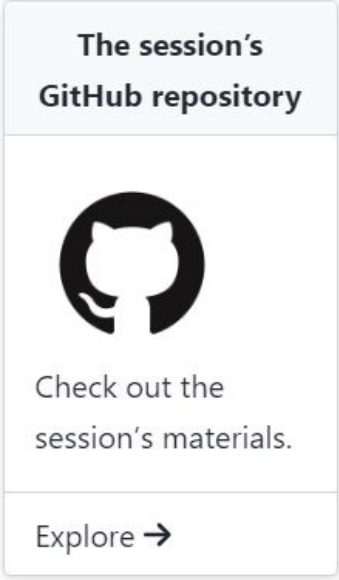


Baillet 2017, Nature Neuroscience

Objectifs de l’atelier

Utiliser les capacités de MNE-Python pour traiter et analyser les données EEG et MEG.
Contenu abordé :

- Prétraitement des données brutes
- Application de techniques avancées comme l’analyse de patterns multivariés (MVPA)
- Application de méthodes de décodage aux tâches d’imagerie motrice
  Dépôt : https://github.com/thecocolab/mne_meeg_ml_main
  MNE : https://mne.tools/stable/documentation/index.html



Section 2

Des données brutes aux réponses évoquées – script 1

Premiers pas avec MNE-Python

Configuration de l’environnement et gestion des données

Chargement du jeu de données d’exemple

Prétraitement et inspection des données brutes

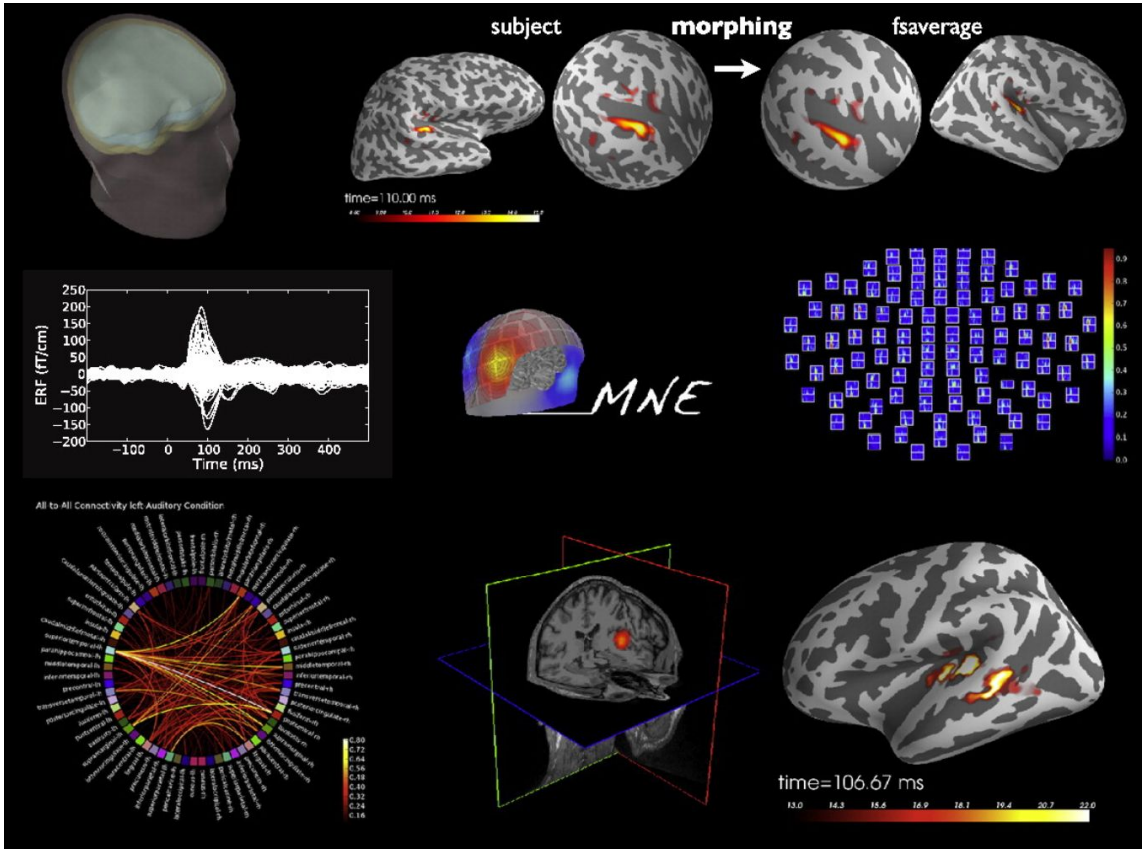
Définition et extraction des époques

Moyennage des époques et visualisation des réponses évoquées

Getting Started with MNE-Python

- **Premiers pas avec MNE-Python**
  Introduction à MNE-Python
  MNE-Python est une puissante bibliothèque Python open source conçue spécifiquement pour l'analyse de données neurophysiologiques telles que l'EEG et la MEG. Elle facilite des chaînes d'analyse complexes allant du prétraitement à l'analyse statistique avancée.
- **Fonctionnalités principales**
  Depuis les données brutes, le prétraitement et la visualisation, jusqu'à l'analyse temps-fréquence et l'estimation de source.
  Intégration avec des bibliothèques d'apprentissage automatique : s'intègre à scikit-learn pour des décodages avancés.
  Portée par la communauté : régulièrement mise à jour avec de nouvelles fonctionnalités et une documentation complète soutenue par une communauté active.
- **Objectif en neuroimagerie**
  Le but principal de MNE-Python est de rendre accessibles des outils statistiques sophistiqués aux non-programmeurs tout en offrant une flexibilité maximale aux utilisateurs experts. Elle prend en charge l'ensemble du flux de travail en recherche M/EEG, de l'acquisition des données à la publication des résultats scientifiques.

Premiers pas avec MNE-Python



Gramfort et al., 2014, Neuroimage

Configuration et gestion des données dans MNE-Python

1. **Configuration de l'environnement**
   MNE-Python fonctionne avec les versions de Python 3.6 et supérieures.
   Vous pouvez l'installer avec la commande :
   `pip install mne` ou `conda install -c conda-forge mne`.

