PLDAC - Etude de différentes techniques pour la classification de signaux d'EEG et MEG

Buton Nicolas

25 mai 2019

Table des matières

Première partie

Introduction

Les signaux étudiés seront des signaux d'Électroencéphalographie(EEG) et de Magnétoencéphalographie(MEG). Le premier consiste à enregistrer les signaux électriques à la surface du crane grâce a des électrodes, le second enregistre l'activité magnétique induite par l'activité des neurones grâce a un ensemble de magnétomètres. Notre étude ce place dans le cadre de l'analyse de série temporelle multivariée. On retrouve ces séries dans plusieurs domaines comme en finance, géophysique et dans notre cas dans les BCI (Brain Computer Interface). Chaque variable possède une dimension temporelle et de plus chaque variable est liée spatialement aux autres.

Une des premières difficultés des interfaces cerveaux machines c'est qu'elles ont besoin de réagir vite cela implique que l'on doit analyser le signal sur un laps de temps court, ce qui nous donne peu de données pour voir les liens entre les différentes variables. Le fait que les électrodes ne sont jamais situées exactement au même endroit sur le crane entre chaque essai et entre chaque personne rajoute une autre difficulté. De plus il peut y avoir beaucoup de bruits générés par le matériel et les mouvements de l'utilisateur par exemple.

Historiquement la classification sur de tels signaux se faisait avec des méthodes linéaires, par exemple avec des SVM ou des régressions logistiques [1]. Cela avait des performances assez faible et il fallait donc répéter l'expérience plusieurs fois. Par la suite Alexandre Barachant a proposé d'utiliser les méthodes riemanniennes [2], celles-ci permettent de prendre en compte la spatialité des données et sont plus robustes. Cette robustesse vient du fait que la géométrie de Riemann permet de manipuler des matrices de covariance. On peut définir une distance entre matrices symétriques définies positives et les matrices de covariance font partie de ce groupe. Ces méthodes ont permis d'attaquer des taches plus compliqués comme le transfert entre participants. On entraîne notre modèle en utilisant un participant pour faire ensuite des prédictions sur un autre participant.

C'est pour cela qu'aujourd'hui dans les méthodes de l'état de l'art on utilise encore la géométrie riemannienne. Par la suite nous allons comparer ces méthodes avec de la géométrie riemannienne avec d'autres méthodes plus classiques, ainsi qu'une méthode de deep learning (un réseaux convolutionel). Nous essayerons plusieurs combinaisons avec des méthodes issues du traitement du signal comme la transformée de Fourier,

des filtres et aussi d'autre façon de traiter les matrices de covariance avec des modèles comme des SVM ou KNN. Par ailleurs nous étudierons aussi l'effet de la taille de la fenêtre.

On pourra voir que pour des cas simples comme l'ouverture et la fermeture des yeux qui possèdent un signal très fort sur un EEG on a déjà de très bon résultats avec des méthodes simples, F1-score de 0.77 sur la classe minoritaire avec une transformée de Fourier puis un SVM avec une taille de fenêtre adaptée (plus de détails dans la partie expérimentation), et avec un "Minimum distance to Mean" et de la géométrie riemannienne on obtient un F1-score de 0.84.

Deuxième partie

État de l'art - la géométrie de Riemann

Nous avons besoins de métrique dans des espaces de Riemann pour pouvoir comparer les matrices entre elles. Cela nous permet de transposer des algorithmes comme les plus proches voisins ou les MDM, ou l'on définit un centre par classe et on regarde de quel centre est plus proche la nouvelle données. Et nous avons aussi besoin d'un façon de vectorisé cette matrice et c'est a cela que sert l'espace tangent.

1 Distance

Les matrices de covariance sont des matrices symétriques et définies positives. Pour deux matrices $\mathbf{\Sigma}_1$ et $\mathbf{\Sigma}_2$ leurs distances est d'après la géométrie de Riemann est déterminé par la formule suivante :

$$\delta_R(\mathbf{\Sigma}_1, \mathbf{\Sigma}_2) = \|\log\left(\mathbf{\Sigma}_1^{-1/2}\mathbf{\Sigma}_2\mathbf{\Sigma}_1^{-1/2}\right)\|_F = \left[\sum_{c=1}^{C}\log^2\lambda_c\right]^{1/2}, \tag{1}$$

où $\lambda_c, c = 1 \dots C$ sont les valeurs propres réelles de $\mathbf{\Sigma}_1^{-1/2}\mathbf{\Sigma}_2\mathbf{\Sigma}_1^{-1/2}$ et C le nombre d'électrodes.
F : Norme de Frobenius

