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

Buton Nicolas

May 18, 2019

Table des matières

Part I

Introduction

Les signaux étudié seront des signaux d'Électroencéphalographie(EEG) et de Magnétoencéphalographie(MEG). Le premier consiste a enregistrer les signaux électrique a 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 magnétomètre.Notre étude ce place dans le cadre de l'analyse de série temporelle multivarié. On retrouve ces séries dans plusieurs domaine comme en finance, géophysique et dans notre cas dans les BCI(Brain Computer Interface). Chaque variable possède une dimension temporel et de plus chaque variable est lié au niveau spatial aux autres.

Une des premières difficulté des interfaces cerveaux machines c'est qu'elle ont besoin de réagir vite et donc on doit analyser le signal sur un laps de temps court, ce qui nous donne peut de données pour voir les liens entre les différentes variables. Une autres difficultés pour traiter ces signaux est le fait que les électrodes ne sont jamais situées exactement au même endroit sur le crane entre chaque essai, et entre chaque personne. De plus il peut y avoir beaucoup de bruit généré par le matériel ou les mouvement de l'utilisateur par exemple.

Historiquement la classification sur de tels signaux ce faisait avec des méthodes linéaire, par exemple avec des SVM, regression logistique. Cela avait des performances assez faible et il fallait donc répéter l'expérience plusieurs fois. Par la suite Alexandre Barachant à proposé d'utilisé les méthodes riemanniennes, celle ci permette de prendre en compte la spatialité de données et sont plus robuste. Cette robustesse vient du fait que la géométrie de Riemann permet de manipuler des matrice de covariance, car on peut définir une distance entre matrice symetrique définit positive, et les matrices de covariance font partie de ce groupe. Ces méthodes ont permit d'attaquer des taches plus compliqué comme le transfert entre sujet. On entraîne notre modèle en utilisant un sujet et par la suite on peut faire des prédiction sur un autre sujet.

C'est pour cela qu'aujourd'hui dans les méthodes de l'état de l'art utilise encore la géométrie riemannienne. Par la suite nous allons comparer ces méthodes avec de la géométrie riemannienne avec d'autres méthodes plus classique mais par forcément toute exploré, ainsi qu'une méthode de deep learning(un réseaux convolutionel). Nous essayerons plusieurs combinaison avec des méthodes d'annalyse de signal comme la transformer de fourier, des filtres et aussi d'autre façon de traiter les matrices de covariance, ainsi que des modeles comme des SVM et KNN.

On pourra voir que pour des cas simple comme l'ouverture et la fermeture des yeux qui possede un singal trés fort sur un EEG on a deja de trés bon resultat avec des méthodes simple, f1-score de 0.77 sur la classe minoriataire avec une transformer de fourier puis un SVM et une taille de fenettre adapté(détaillé dans la partie expériementation), et avec un Minimum distance to Mean et de la geometrie riemannienne on obtient un F1-score de 0.84.

Ps : Chanel et canaux seront utilisé de façon interchangeable dans la suite de ce document.

Part II

État de l'art - la géometrie de riemann

1 Distance

Les matrices de covariance sont des matrices symétrique et définit positive. Pour deux matrice $\mathbf{\Sigma}_1$ et $\mathbf{\Sigma}_2$ leurs distances est d'après la géométrie de Riemann est la 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 géometrique

Pour définir la matrice moyenne nous ne possédons pas d'expression explicite. Fréchet mean

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

Pour la calculer on peut utiliser une descente de gradient.

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 distance dans l'espace tangent et cela fonctionne assez bien localement.

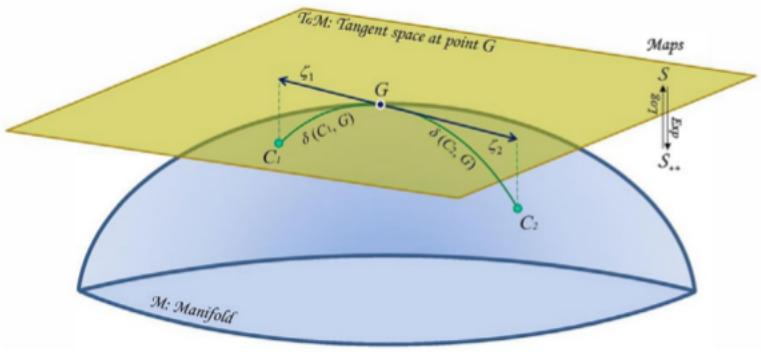


Figure 1: Affichage del'espace tangent

Article sur la géométrie de Riemann :