2. **Importation des bibliothèques nécessaires**
   Commencez votre script Python ou votre notebook en important MNE et d'autres bibliothèques essentielles comme NumPy pour les calculs numériques et Matplotlib pour les visualisations.
   Exemple :
   `import mne`, `import numpy as np`, `import matplotlib.pyplot as plt`.

3. **Chargement des données d'exemple**
   MNE-Python propose des jeux de données d'exemple idéaux pour apprendre et tester les fonctionnalités de la bibliothèque.

Configuration et gestion des données dans MNE-Python

```
<Raw | sample_audvis_filt-0-40_raw.fif, 376 x 41700 (277.7 s), ~3.2 MB, data not loaded>
<Info | 14 non-empty values
 bads: 2 items (MEG 2443, EEG 053)
 ch_names: MEG 0113, MEG 0112, MEG 0111, MEG 0122, MEG 0123, MEG 0121, MEG ...
 chs: 204 Gradiometers, 102 Magnetometers, 9 Stimulus, 60 EEG, 1 EOG
 custom_ref_applied: False
 dev_head_t: MEG device -> head transform
 dig: 146 items (3 Cardinal, 4 HPI, 61 EEG, 78 Extra)
 highpass: 0.1 Hz
 hpi_meas: 1 item (list)
 hpi_results: 1 item (list)
 lowpass: 40.0 Hz
 meas_date: 2002-12-03 19:01:10 UTC
 meas_id: 4 items (dict)
 nchan: 376
 projs: PCA-v1: off, PCA-v2: off, PCA-v3: off, Average EEG reference: off
 sfreq: 150.2 Hz
```

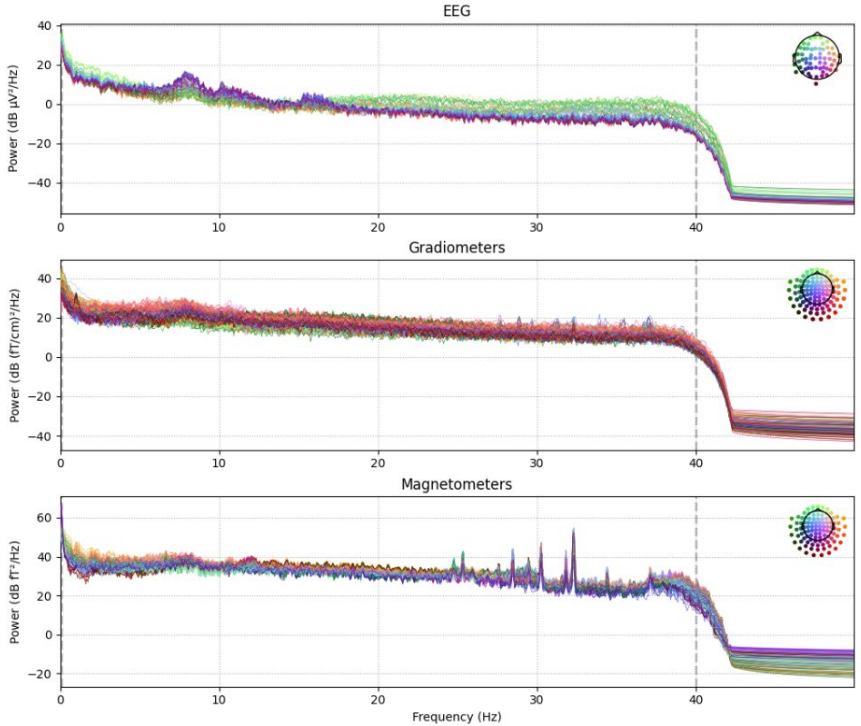
Left Auditory
Right Auditory
Left visual
Right visual

Chargement du jeu de données d’exemple

1. **Chargement des données**
   Jeu de données d’exemple fourni par MNE.

2. **Inspection des données**
   Examinez la structure et le contenu des données : `print(raw.info)`
   Nombre de canaux, fréquence d’échantillonnage, durée d’enregistrement.

3. **Visualisation des données brutes**
   Inspection visuelle des données brutes : `raw.plot()`
   Ce graphique interactif permet de faire défiler différents segments de données, de marquer les canaux défectueux et d’inspecter les événements.



Prétraitement et inspection des données brutes

1. **Aperçu du prétraitement**
   Le prétraitement est essentiel pour garantir la qualité et la fiabilité de l'analyse EEG/MEG.
   Il inclut le filtrage pour supprimer le bruit et les artéfacts, ainsi que l'identification des canaux défectueux qui pourraient fausser l'analyse.

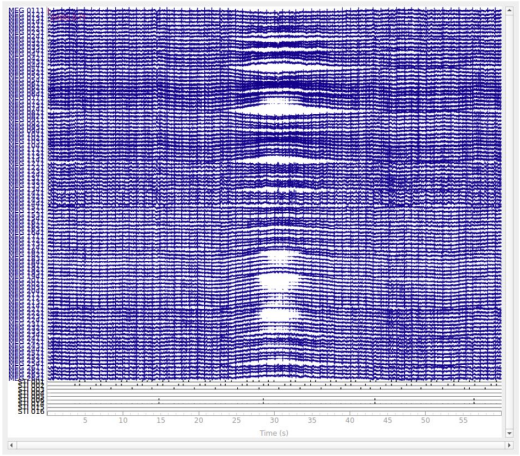
2. **Filtrage des données**
   Vous pouvez appliquer un filtre passe-bande pour conserver une plage de fréquences spécifique.
   Réinspectez les données pour vous assurer que le filtre a été correctement et efficacement appliqué.
   Visualisez les données filtrées pour observer la densité spectrale de puissance.
   Cela aide à identifier tout bruit de ligne résiduel ou autres artéfacts.