2 Moyenne géométrique

Pour définir la matrice moyenne nous ne possédons pas d'expression explicite. Nous définissons donc une matrice $\mathbf{\Sigma}_i$, matrice moyenne que nous initialisons aléatoirement et ensuite on procède par descente de gradient en utilisant la formule suivante :

$$\mathfrak{G}\left(\mathbf{\Sigma}_1, \ldots, \mathbf{\Sigma}_I\right) = \arg\min_{\mathbf{\Sigma}} \sum_{i=1}^{I} \delta_R^2\left(\mathbf{\Sigma}, \mathbf{\Sigma}_i\right). \tag{2}$$

3 Espace tangent

On peut projeter un point de l'espace de Riemann définit par une matrice NxN sur un espace tangent avec N(N+1)/2 dimensions. Cet espace tangent est un espace euclidien. On peut approximer des distances de l'espace de Riemann par des distances dans l'espace tangent et cela fonctionne assez bien localement. Il a été montré que le point que l'on choisis pour définir notre espace tangent importe peu.

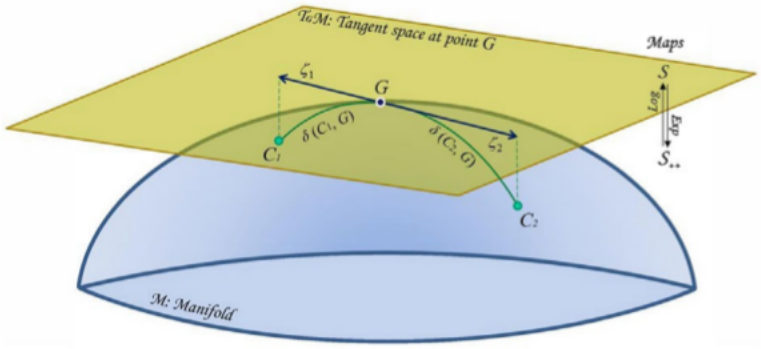


FIGURE 1: Affichage de l'espace tangent

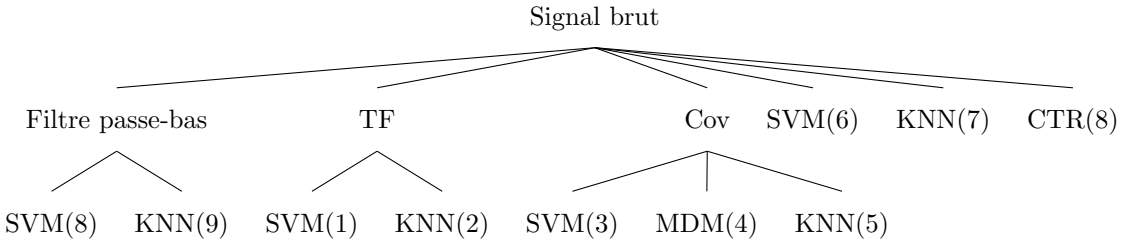
Pour plus de détails sur la géometrie de riemann il y a cet article d'A. Barachant [2].

Troisième partie

Plan d'expérience

1 Arbre des modèles

Nous allons représenter à l'aide d'un arbre les différentes méthodes que nous allons tester par la suite.



Légende :

Cov : Matrice de covariance

TF : Transformée de Fourier

MDM : Minimum Distance to Mean

CTR : Chaîne de traitement riemannienne, détaillée plus loin.

2 Format des données

En entrée deux formats sont possible. Le premier est une série temporelle pour chaque électrodes ainsi que les labels pour chaque donnée d'entrée.

Notons :

C : le nombre de canaux.

T : le temps écoulé en seconde pendant l'enregistrement des données.

f : la fréquence d'échantillonnage du signal.

D_E : durée d'un essai.

N_T : le nombre de trial.

Nous disposons donc d'une matrice de taille $C * (T * f)$, ainsi que d'un vecteur de label de taille $(T * f)$.

Mais un deuxième format d'entré est possible, en découpant par trial, a chaque trial correspond un label. Nous disposons donc d'un tenseur de taille $(D_E * f) * C * N_T$, et d'un vecteur de label de taille N_T.
On peut passer du premier formalisme a l'autre en découpant par trial (en définissant une taille de fenêtre) mais nous avons plusieurs possibilités pour savoir quel label associé à ce trial (label majoritaire,premier label,dernier label,etc...).

3 Description des modèles

Une donnée correspond à un essai. On le note $x \in \mathbb{R}^{C*D_E}$. Si les signaux sont présentés en continu il faut les découper en choisissant une taille de fenêtre pour les analysés.