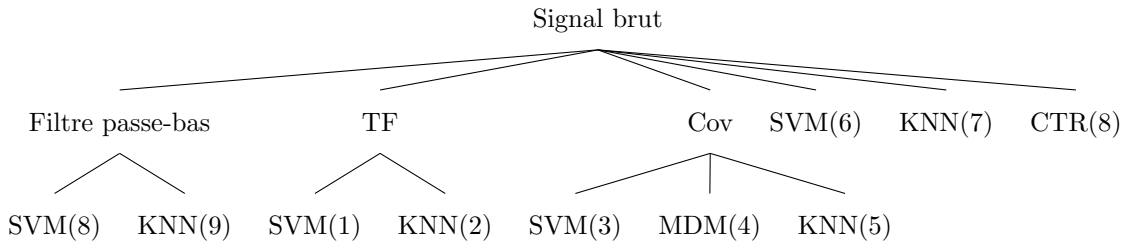
- Brain invader [1]
- Geometrie de riemann [2]

Part III

Les différents modèles

1 Arbre des modèles

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



Légende :

Cov : Matrice de covariance

TF : Transformé de fourier

MDM : Minimum Distance to Mean

CTR : Chaine de traitement riemannienne, détaillé plus loin.

2 Format des données

En entrée deux format sont possible. Le premier est une serie temporel pour chaque éléctrodes ainssi que les labels pour chaque données 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.

TT : le temps d'un trial.

NT : le nombre de trial.

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

Mais un deuxieme format d'entrée est possible, en découpant par trial, a chaque trial correspond un label.

Nous disposons donc d'un tenseur de taille (TT*f)xCxNT, et d'un vecteur de label de taille NT.

On peut passer du premier formalisme a l'autre en découpant par trial(en définissant une taille de fenêtre) mais nous avons plusieurs possibilité pour savoir quels labels associé a ce trial (label majoritaire,premier label,dernier label,etc...).

3 Description des modèles

3.1 1 et 2 - SVM et KNN avec une transformer de fourrier

La première étape de notre modèle est d'effectuer une transformer de fourrier, c'est a dire passé d'un signal temporel à un signal fréquentiel. Pour calculer nous utiliserons l'algorithme FFT(Fast fourier transform).Nous garderons uniquement le modules pour rester avec des valeurs réelles plutôt que complexe.

Ensuite nous utilisons un SVM pour prédire les labels dans un cas et un KNN dans l'autre.

3.2 3,4 et 5 - SVM,MDM et KNN avec matrice de covariance

Pour cette approche nous devons passer au deuxième format de données(décrit dans la partie précédente), si les données nous sont donné dans le premier format nous devons définir une taille de fenêtre et une méthode d'attribution des labels.
Ensuite on calcul la matrice de covariance :
$\mathbf{\Sigma}_i = \frac{1}{C}\tilde{\mathbf{X}}\overline{\mathbf{X}}^T$ Avec X ma matrice d'entrée.

3.3 6 et 7 - SVM et KNN sur les données brut

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.

3.4 8 et 9 - SVM et KNN avec un filtre passe bas

Nous définissons un filtre passe bas avec la moyenne sur un fennettre glissante. Donc nous avons deux paramettre le premier est la longueur de la fennetre, sur combien on moyenne et le deuxieme est le decallage de cette fennettre. On peut mettre la longueur de la fennettre egale au decalage si on ne veux pas de chevauchement.

3.5 8 - Chaine de traitement riemannienne

Par la suite je vais décrire la chaine de traitement etape par etape.
1 - On garde uniquement 1 secondes sur les 1.5 secondes de signal car le stimulus intervient qu'après 0.5 secondes.
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 K canaux virtuel par classe.
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 les W par descente de gradient. En maximisant cette fonction :

$$\mathbf{w}^* = \arg\max_{\mathbf{w}} \frac{\mathbf{w}^T \mathbf{P}^{(k)} \mathbf{P}^{(k)T} \mathbf{w}}{\mathbf{w}^T \mathbf{X}\mathbf{X}^T \mathbf{w}}$$

On introduit les Zi.

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

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

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

Et l'on calcul la covariance de cette nouvelle matrice de caractéristique.