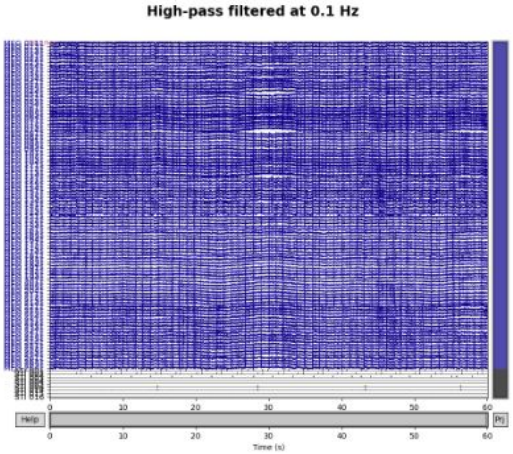
3. **Marquage des canaux défectueux**
   L'identification et le marquage des canaux défectueux sont des étapes cruciales du prétraitement.
   Un canal peut être considéré comme défectueux s'il présente un bruit excessif, un signal plat ou des lectures anormales.
   Vous pouvez les marquer manuellement en examinant le graphique ou automatiquement à l'aide de critères statistiques.

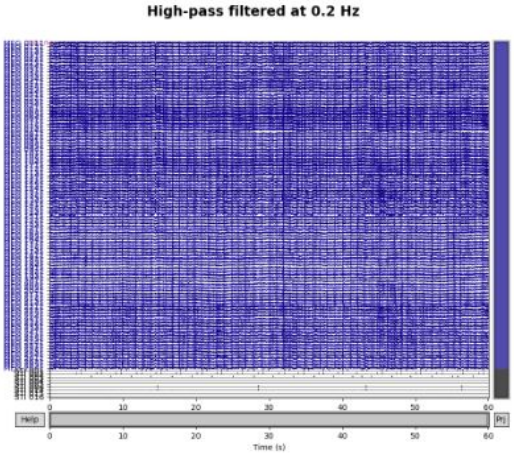
Prétraitement et inspection des données brutes

Filtrage des données

Original data

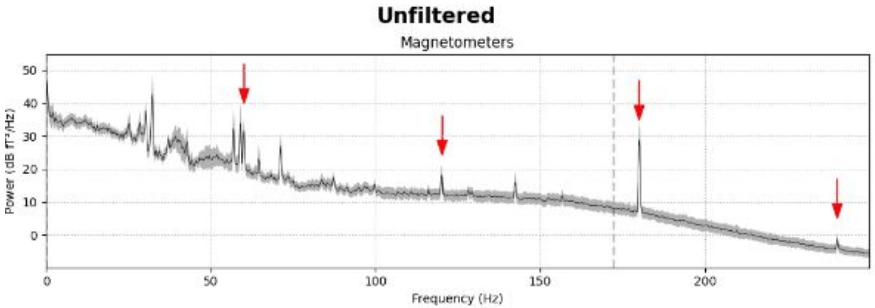


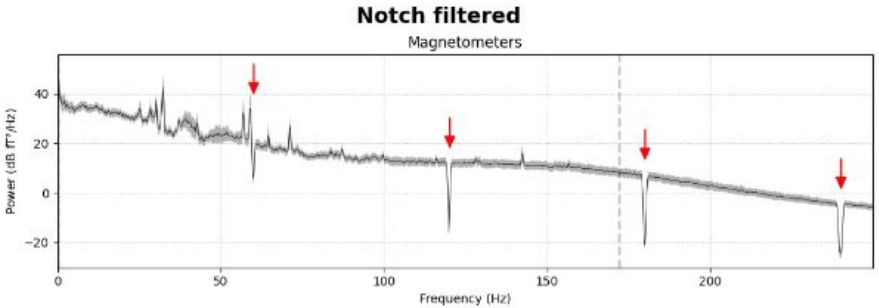




Prétraitement et inspection des données brutes

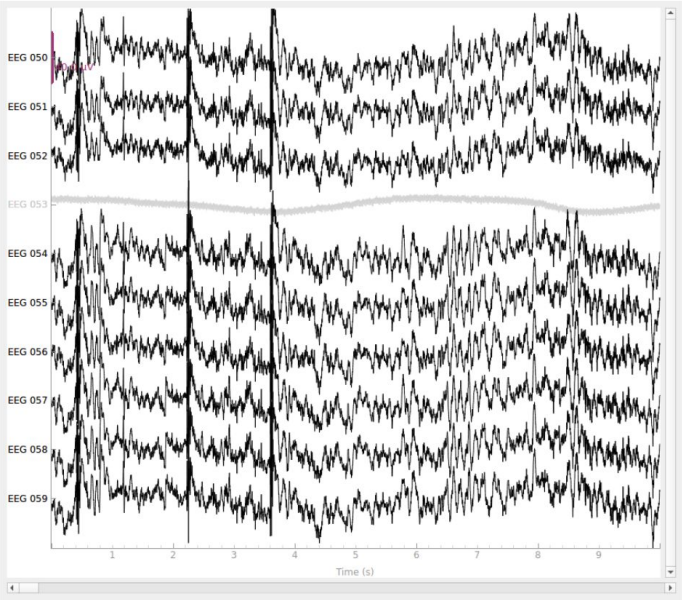
Filtrage des données



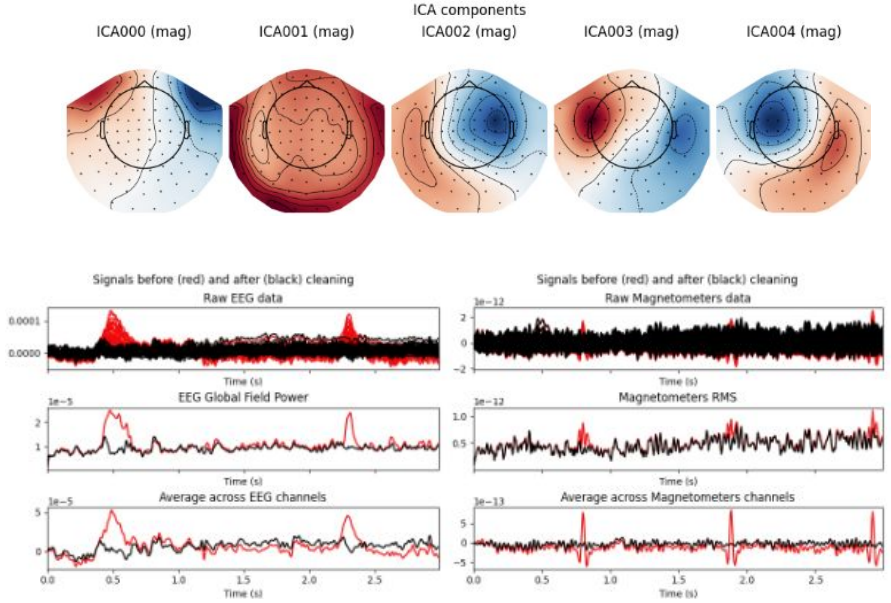


Prétraitement et inspection des données brutes

Marquage des canaux défectueux



Correction avec l'ICA (Analyse en Composantes Indépendantes)



