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

Buton Nicolas

May 16, 2019

Table des matières

I Introduction **1**

II La géometrie de riemann **1**

1 Distance **1**

2 Point moyen **2**

3 Espace tangent **2**

III Prise en main sur un dataset simple **2**

1 Decription du dataset eye close/eye open **3**

2 Les diférentes méthodes **4**

3 Résultat des différents algorithmes **4**

3.1 les données brut 4
3.2 KNN sur les données brut 5
3.3 Riemann Cov MDM 5
3.4 Riemann Cov KNN 6
3.5 Perceptron filtre passe bas 7
3.6 KNN filtre passe bas 8
3.7 Perceptron transformée de fourier 9
3.8 KNN transformée de fourier 10
3.9 Tableau récapiltulatif 10

IV Brain Invader **10**

1 Description de la tache de machine learning **11**

V DecMeg2014 **12**

1 Description du dataset **12**

2 Les différentes étapes **12**

VI Conclusion **15**

Part I
Introduction

Les signaux étudié seront des signaux d'Électroencéphalographie(EEG) et de Magnétoencéphalographie(MEG). Le premier consiste a enregistrer les signaux elesctrique a la surface du crane grace a des éléctrodes, le second enregistre l'activité magnétique induite par l'activité des neurones grace a un ensemble de magnétometre.

L'une des difficulter pour traiter ces signaux est que les electrodes ne sont jamais situer exactement au meme endroit sur le crane entre chaque essai, et chaque personne. De plus il peut avoir beaucoup de bruit generer par le materiel ou les mouvement de l'utilisateur par exemple.

Pour etre resitant au bruit on a donc besoin de manipulier des matrices de covariance et pour cela un moyen est d'utiliser la géometrie de riemann qui peut définir une distance entre matrice definit positive, les matrices de covariance font partie de ce groupe.

C'est pour cela que dans les methodes de l'etat de l'art la géometrie riemanienne est utilisé. Ces méthode permette aussi un meilleur succées pour le transfert d'un sujet à un autre.

Par la suite nous allons comparer ces méthodes avec de la géometrie riemannienne avec d'autres methodes plus classique, ainssi qu'une methode de deep learning.

Dans ce rapport nous etuidirons nos méthodes sur 3 datasets différents : Eye close/eye open, brain invader et DecMeg2014.
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 :
- entrainement sur une partie de 1 et eval sur 1 (validation croisée sur un seul)
- entrainement sur 1 et évaluation sur tous les autres (16 expériences a faire)
- entrainement sur 15 et évaluation sur 1 (16 expériences a faire)
- entrainement sur 16 et eval sur 16 tout mélanger avec validation croisée.

Ces façon d'évaluer nous permette de tester plusieurs caractéristique de nos modeles dont la capacité de transfert d'un sujet a l'autre.
Pour chaque test on sauvegardera le f1 score sur la classe minoritaire.

Part II
La géometrie de riemann

1 Distance

On a comme article ça [1] et ça [2].
Pour deux matrice $\mathbf{\Sigma}_1$ et $\mathbf{\Sigma}_2$ leurs distances est d'apres la geometrie de riemann 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'éléctrodes.
F :Norme de Frobenius

2 Point moyen

Pour définir la matrice moyenne nous ne possedons pas d'expression explicite.

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

Pour la calculer on peut utiliser une descente de gradiant.

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 dimmenssions.

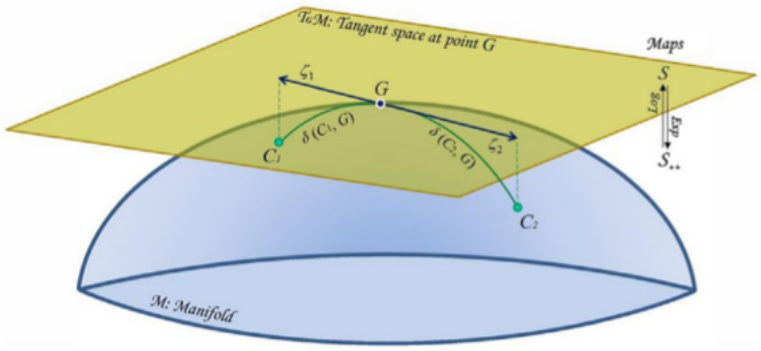


Figure 1: Affichage del'espace tangent

Part III
Prise en main sur un dataset simple

1 Decription du dataset eye close/eye open

Ce dataset contient les données enregister 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 échantilloné 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 ecart type de 5.27 et une médianne 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 éléctrodes.