1 et 2 - SVM et KNN avec une transformée de Fourier
La première étape de notre modèle est d'effectuer une transformée de Fourier, c'est a dire passer d'un signal temporel à un signal fréquentiel. Pour calculer cette transformation nous utiliserons l'algorithme FFT (Fast Fourier transform). Nous garderons uniquement le module pour rester avec des valeurs réelles plutôt que complexes. Ensuite nous utilisons un SVM linéaire pour prédire les labels dans un cas et un KNN euclidien dans l'autre. Les données apres pré-traitement $x_{fft} \in \mathbb{R}^{C*\left\lfloor \frac{D_E}{2} \right\rfloor}$

3,4 et 5 - SVM,MDM et KNN avec matrice de covariance
Pour cette approche nous devons passer au deuxième format de données. On calcule la matrice de covariance.$\Sigma = x^T * x \in \mathbb{R}^{C*C}$. Ces représentation sont reconnues très efficaces dans l'approche Riemannien. Nous allons vérifier ce que ça donne avec des approches plus classiques.

6 et 7 - SVM et KNN sur les données brutes
Cette approche est la plus simple et on utilise simplement un vecteur de la taille le nombre de capteur, c'est à dire un relevé a un instant t des capteurs pour prédire un labels.

8 et 9 - SVM et KNN avec un filtre passe bas
Nous définissons un filtre passe bas avec la moyenne sur un fenêtre glissante. Donc nous avons deux paramètres le premier est la longueur de la fenêtre, sur combien on moyenne et le deuxième est le décalage de cette fenêtre. On peut mettre la longueur de la fenêtre égale au décalage si on ne veux pas de chevauchement.

8 - Chaîne de traitement riemannienne

Par la suite je vais décrire la chaîne de traitement étape par étape. Cette chaîne de traitement est spécifique au dataset DecMeg2014.

1 - On garde uniquement 1 seconde sur les 1.5 seconde de signal car le stimulus intervient qu'après 0.5 seconde.

2 - On filtre le signal avec un filtre passe bande de Butterworth d'ordre 5 entre 1Hz et 20Hz.

3 - Par la suite on effectue un filtrage spatial, on extrait C_{virt} canaux virtuels par classe et K le nombre de classes [**?**]. On commence par définir la moyenne sur chaque classe :

$$\mathbf{P}^{(k)} = \frac{1}{|\mathcal{I}^{(k)}|} \sum_{i \in \mathcal{I}^{(k)}} \mathbf{X}_{i_1}$$

Ensuite notre but est d'optimiser w par descente de gradient. En minimisant cette fonction :

$$f(X, w) = -\frac{\mathbf{w}^T \mathbf{P}^{(k)} \mathbf{P}^{(k)T} \mathbf{w}}{\mathbf{w}^T \mathbf{X} \mathbf{X}^T \mathbf{w}}$$

On introduit les Zi :

$$\mathbf{Z}_i = \mathbf{W}^T \mathbf{X}_i$$

4 - On définit une nouvelle entrée comme étant la concaténation :

- du signal moyen de la classe 1 auquel on multiplie par W0 pour les projeter sur les canaux virtuels appris précédemment.
- idem pour la classe 2
- Et ensuite le signal Xi projeté sur les 8 canaux, qui est Zi.

Cela nous permet de définir de nouvelles caractéristiques, donc a chaque Xi d'entrée sur C canaux on le projette sur Cvirt canaux.

$$\tilde{\boldsymbol{Z}}_i = \begin{bmatrix} \mathbf{W}^{(0)T} \mathbf{P}^{(0)} \\ \mathbf{W}^{(1)T} \mathbf{P}^{(1)} \\ \mathbf{Z}_i \end{bmatrix}$$

5 - On calcule la covariance spatiale de cette nouvelle matrice de caractéristique.