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

4 - On définit une nouvelle entrée comme étant la concaténation :
- du signal moyen de la classe 1 auquel on multiple par W0 pour les projeter sur les channels virtuelles appris précédemment.
- idem pour la classe 2
- Et ensuite le signal Xi projeter sur les 8 channels.

5 - On calcul la covariance de cette matrice.

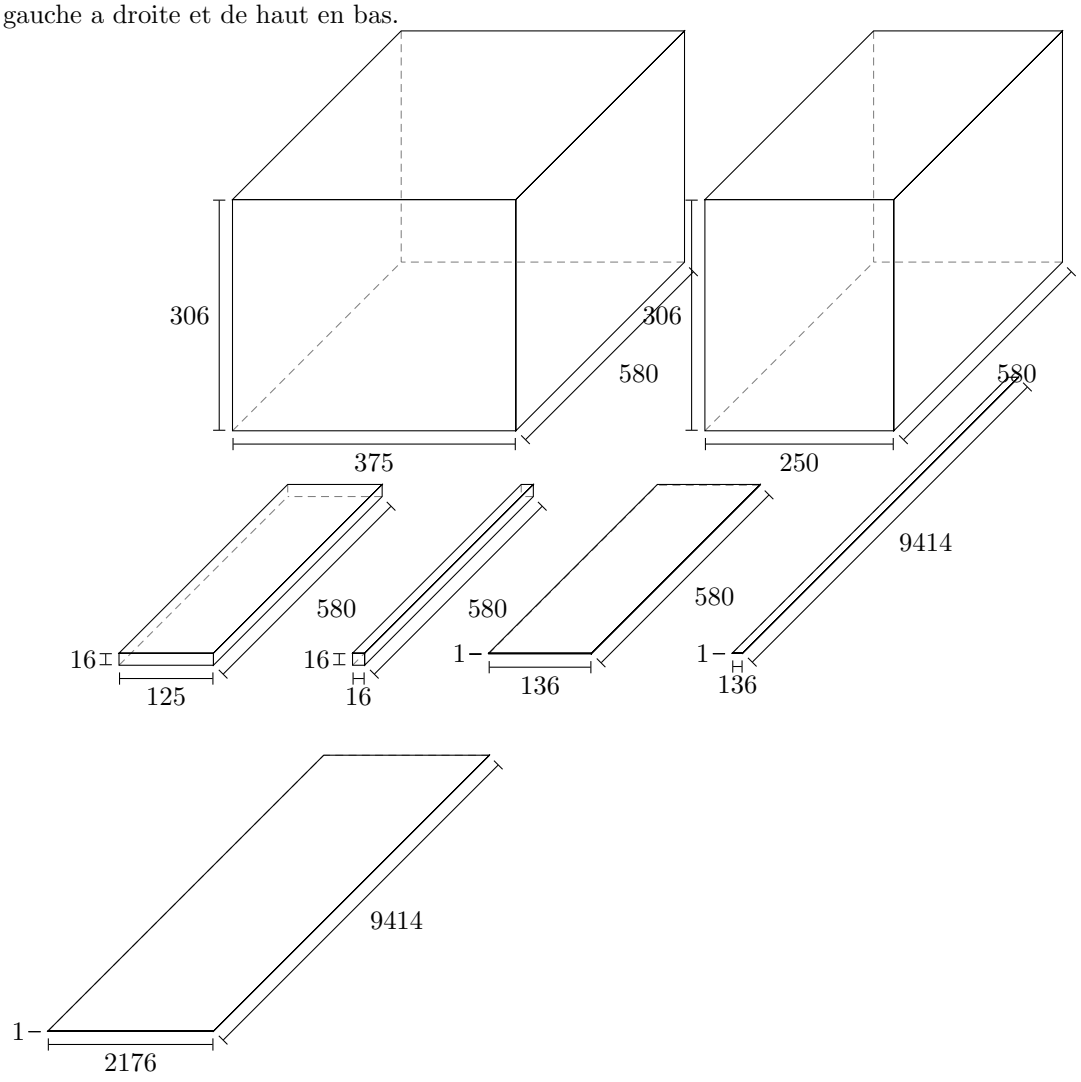
6 - On projette cette matrice sur l'espace tangent.

7 - On estime pour les 15 autre sujets avec ce filtre spatial 8 - On fait la même chose pour les 15 autres sujet et on concaténe.

9 - Régression logistique avec une régularisation lasso.