On commence par visualiser le signal des 16 éléctrodes ainssi que leurs labels associé au cours du temps.

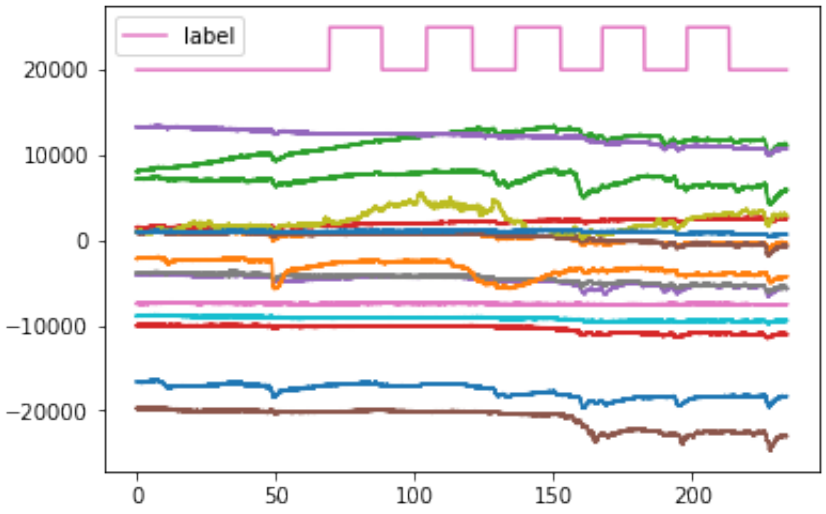
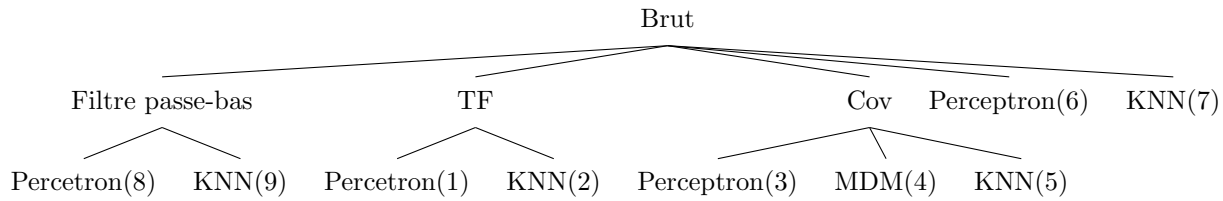


Figure 2: Affichage des données brut

2 Les diférentes méthodes

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



Légende :
Cov : Matrice de covariance
TF : Transformé de fourier
MDM : Minimum Distance to Mean

Prédiction théorique :
La méthode 6 et 7 ne devrais pas fonctioner 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.

3 Résultat des différents algorithmes

3.1 les données brut

$clf = SGDClassifier(loss = "perceptron", eta0 = 1e{-}4, learning_rate = "constant", penalty = None, tol = 1e-1, max_iter = 10000, shuffle = True)$ Cross validation avec 5 parties :
F1 Score : 0.5390625

3.2 KNN sur les données brut

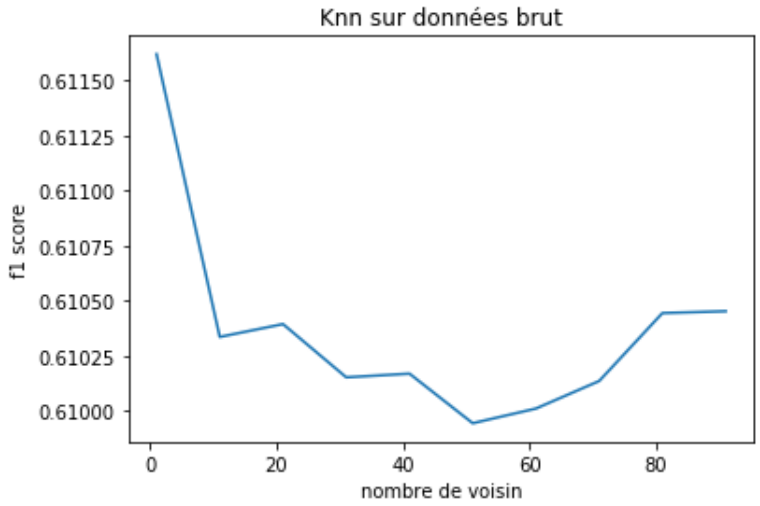


Figure 3: F1 Score(en cross validation) du knn en fonction du pourcentage des données utilisé pour le train