$$\mathbf{\Sigma}_i = \frac{1}{N} \tilde{\mathbf{Z}}_i \tilde{\mathbf{Z}}_i^T$$

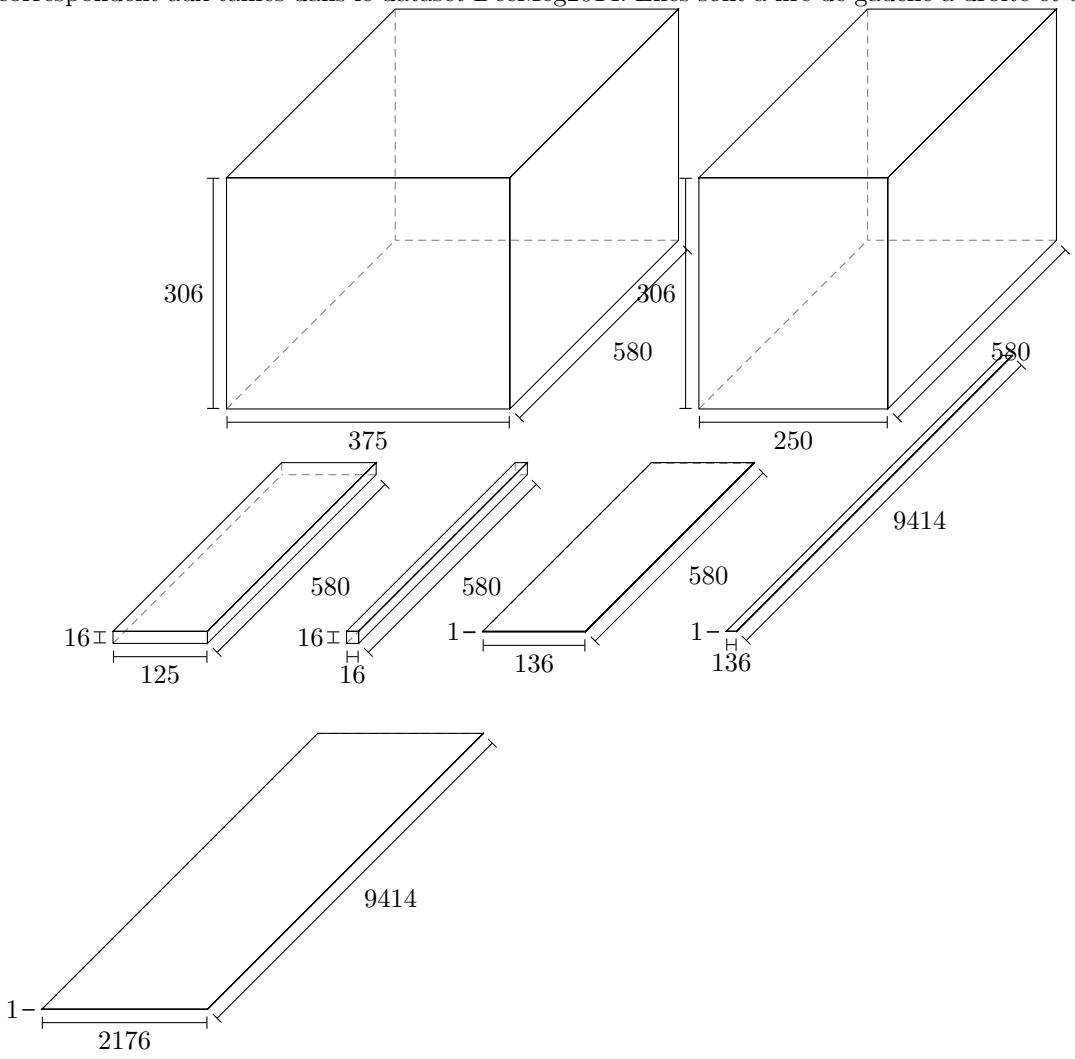
6 - On projette cette matrice sur l'espace tangent.

7 - On estime pour les 15 autres participants avec ce filtre spatial

8 - On fait la même chose pour les 15 autres participants et on concatène.

9 - On fait une régression logistique avec une régularisation lasso.

La figure suivante représente la forme des matrices au fur et à mesure du processus. Les tailles des matrices correspondent aux tailles dans le dataset DecMeg2014. Elles sont à lire de gauche à droite et de haut en bas.



Quatrième partie

Experimentations

Nous allons étudier nos méthodes sur 3 datasets différents : Eye close/eye open, brain invader et Dec-Meg2014. Dans un premier temps nous regarderons la valeur de K optimale pour les différents KNN, par la suite nous nous intéresserons à la taille de la fenêtre optimale, puis pour finir quelles méthodes fonctionnent le mieux selon la situation. Tout le code et les résultats correspondant aux expériences peuvent être trouvés sur github. [1]

1 Description des différents dataset

1.1 Description du dataset eye close/eye open

Le dataset eye close/eye open contient les données enregistrées avec un casque EEG où l'on a demandé aux participants d'ouvrir ou de fermer les yeux a certain moment. La tâche a accomplir est de classifier à chaque enregistrement si la personne a les yeux ouverts ou fermés.