Les figures suivantes représentes la forme des matrices au fur et a mesure du processus. Elles sont a lire de

gauche a droite et de haut en bas.



Part IV

Experimentations

Dans ce rapport nous étudierons nos méthodes sur 3 datasets différents : Eye close/eye open, brain invader et DecMeg2014.

Tout le code et les résulats correspondant aux expérience peuvent etre trouvés sur github. [1]

1 Prise en main sur un dataset simple

1.1 Decription du dataset eye close/eye open

Ce dataset contient les données enregistré avec un casque EEG ou l'on a demandé au sujet d'ouvrir ou de fermer les yeux a certain moment. La tache a accomplir est de classifier a chaque enregistrement si la personne a les yeux ouvert ou fermé.

Le signal est échantillonné à 512Hz. Il y a 7 femmes et 13 hommes pour un total de 20 participant 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 sujets 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 associé au cours du temps.

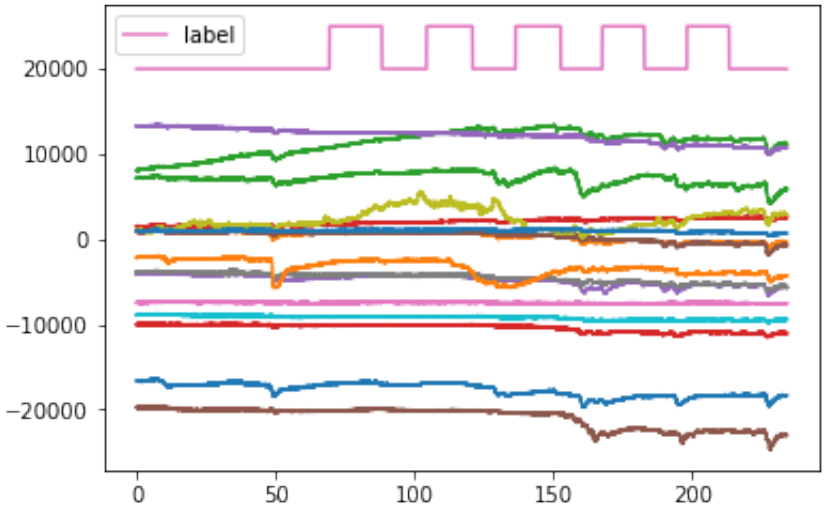


Figure 2: Affichage des données brut

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

1.2 Résultat des différents algorithmes

Prédiction théorique :
La méthode 6 et 7 ne devrais pas fonctionner car avec une seule données c'est difficile de faire quoi que ce soit.
La méthode 8 et 9 ne devrais pas fonctionner car il n'y aura pas invariance par translation.

Toute les resulats suivant on été obtenu avec les données d'un seul sujet, avec une validation croisée en 5 parties.

1.2.1 Valeurs de k pour KNN

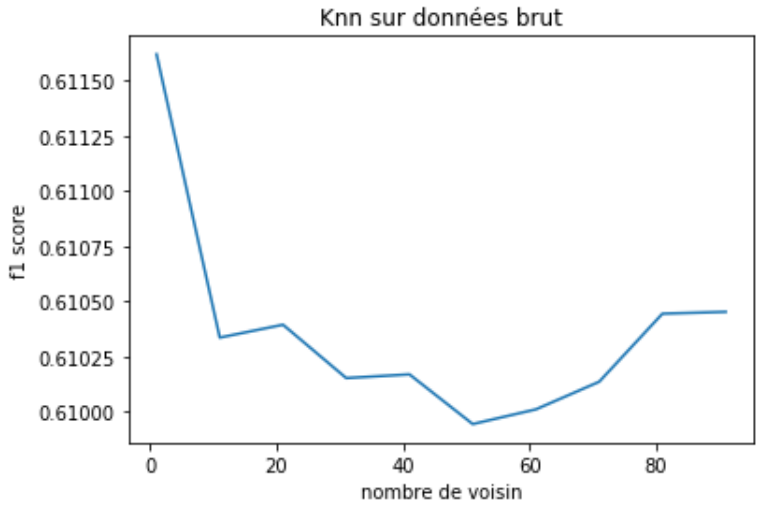
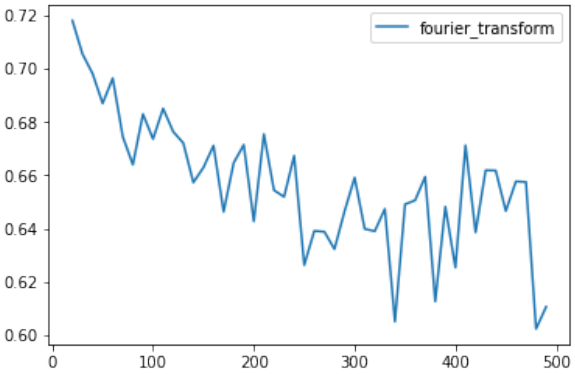