Définition et extraction des époques

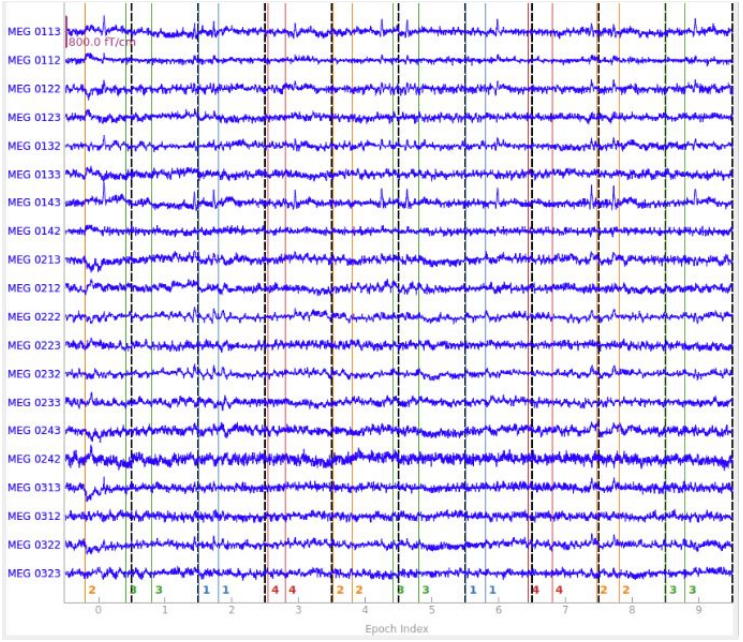
1. **Introduction aux époques**
   Les époques sont des segments de données extraits autour d'événements spécifiques, utilisés pour analyser des activités liées à des événements.
   Elles sont synchronisées sur des stimuli externes ou des événements internes, tels que des stimuli sensoriels ou des actions motrices.

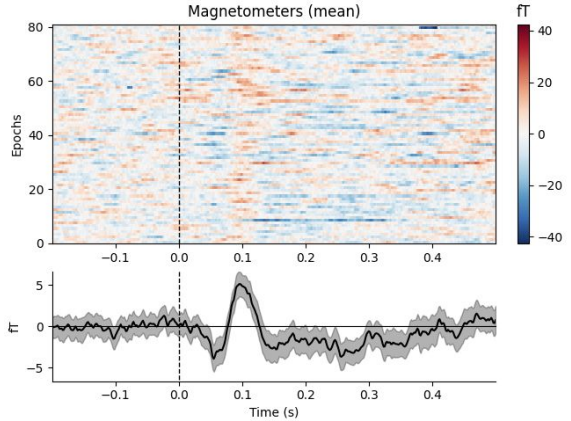
2. **Extraction des événements**
   Identifier les événements autour desquels créer les époques avec des déclencheurs :
   `events, event_id = mne.events_from_annotations(raw)`

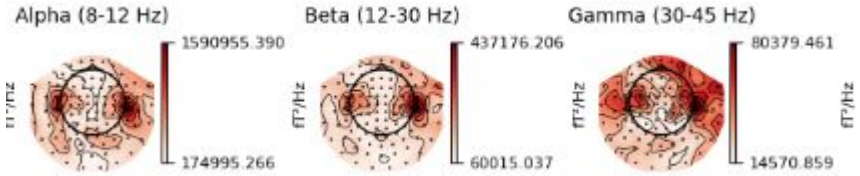
3. **Création des époques**
   Définir une ligne de base avant l'événement et une période après l'événement :
   `epochs = mne.Epochs(raw, events, event_id, tmin=-0.2, tmax=0.5, baseline=(None, 0), preload=True)`

4. **Inspection des époques**
   `epochs.plot()` permet de visualiser les époques dans un graphique interactif, de les faire défiler et de marquer celles qui sont jugées incorrectes.

Définition et extraction des époques







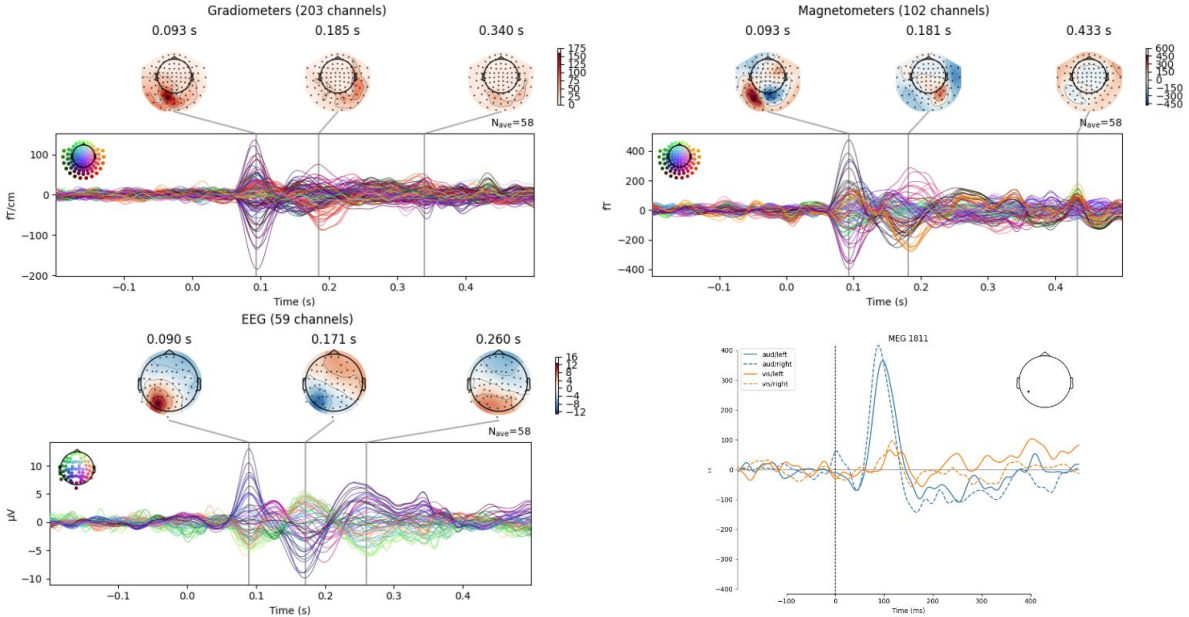
Moyennage des époques et visualisation des réponses évoquées