Le signal est échantillonné à 512Hz. Il y a 7 femmes et 13 hommes pour un total de 20 participants pour ce dataset. L'age moyen est de 25.8 ans avec un écart type de 5.27 et une médiane a 25.5 ans. 18 participants ont entre 19 et 28 ans et deux participants ont respectivement 33 ans et 44 ans.Le casque d'enregistrement est composé de 16 électrodes.

On commence par visualiser le signal des 16 électrodes ainsi que leurs labels associés au cours du temps. On peut voir que cette tâche, même si c'est une tâche relativement facile pour une machine comme nous allons le voir plus loin, elle reste très difficile voir impossible pour un humain.

1. https ://github.com/rootNico/PLDAC

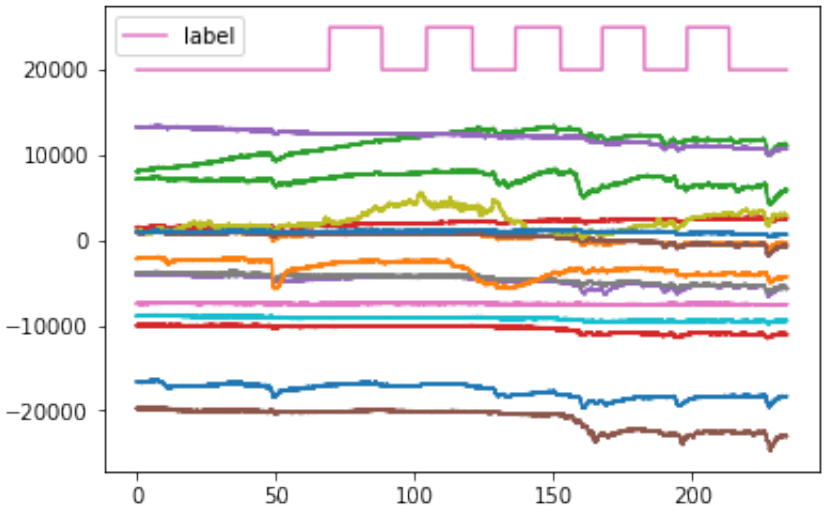


FIGURE 2: Affichage des données brutes

1.2 Description du dataset brain invader

Il y a 24 participants qui ont participé à l'éxperience (12 femmes), avec une moyenne d'age (écart type) de 25.96 (4.46). Les 12 électrodes en vert sur l'image ci-dessous était utilisé.

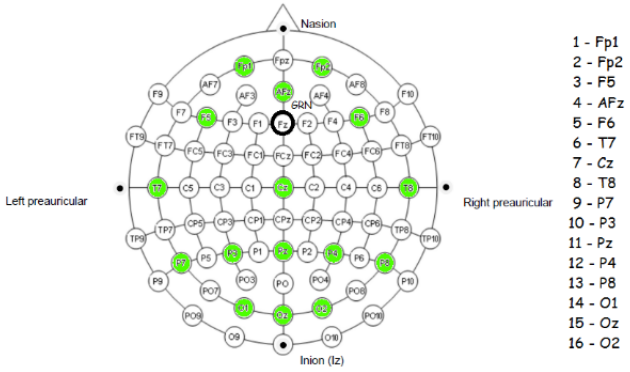


FIGURE 3: Affichage des données brutes

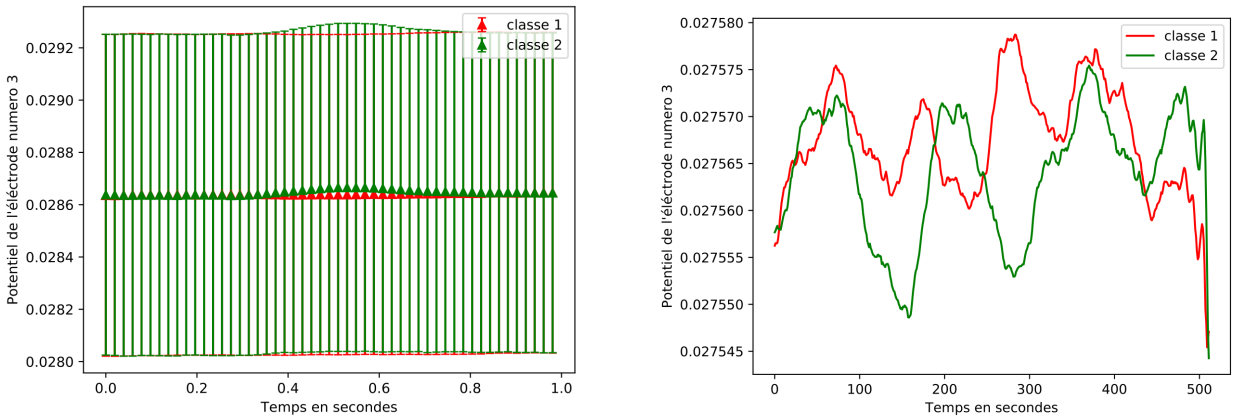
Ce dataset a été enregistré avec les participants placés devant un pc ou une grille d'alien était représenté sur un écran avec un alien en particulier qui était en rouge. Le but pour le participant est de viser l'alien rouge. Pour cela 12 flashs dont 2 comprennent l'alien ciblé sont émis, le but de l'algorithme est de voir par quel flash le participant est intéressé. Un flash est composé de 6 alien. En tout il y avait 36 alien sur l'écran, une grille de 6X6. Il y a plusieurs tentatives pour détruire l'alien.
Les deux tâches suivantes sont à effectuer mais comme nous allons le voir dans la partie résultat nous n'avons pas toute les informations pour traiter la deuxième tache.
Première tache : Classification binaire (le flash nous intéresse, s'il y a l'alien cible dedans)
Deuxième tache : Il nous faut trouver les deux flashs comprenant l'alien ciblé dans le groupe des 12 flashs.