Figure 3: F1 Score(en cross validation) du knn sur les données brute en fonction du nombre de voisin

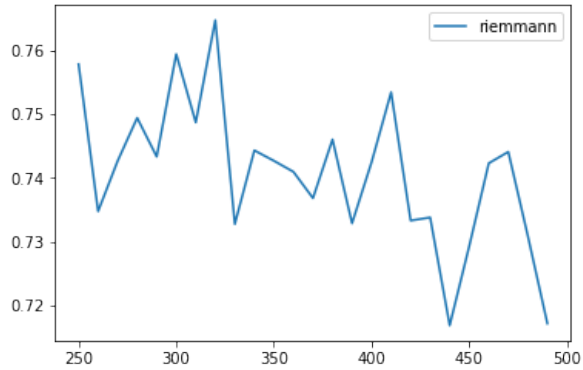
On peut voir que la valeur de K=1 est meilleur.

1.2.2 Études sur la longueur de la fenêtre optimale

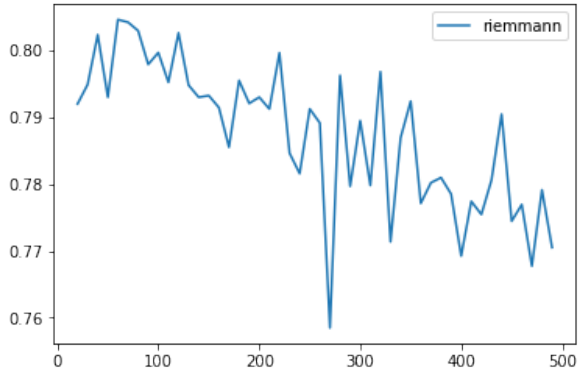
On remarque qu'avec des tailles de fennettre trés petite on a de trés bon résultat mais cela vient du fait qu'on utilise des algorithme tel que le KNN et le MDM qui



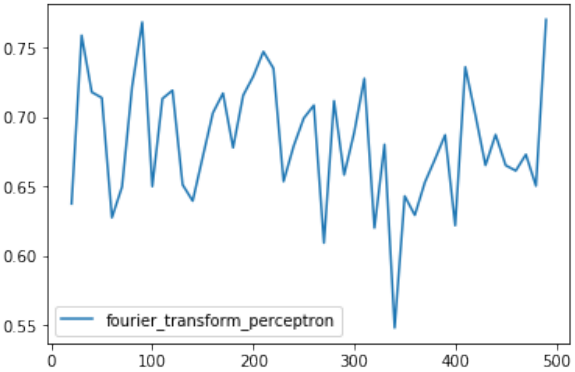
(a) KNN transformée de fourier



(b) Riemann Cov KNN



(c) riemann MDM



(d) SVM transformée de fourier

Figure 4: F1 Score(en cross validation) en fonction du nombre de données par paquet

1.2.3 Tableau récapiltulatif

f1-score arrondie à deux chiffre apres la virgule.

Nom de l'algorithme	f1-score
SVM sur les données brut	0.54
KNN sur les données brut	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