1. **Comprendre les réponses évoquées**
   Les réponses ou potentiels évoqués sont des signaux moyennés, synchronisés à des stimuli ou événements externes.
   Ils mettent en évidence la réponse directe du cerveau à des stimuli spécifiques.
   Ils permettent également de réduire le bruit non synchronisé à l'événement.

2. **Moyennage des époques**
   Pour moyenner les époques et créer une réponse évoquée :
   `evoked = epochs.average()`
   Cette commande calcule le signal moyen de toutes les époques pour chaque canal.

3. **Visualisation des réponses évoquées**
   `evoked.plot()` permet d'afficher la réponse évoquée.
   On observe des pics correspondant à des composantes sensorielles ou cognitives typiques, telles que le P300 ou le N170.

Moyennage des époques et visualisation des réponses évoquées



Section 3

Statistiques multivariées (Décodage / MVPA) – script 2

Introduction à l’analyse multivariée

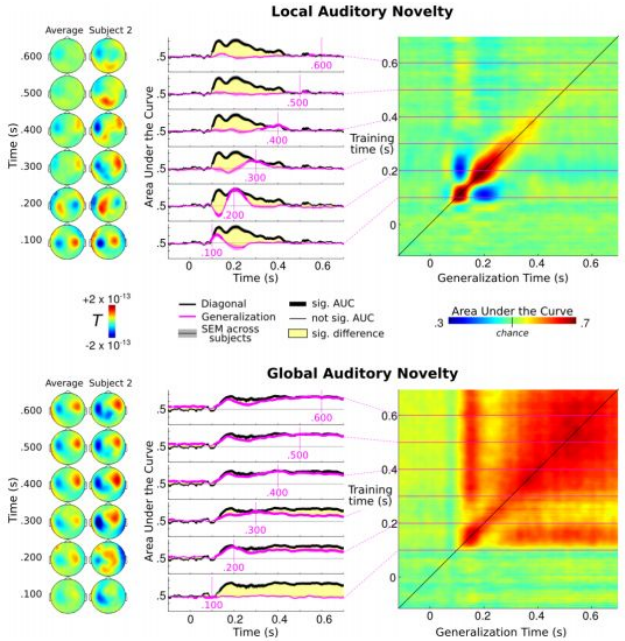
Classification avec Linear Discriminant Analysis (LDA)

Analyses temporelles

Introduction à l'analyse multivariée

1. **Aperçu de l'analyse multivariée (Décodage / MVPA)**
   Techniques statistiques permettant de prédire ou de classifier des points de données à partir de plusieurs variables.
   Utilisées pour prédire des états cognitifs ou des conditions expérimentales à partir de signaux cérébraux complexes.
   Permettent de capturer des motifs subtils à travers plusieurs canaux et moments temporels (multivarié).
   Outils puissants pour analyser des signaux neuronaux en fonction du temps.
   Fournissent des informations sur la dynamique spatiale et temporelle du fonctionnement cérébral.

2. **Intégration avec Scikit-Learn dans MNE-Python**
   Scikit-Learn est une bibliothèque d'apprentissage automatique en Python utilisée pour des algorithmes de classification, de régression et de regroupement (clustering).



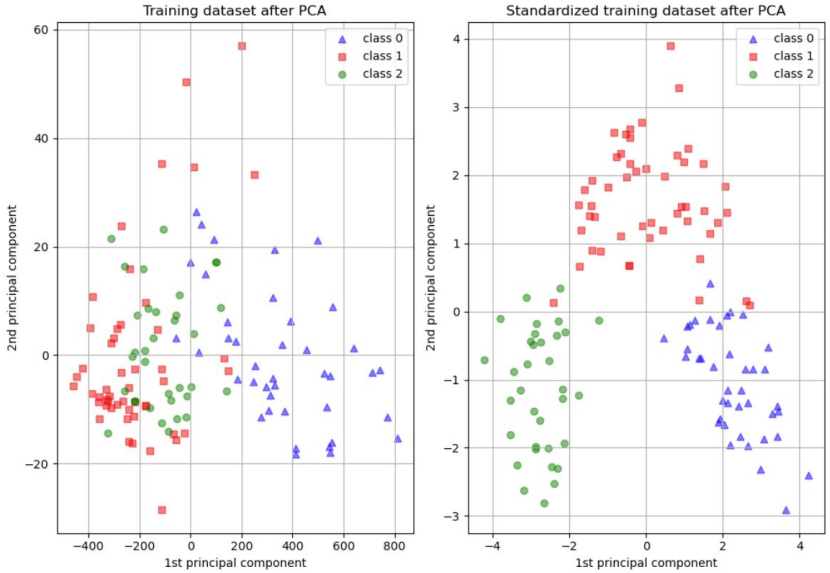
King et al., 2014, Plos One

Classification avec l'analyse discriminante linéaire (LDA)

1. **Sélection des caractéristiques**
   Les caractéristiques peuvent inclure des points temporels spécifiques ou des bandes de fréquence pertinentes pour les tâches ou états cognitifs étudiés.

2. **Normalisation des données**
   Il est important de normaliser ou de standardiser les données pour s'assurer que toutes les caractéristiques contribuent également à l'analyse.
   `StandardScaler` soustrait la moyenne et met à l'échelle les données selon une variance unitaire.