Plusieurs dataset utilisent ce paradigme de P300. On présente un stimulus a un participant et environ 300 ms après on peut détecter une activité cérébrale. En le répétant plusieurs fois cela permet a l'utilisateur de faire un choix par la pensée.

Figure 4 : Par la suite on peut voir sur l'image de droite le potentiel de l'électrode numéro 3 selon la classe au cours du temps. Et pour se rendre un peu mieux compte de la difficulté de la tâche on peut voir sur le graphique de gauche la moyenne sur tous les essais avec l'écart type, on peut différencier les deux classes mais l'écart entre les moyennes est très petit en comparaison des grandes valeurs des écarts types. Les graphiques sur les autres électrodes montrent tous à peu prêt la même chose ou parfois même encore moins d'écart de moyenne entre les deux classes.

1.3 Description du dataset DecMeg2014

La tâche que l'on doit réaliser avec le dataset DecMeg2014 est une classification binaire. Le but est de déterminer si le stimulus visuel est un visage clair ou brouillé qui est montré au participant. L'activité de leurs cerveaux est enregistré grâce à un appareil de magnetoencéphalographie. Cet appareil dispose de 306 magnétomètres.
Ce dataset est extrait d'une compétition kaggle du même nom.
Il y a eu 23 participants à ce test avec environ 580 essais par participant. Nous disposons de 16 participants avec leurs labels associé et 7 participants où nous avons uniquement les données.



(a) visuel classe mean std

(b) visuel data 0-1

FIGURE 4: Visualisation des classes

2 Étude sur les valeurs de k pour les différents KNN

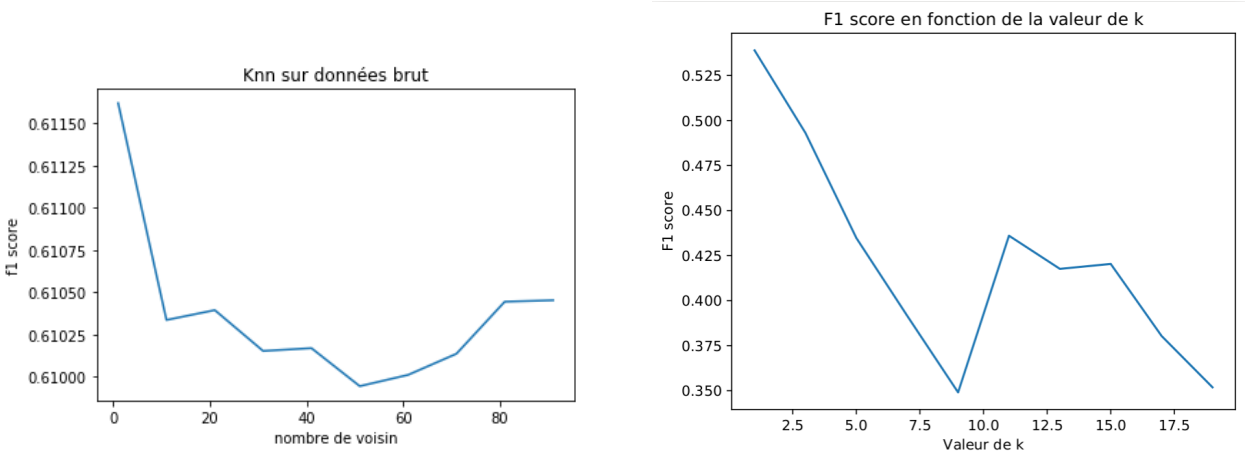
On peut voir que la valeur de K=1 est meilleur.