$neigh = KNeighborsClassifier(n_neighbors = 10)$

3.3 Riemann Cov MDM

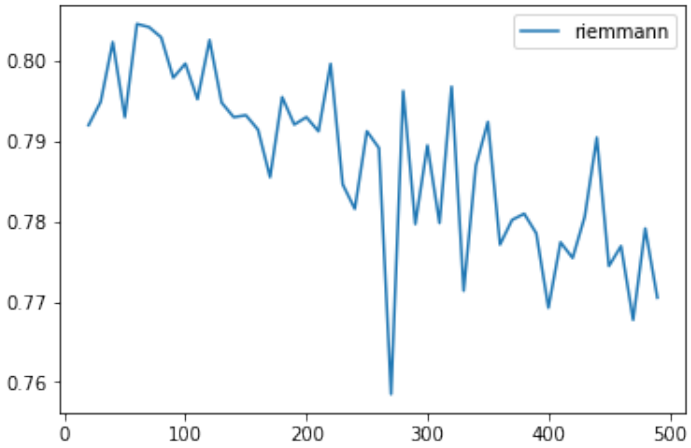


Figure 4: F1 Score(en cross validation) de riemann MDM en fonction du nombre de données par paquet

estimer la matrice de covariance
$cov = pyriemann.estimation.Covariances().fit_transform(X)$
validation croisée
$mdm = pyriemann.classification.MDM()$

3.4 Riemann Cov KNN

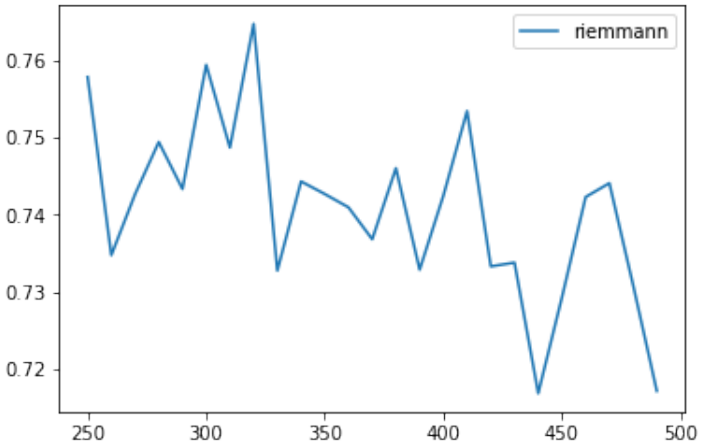


Figure 5: F1 Score(en cross validation) du riemann knn en fonction du nombre de données par paquet

estimer la matrice de covariance
$cov = pyriemann.estimation.Covariances().fit_transform(X)$
validation croisée
$knn = pyriemann.classification.KNearestNeighbor(n_neighbors = 10)$

3.5 Perceptron filtre passe bas

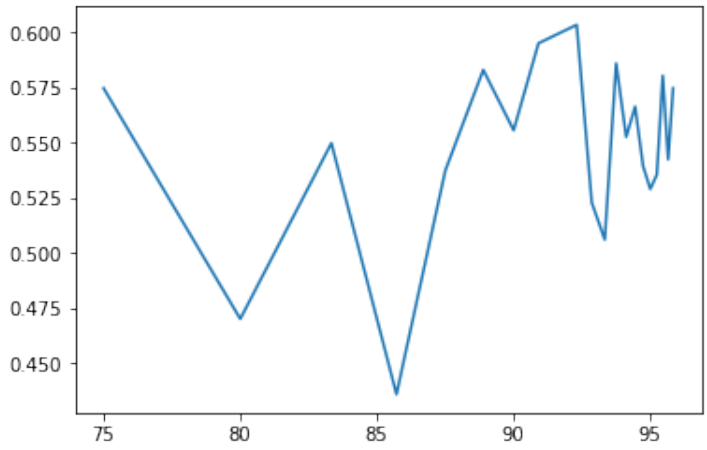


Figure 6: F1 Score(en cross validation) du perceptron en fonction du pourcentage des données utilisé pour le train

modele :
$clf = SGDClassifier(loss = "perceptron", eta0 = 1e{-}4, learning_rate = "constant", penalty = None, tol = 1e-1, max_iter = 10000, shuffle = True)$