3. **Organisation des matrices de données**
   Organisez vos données en matrices de caractéristiques (`X`) et étiquettes (`y`).
   `X` peut contenir les amplitudes de signal ou densités spectrales de puissance pour différents canaux et instants,
   tandis que `y` représente les étiquettes de conditions expérimentales ou les réponses des participants :
   `X = epochs.get_data(); y = epochs.events[:, -1]`



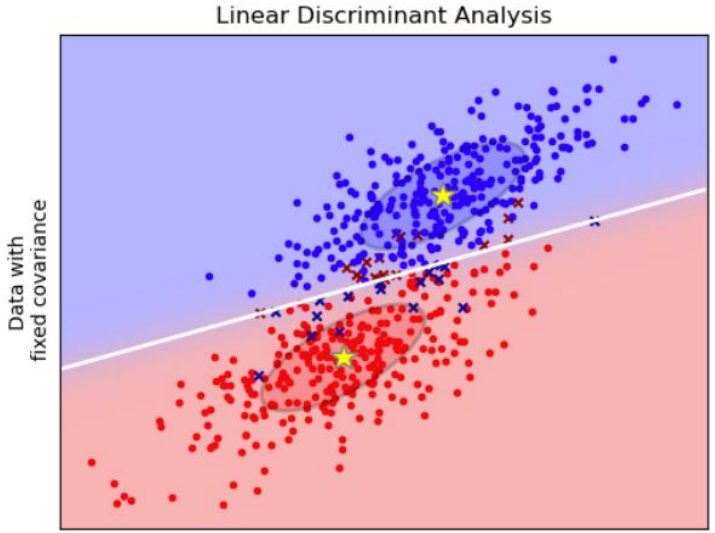
Out:
```
Prediction accuracy for the normal test dataset with PCA
81.48%

Prediction accuracy for the standardized test dataset with PCA
98.15%
```

Classification avec l’analyse discriminante linéaire

1. **Validation croisée**
   Permet de s’assurer que le modèle se généralise bien à de nouvelles données.
   Utilisation de techniques de validation croisée telles que `KFold` ou `StratifiedKFold`.
   On peut générer plusieurs divisions d'entraînement/test pour valider la stabilité et la précision du modèle.

```
from sklearn.model_selection import ShuffleSplit
cv = ShuffleSplit(10, test_size=0.2, random_state=42)
scores = cross_val_score(clf, X, y, cv=cv)
```

2. **Évaluation des performances du classificateur**
   On utilise des métriques telles que l’exactitude (accuracy), la précision, le rappel (recall), ou l’aire sous la courbe ROC.

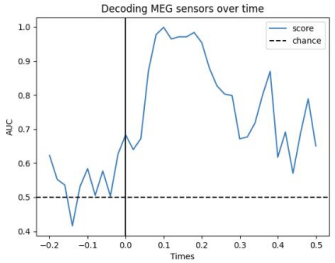


Analyse temporelle

1. **Décodage en fonction du temps**
   Le décodage temporel consiste à analyser comment le motif d'activité cérébrale évolue au fil du temps,
   ce qui permet de mieux comprendre la dynamique du traitement neuronal.

2. **Implémentation du décodage temporel**
   On utilise la LDA pour réaliser un décodage à chaque point temporel de l'essai.

```
from mne.decoding import SlidingEstimator
clf = make_pipeline(StandardScaler(), LinearDiscriminantAnalysis())
time_decod = SlidingEstimator(clf)
scores = cross_val_multiscore(time_decod, X, y, cv=5, n_jobs=1)
mean_scores = np.mean(scores, axis=0)
```



3. **Interprétation des résultats**
   On identifie les points temporels pour lesquels la classification dépasse significativement le hasard,
   ce qui révèle les moments clés de différenciation neuronale.

Décodage dans l'espace source

1. **Préparation à la reconstruction de sources**

   La reconstruction de sources consiste à remapper les données EEG/MEG observées aux localisations spatiales dans le cerveau à l'aide d'un opérateur inverse.

   ```
   inverse_operator = read_inverse_operator(fname_inv)
   ```

   On utilise l'estimation à norme minimale (MNE) pour effectuer la reconstruction des sources à partir des époques.

2. **Application du modèle inverse**

   On applique le modèle inverse aux données d'époques :

   ```
   stcs = apply_inverse_epochs(epochs_meg, inverse_operator, lambda2=1.0 / snr**2,
   method='dSPM', pick_ori='normal')
   ```

3. **Décodage à partir de l'espace source**

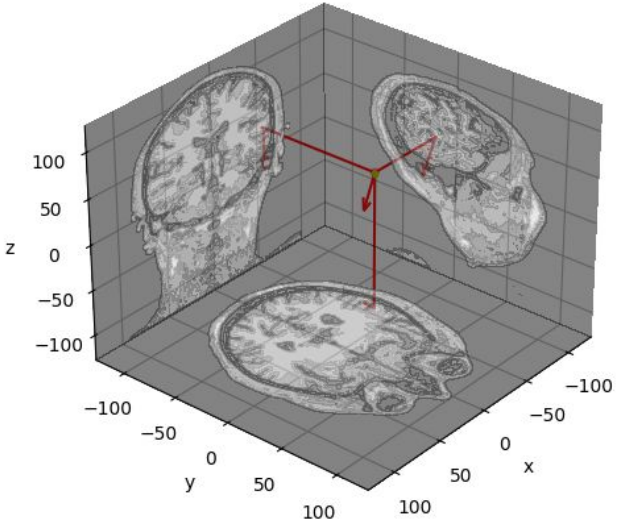
   On peut visualiser les estimations de source pour observer quelles régions cérébrales sont actives pendant les tâches :