3 Étude sur la longueur de la fenêtre optimale

Les résultats sont obtenus avec une cross validation sur tous les participants mais en ne prenant en compte qu'un fichier par participant pour une question de mémoire vive. F1-score des différents algorithmes sur brain invader :

nom de l'algorithme	1 seconde	0,1 seconde	0,04 seconde	moyenne
passe bas KNN	0,002487562189055	0,126738794435858	0,004889975550122	0,044705444058345
knn tf	0,019002375296912	0,029919447640967	0,124804992199688	0,057908938379189
knn brut	0,071287128712871	0,104810996563574	0,027681660899654	0,067926595392033
SVM brut	0,181818181818182	0	0,214285714285714	0,132034632034632
cov SVM	0	0,181818181818182	0,235294117647059	0,13903743315508
SVM tf	0,125	0,125	0,181818181818182	0,143939393939394
passe bas SVM	0,214285714285714	0,214285714285714	0,235294117647059	0,221288515406162
riemann MDM	0,269889224572004	0,256206554121152	0,249661705006766	0,258585827899974
moyenne	0,110471273359342	0,129847461108181	0,159216308131781	0,133178347533101

TABLE 1: Résultat sur le dataset brain invader 1 participant



(a) knn sur les données brutese

(b) Riemann knn

FIGURE 5: F1 Score(en cross validation) en fonction du nombre de voisin

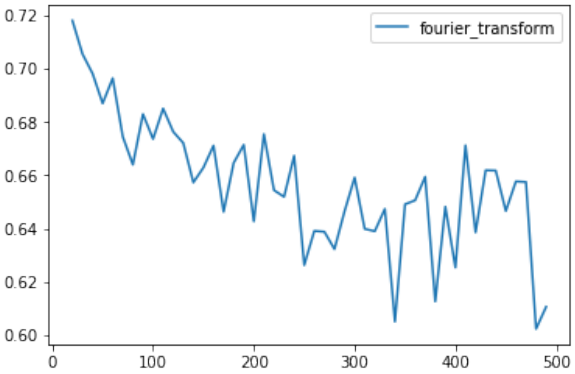
Dans les figures ci-dessous on remarque qu'avec des tailles de fenêtre très petite on a de très bon résultat mais cela vient du fait qu'on utilise des algorithmes tel que le KNN et le MDM et que de plus on est sur un seul participant, donc on va juste chercher à faire correspondre les signaux qui sont proche temporellement et comme on modifie les labels par pallier, cela donne de bon résultat de retrouvé les voisins proches et dire le même label. Et pour le MDM c'est la même chose on trouve un centre géométrique qui fonctionne mais uniquement pour ce participant. Je pense qu'on ne peut pas tirer assez d'information sur les corrélations des différents canaux pour faire du transfert, je n'ai malheureusement pas eu le temps de faire les tests.

4 Comparaison des différentes méthodes

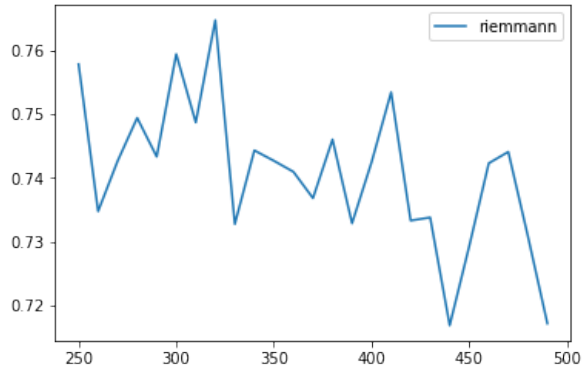
Tous les résultats suivant ont été obtenu avec les données d'un seul participant, avec une validation croisée en 5 parties.

4.1 Tableau récapitulatif eye close/open

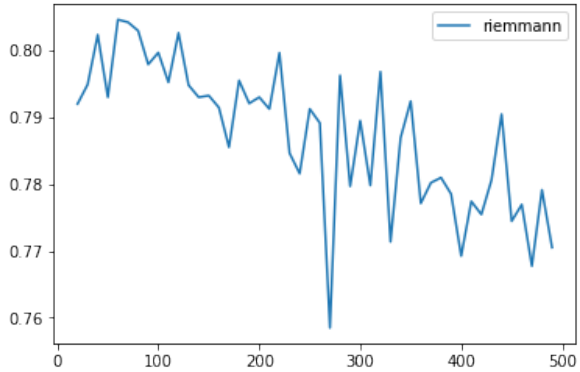
Ci-dessous nous avons le tableau récapitulatif des F1-score en validation croisée en 5 parties sur les données d'un seul participant . On remarque que les approches de Riemann ont les meilleurs scores et ensuite le SVM avec transformée de Fourier.