3.6 KNN filtre passe bas

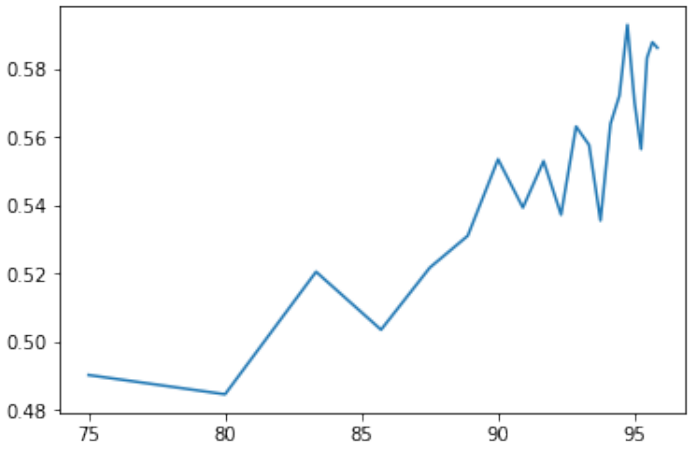


Figure 7: F1 Score(en cross validation) du knn en fonction du pourcentage des données utilisé pour le train

estimer la matrice de covariance
$neigh = KNeighborsClassifier(n_neighbors = 10) y_pred = cross_val_predict(neigh, donnees, labels, cv = k)$

3.7 Perceptron transformée de fourier

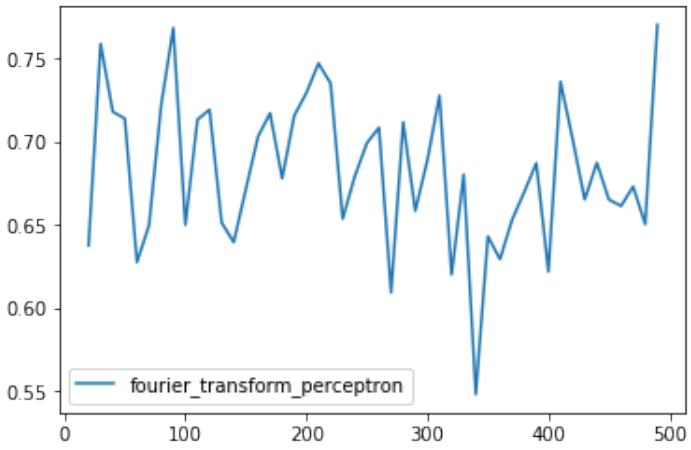


Figure 8: F1 Score(en cross validation) du perceptron en fonction

estimer la matrice de covariance
$cov = pyriemann.estimation.Covariances().fit_transform(X)$
validation croisée
$knn = pyriemann.classification.KNearestNeighbor(n_neighbors = 10)$

3.8 KNN transformée de fourier

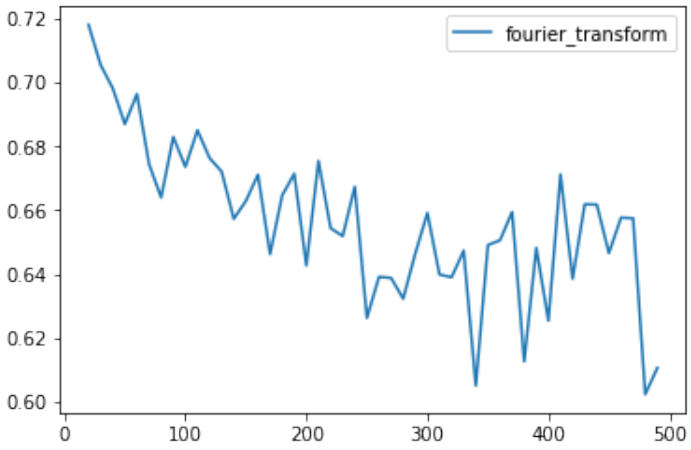


Figure 9: F1 Score(en cross validation) du knn en fonction

estimer la matrice de covariance
$cov = pyriemann.estimation.Covariances().fit_transform(X)$
validation croisée
$knn = pyriemann.classification.KNearestNeighbor(n_neighbors = 10)$

3.9 Tableau récapiltulatif

f1-score arrondie à deux chiffre apres la virgule.

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

Part IV
Brain Invader

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 etait représenté sur un ecran.12 flashs dont 2 comprennent l'alien ciblé. Détécté les flash ou il y a l'alien en regardant l'activité cérébrale.Plusieurs tentative pour détruire l'alienne.
Premiere tache : Classification binaire (le flash nous interesse(il y a l'alien cible dedans) ou pas)
Deuxieme tache : dans le groupe de 12 ou sont les deux flashs avec l'alien cible.