2 Brain Invader

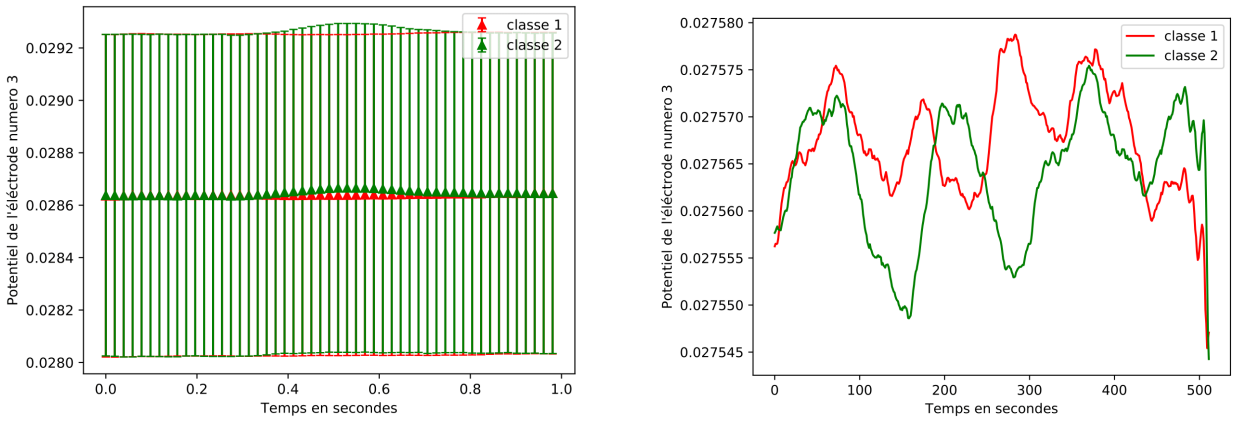
2.1 Description de la tache de machine learning

Ce dataset à été enregistrer avec les sujets placer devant un pc ou une grille d'alien était représenté sur un écran.12 flashs dont 2 comprennent l'alien ciblé. Détecté les flash ou il y a l'alien en regardant l'activité cérébrale.Plusieurs tentative pour détruire l'alien.
Première tache : Classification binaire (le flash nous intéresse(il y a l'alien cible dedans) ou pas)
Deuxième tache : dans le groupe de 12 ou sont les deux flashs avec l'alien cible.

F1-score des différents algorithme sur brain invader :
C'est resultat sont obtenu avec une cross validation sur tout les sujets mais en ne prenant en compte qu'un fichier par sujet pour une question de mémoire.

Comme on peut le voir tout les algorithme de sont pas présent, certain etait trop long a faire tournée et d'autre n'avais pas encore été introduit et le seront pour le dataset DecMeg2014 qui suit.

Aucun pré traitement n'a été fait sur le signal pour la géometrie de riemann et cela change beaucoup les resultats juste un filtre passe bande peut amélioré les resultats comme on va le voir par la suite. C'est aussi pour cela qu'on retrouve un SVM en deuxieme position.



(a) visuel classe mean std

(b) visuel data 0-1

Figure 5: Visualisation des classes

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

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

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.

3 DecMeg2014

3.1 Description du dataset

La tache 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 un brouiller qui est montré au participant.L'activité de leurs cerveaux est enregistrer grâce à un appareil de magnetoencéphalographie. Cet appareil dispose de 306 magnétomètres.
Ce dataset est extrait d'une compétition kaggle du même nom.
23 sujets on participé à ce test avec environ 580 trials par sujet. Nous disposons de 16 sujets avec leurs labels associé et 7 sujets ou nous avons uniquement les données.

3.2 Méthode d'évaluation

Sur le dernier dataset DecMeg2014 nous disposons des données pour 16 sujets, nous procéderons donc comme décris ci-dessous pour l'évaluation de nos différentes méthodes :
Méthode 1 : entrainement sur une partie de 1 et eval sur 1 (validation croisée sur un seul)
Méthode 2 : entrainement sur 1 et évaluation sur tous les autres (16 expériences a faire)
Méthode 3 : entrainement sur 15 et évaluation sur 1 (16 expériences a faire)
Méthode 4 : entrainement sur 16 et eval sur 16 tout mélanger avec validation croisée.
Ces façon d'évaluer nous permette de tester plusieurs caractéristique de nos modèles dont la capacité de transfert d'un sujet a l'autre.
Pour chaque test on sauvegardera le f1 score sur la classe minoritaire.

Part V

Conclusion

D'après nos résultat nous pouvons conclure que d'une part les pré traitement sur le signal sont trés important et que pour le moment avec des méthodes classique de deep learning on n'arrive pas a faire mieux qu'avec des méthodes riemanniennes. Il pourrais etre interessant de combiner des methodes de deep learning et de géometrie de riemann.

References

[1] M. Congedo, M. Goyat, N. Tarrin, G. Ionescu, L. Varnet, B. Rivet, R. Phlypo, N. Jrad, M. Acquadro, and C. Jutten, ""brain invaders": a prototype of an open-source p300- based video game working with the OpenViBE platform," p. 7.

[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.