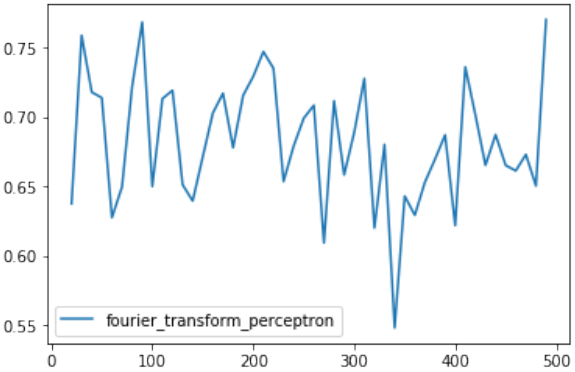
(a) KNN transformée de fourier



(b) Riemann Cov KNN



(c) riemann MDM



(d) SVM transformée de fourier

FIGURE 6: F1 Score(en cross validation) en fonction du nombre de données par paquet

Nom de l'algorithme	f1-score
SVM sur les données brutes	0.54
KNN sur les données brutes	0.61
Riemann Cov MDM	0.84
Riemann Cov KNN	0.77
SVM filtre passe bas	0.60
KNN filtre passe bas	0.59
SVM transformée de Fourier	0.77
KNN transformée de Fourier	0.72

TABLE 2: Résultat sur le dataset eye close/open

4.2 Brain Invader

Comme on peut le voir(ref : Table 1) tout les algorithmes ne sont pas présent, certain était trop long à faire tourner et d'autre n'avais pas encore été introduit et le seront pour le dataset DecMeg2014 qui suit. Aucun prétraitement n'a été fait sur le signal pour la géométrie de Riemann et cela change beaucoup les résultats juste un filtre passe bande peut améliorer les résultats comme on va le voir par la suite. C'est aussi pour cela qu'on retrouve un SVM en deuxième position. Il n'a pas été possible de comparer les résultats avec les résultats de l'auteur du dataset car il manquait la composition des groupe d'alien qui clignotait.

4.3 DecMeg2014

Méthodes d'évaluation Sur le dernier dataset DecMeg2014 nous disposons des données pour 16 participants, nous procéderons donc comme décris ci-dessous pour l'évaluation de nos différentes méthodes :
Méthode 1 : entraînement sur une partie de 1 et eval sur 1 (validation croisée sur un seul)
Méthode 2 : entraînement sur 1 et évaluation sur tous les autres (16 expériences a faire)
Méthode 3 : entraînement sur 15 et évaluation sur 1 (16 expériences a faire)
Méthode 4 : entraînement sur 16 et eval sur 16 tout mélanger avec validation croisée.
Ces façons d'évaluer nous permettent de tester plusieurs caractéristiques de nos modèles dont la capacité de transfert d'un participant à l'autre.
Pour chaque test on sauvegardera le F1-score sur la classe minoritaire.

Nom de l'algorithme	Méthode 1	Méthode 2	Méthode 3	Méthode 4
CTR	0,717	0.182	0.673	0.693
conv1D	0,133	0.04	0.09	0.26
passe bas SVM	0.0	0.0	0.0	0
riemann MDM	0,5524	0.12	0.43	0.49
SVM brut	0.0	0.0	0	0.0
SVM TF	0.0	0.0	0.0	0.0

Cinquième partie

Conclusion

D'après nos résultat nous pouvons conclure que d'une part les prétraitement sur le signal sont très important et que pour le moment avec des méthodes classiques de deep learning on n'arrive pas à faire mieux qu'avec des méthodes riemanniennes. Il pourrait être intéressant de combiner des méthodes de deep learning et de géométrie de Riemann.

Références

[1] G. N. Garcia, T. Ebrahimi, and J. Vesin, "Support vector EEG classification in the fourier and time-frequency correlation domains," pp. 591–594.

[2] M. Congedo, A. Barachant, and R. Bhatia, "Riemannian geometry for EEG-based brain-computer interfaces ; a primer and a review," vol. 4, no. 3, pp. 155–174.