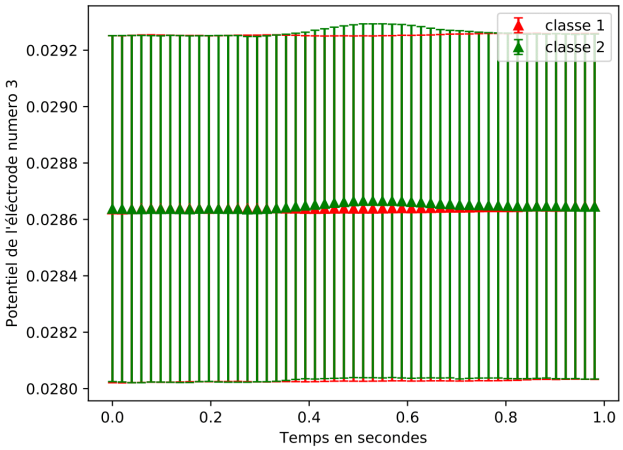


Figure 10: visuel classe mean std

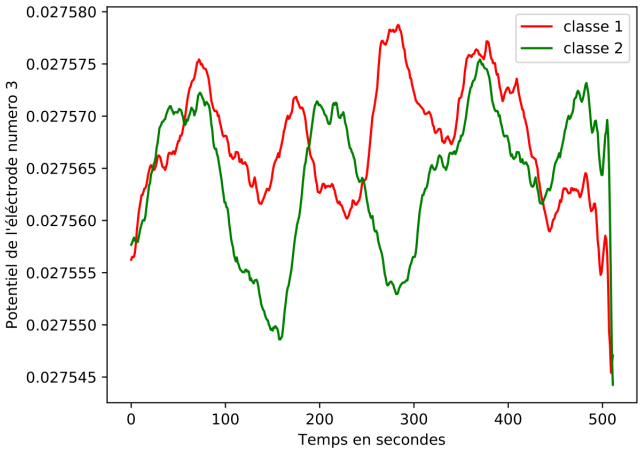


Figure 11: visuel data 0-1

F1-score des différents algorithme sur brain invader :

nom de l'algorithme	1 seconde	0.1 seconde	0.04 seconde
perceptron brut	0.323051948051948	0.326461038961039	0.267532467532467
perceptron tf	0.451136363636364	0.323051948051948	0.408313347178491
riemann MDM	0.441884775795033	0.450794330760887	0.441463342031226
knn brut	0.391033345601445	0.421304978708651	0.477791812169029
cov perceptron	0.293170459768325	0.32987012987013	0.267283968878502
passe bas KNN	0.468170847379269	0.417829605960706	0.467431891494059
knn tf	0.399253802094698	0.470427433051929	0.453723364485908
passe bas perceptron	0.267532467532467	0.326461038961039	0.451136363636364
conv1D	0.384761943288956	0.326461038961039	0.33686397710642

Part V
DecMeg2014

1 Description du dataset

2 Les différentes étapes

1 - On garde uniquement 1 secondes sur les 1.5 secondes de signal car le stimulus intervient qu'apres 0.5 secondes.
2 - On filtre le signal avec un filtre passe bande de Butterworth d'ordre 5 entre 1Hz et 20Hz.

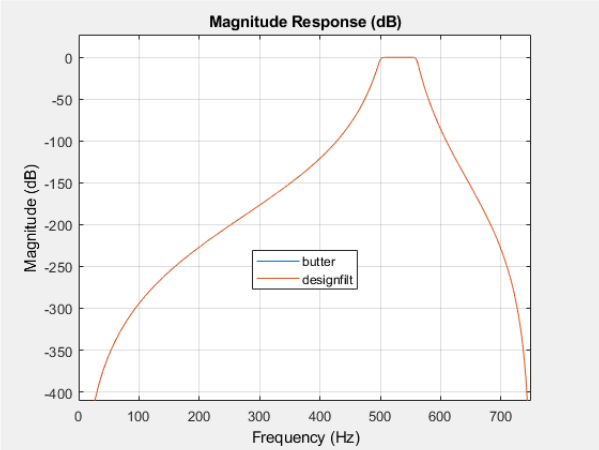


Figure 12: butterworth filter

3 - Par la suite on effectue un filtrage spacial, on extrait 4 channel virtuel par classe.