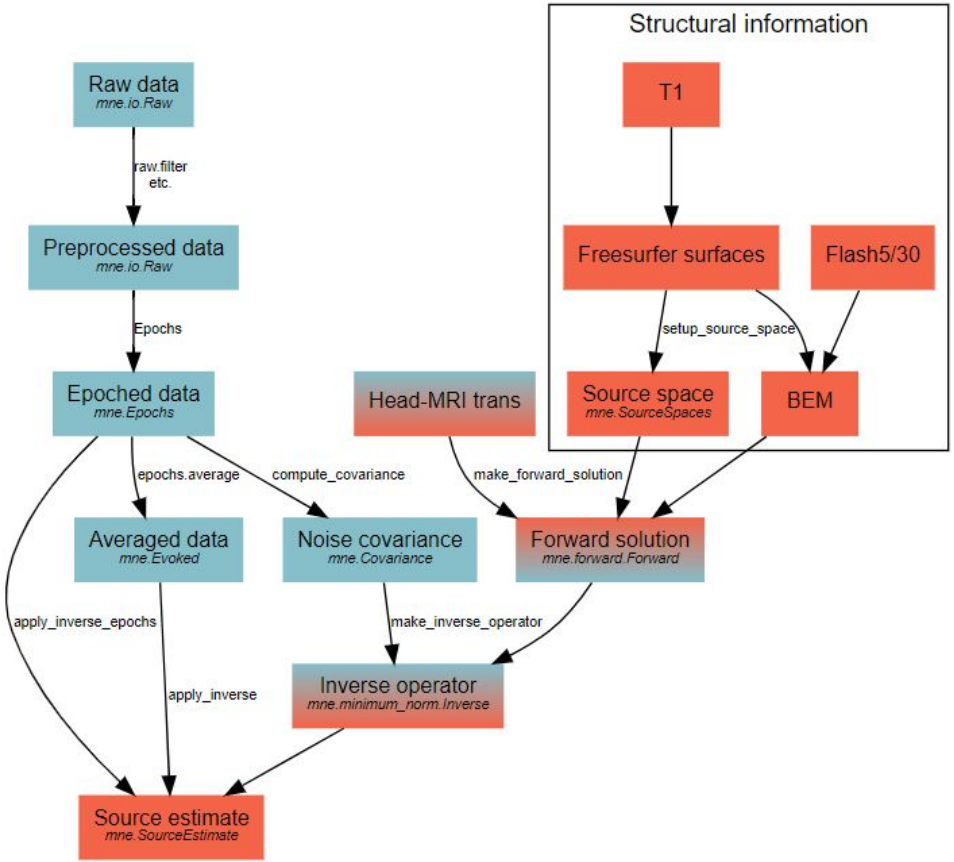
   ```
   stc.plot()
   ```

   Il est également possible de décoder des états cognitifs à partir des données reconstruites en transformant les estimations de source en vecteurs de caractéristiques exploitables par des modèles d'apprentissage automatique :

   ```
   X = np.array([stc.data for stc in stcs]); y = epochs.events[:, 2]
   ```

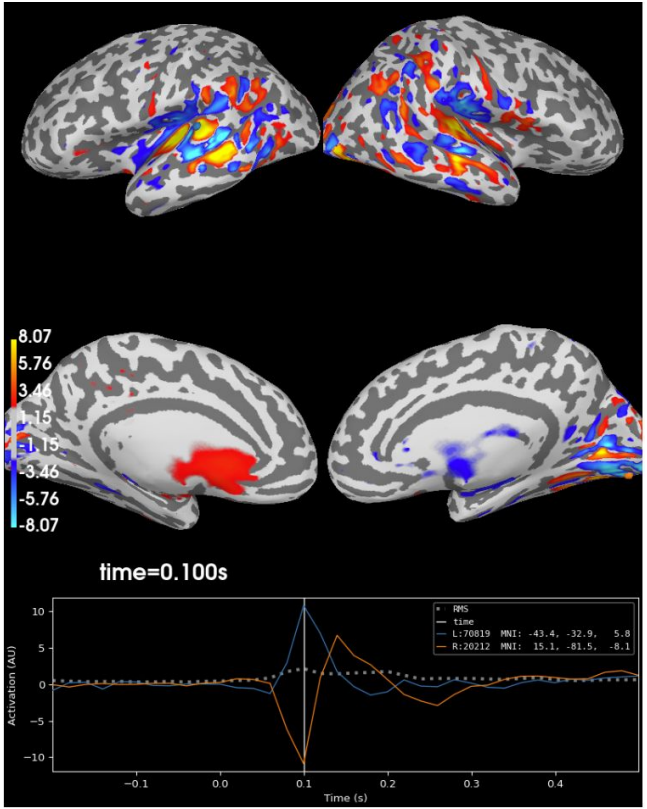
Décodage dans l'espace source

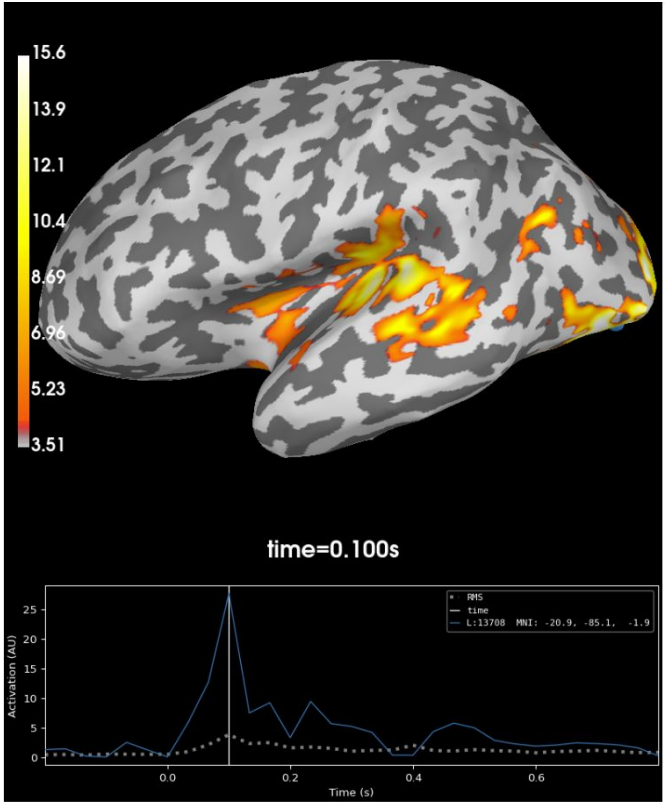




Workflow of the MNE software #

Décodage dans l’espace source





Section 4

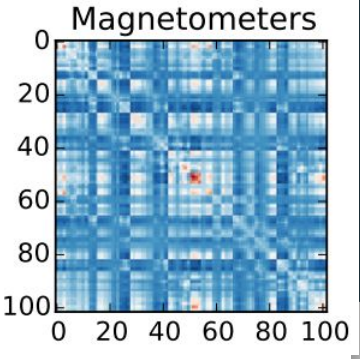
Décodage de l’imagerie motrice avec le Common Spatial Pattern (CSP) – script 3

Introduction au décodage de l’imagerie motrice

Données et application CSP

Intégrer CSP etLDA

Décodage de l’imagerie motrice avec CSP



1. **Comprendre l’imagerie motrice**

   L’imagerie motrice consiste à imaginer le mouvement de différentes parties du corps sans effectuer réellement le mouvement.

   Elle permet d’identifier les substrats neuronaux impliqués dans la planification et l’exécution motrice.

2. **Rôle du CSP dans l’imagerie motrice**

   Le *Common Spatial Pattern* (CSP) est une technique statistique utilisée pour mettre en évidence les composantes du signal présentant les plus grandes différences entre deux types d’imagerie motrice (par exemple, imaginer bouger les mains vs. les pieds).

   Le CSP filtre le signal de manière à maximiser la variance pour une condition tout en la minimisant pour l’autre,

   ce qui améliore efficacement le rapport signal/bruit pour les tâches de classification.

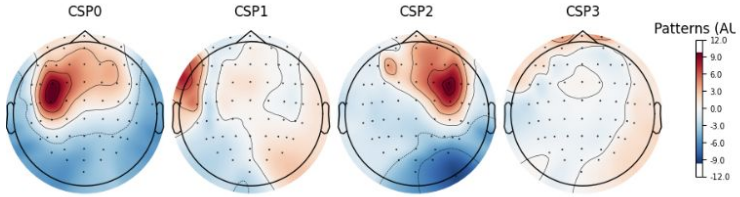
   Il est souvent utilisé en combinaison avec des classificateurs d’apprentissage automatique.

Application du CSP aux données EEG

1. **Application du CSP aux données EEG**
   Extraction des événements d'imagerie motrice
   Application d'un filtre passe-bande adapté aux tâches motrices, typiquement dans la plage 7–30 Hz.
   Identification et extraction des événements, par exemple : imaginer bouger les mains vs. les pieds.

2. **Application du CSP**
   Créer des époques centrées sur ces événements, avec une fenêtre temporelle suffisante pour capturer l'activité pertinente.
   Appliquer l'algorithme CSP à ces époques pour dériver des filtres spatiaux maximisant la différence de variance entre les deux conditions (mains vs. pieds) :

```
csp = CSP(n_components=4, reg=None, log=True)
csp.fit_transform(epochs.get_data(), labels)
```



3. **Visualisation des filtres et patrons CSP**
   Visualiser les filtres spatiaux et patrons dérivés du CSP.
   Les filtres se projettent sur le cuir chevelu, mettant en évidence les zones importantes pour distinguer les mouvements imaginés :

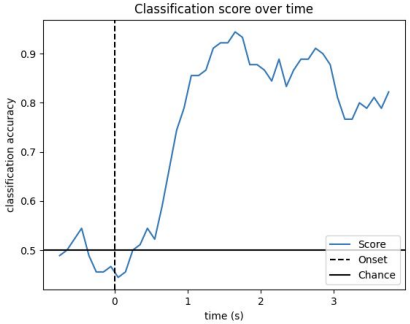
```
csp.plot_patterns(epochs.info)
```

Intégration du CSP avec la LDA

1. **Appliquer la LDA**
   Configurer un classificateur (LDA) utilisant les caractéristiques extraites par le CSP pour distinguer l'imagerie des mains vs. des pieds :
   ```
   clf = Pipeline([('CSP', CSP(n_components=4)), ('LDA', LinearDiscriminantAnalysis())])
   ```

2. **Mise en place de la validation croisée**
   Utiliser la méthode `ShuffleSplit` pour créer des divisions entraînement/test :
   ```
   cv = ShuffleSplit(n_splits=10, test_size=0.2, random_state=42)
   ```

3. **Évaluation des performances du classificateur**
   Évaluer la performance du classificateur sur différents sous-ensembles de données via la validation croisée,
   afin d'examiner la stabilité et la fiabilité du modèle.
   Calculer et visualiser les scores de validation croisée :
   ```
   scores = cross_val_score(clf, X, y, cv=cv)
   ```



MERCI

MNE
MEG + EEG ANALYSIS & VISUALIZATION

scikit learn
machine learning in Python

Repository :
https://github.com/thecocolab/mne_meeg_ml_main

MNE : https://mne.tools/stable/documentation/index.html

Présenté par:

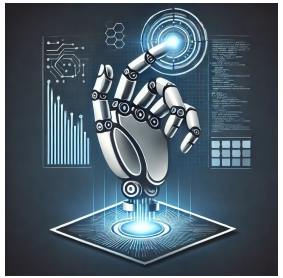
Hamza Abdelhedi, Vanessa Hadid

Yann Harel et Annalisa Pascarella

École d'été NeuroQAM - 13 mai 2025

Questions?

Section 5 : Activité pratique



1. **Exercices pratiques**
   Appliquer le prétraitement et l'extraction des époques sur le jeu de données téléchargé.
   Identifier les réponses évoquées liées à différents stimuli.
   Comparer les classificateurs en utilisant les caractéristiques extraites par CSP pour décoder l'imagerie motrice.
   Explorer le décodage temporel sur un jeu de données de votre choix pour identifier les moments de changement significatif dans l'activité cérébrale.

2. **Apprentissage continu**
   Nous vous encourageons à explorer toutes les possibilités offertes par MNE-Python et sa documentation/tutoriels en ligne très riches.
   Participez à des forums et communautés en ligne consacrés à la recherche EEG/MEG pour partager vos résultats, poser des questions et rester à jour sur les développements récents dans le domaine.