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

Figure 13: P moyen

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

Figure 14: spatial filtering

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

Figure 15: Zi

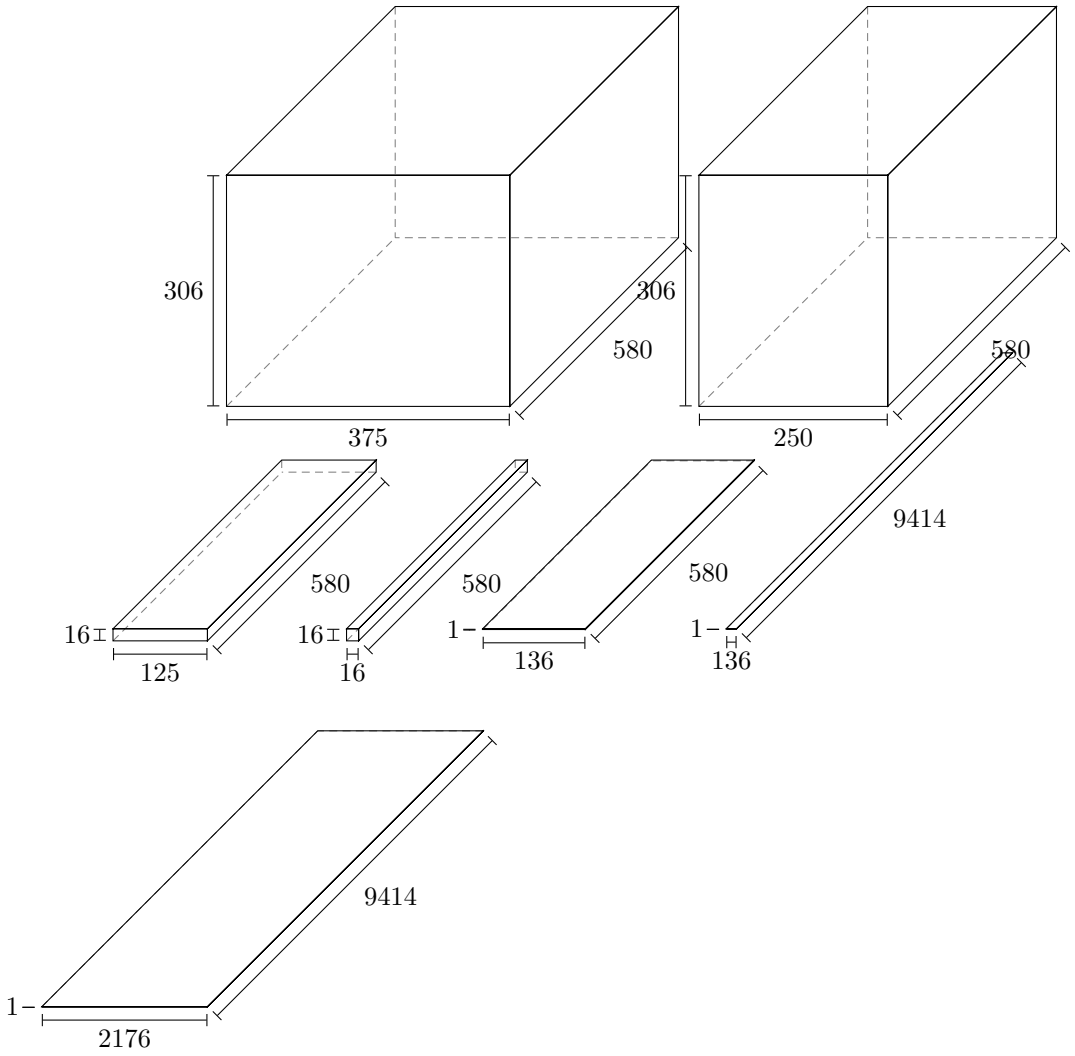
$$\tilde{\mathbf{Z}}_i = \begin{bmatrix} \mathbf{W}^{(0)^T} \mathbf{P}^{(0)} \\ \mathbf{W}^{(1)^T} \mathbf{P}^{(1)} \\ \mathbf{Z}_i \end{bmatrix}.$$

Figure 16: features space

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

Figure 17: covariance

4 - On definit une nouvelle entrée comme etant la concatenation :
- du signal moyen de la classe 1 auquel on multiple par W0 pour les prejeters sur les channels virtuelles appris precedement.
- 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 projete cette matrice sur l'espace tangent.
7 - On estime pour les 15 autre sujets avec ce filtre spatial 8 - On fait la meme chose pour les 15 autres sujet et on concatenne.
9 - Regression 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 VI
Conclusion

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.