

PLDAC

Buton Nicolas

February 15, 2019



Table des matières

I	Intro	1
1	Description du dataset	1
2	Les différentes méthodes	2
II	Résultat des différents algorithmes	2
1	Perceptron sur les données brut	3
2	KNN sur les données brut	4
3	Riemann Cov MDM	5
4	Riemann Cov KNN	6
5	Perceptron filtre passe bas	7
6	KNN filtre passe bas	8
7	Perceptron transformée de fourier	9
8	KNN transformée de fourier	10

Part I

Intro

1 Description du dataset

Fréquence d'échantillonnage : 512Hz

Nombre de participant : 20 (7 femmes, 13 hommes)

Age :

mean: 25.8

sd : 5.27

median 25.5

18 subjects between 19 and 28 years old.

Two participants with age 33 and 44 were outside this range.

Nombre d'électrode : 16

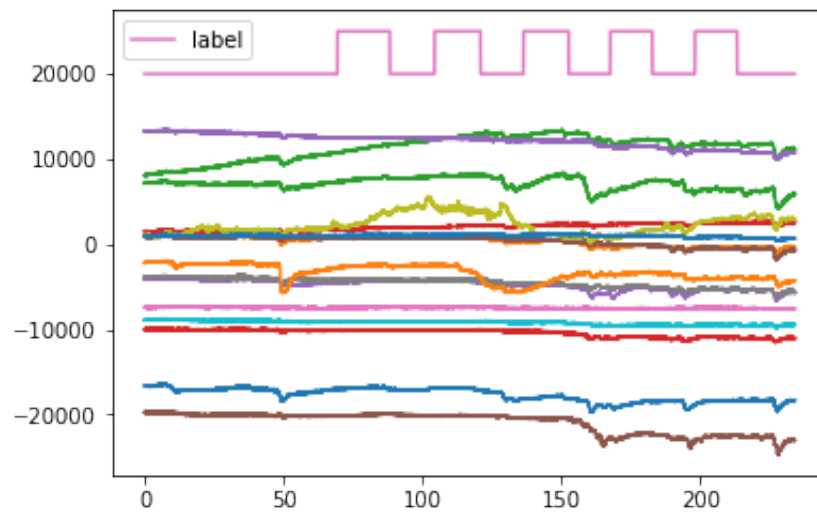
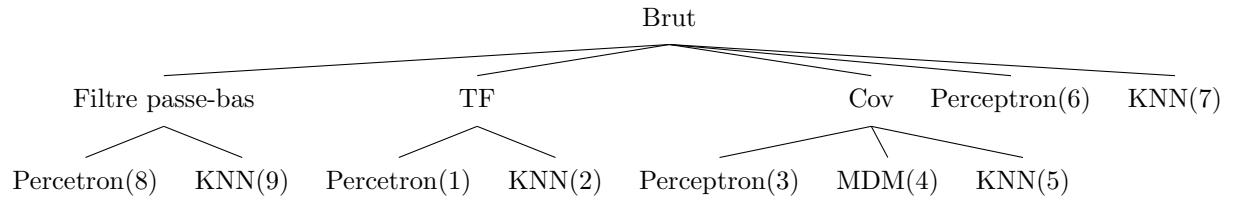


Figure 1: Affichage des données brut

2 Les différentes méthodes



Légende :

Cov : Matrice de covariance

TF : Transformé de fourier

MDM : Minimum Distance to Mean

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 et on ne pourra pas connaitre le debut.

Part II

Résultat des différents algorithmes

1 Perceptron sur les données brut

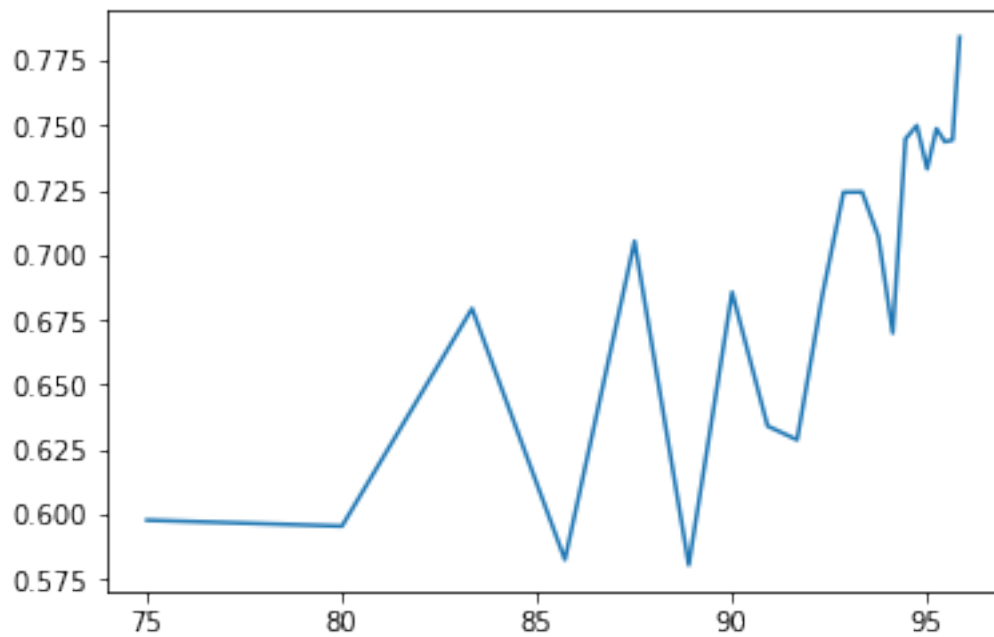


Figure 2: F1 score(en cross validation) du perceptron en fonction du pourcentage des données utilisé pour le train

```
clf = SGDClassifier(loss = "perceptron", eta0 = 1e - 4, learning_rate =  
"constant", penalty = None, tol = 1e - 1, max_iter = 10000, shuffle = True)
```

2 KNN sur les données brut

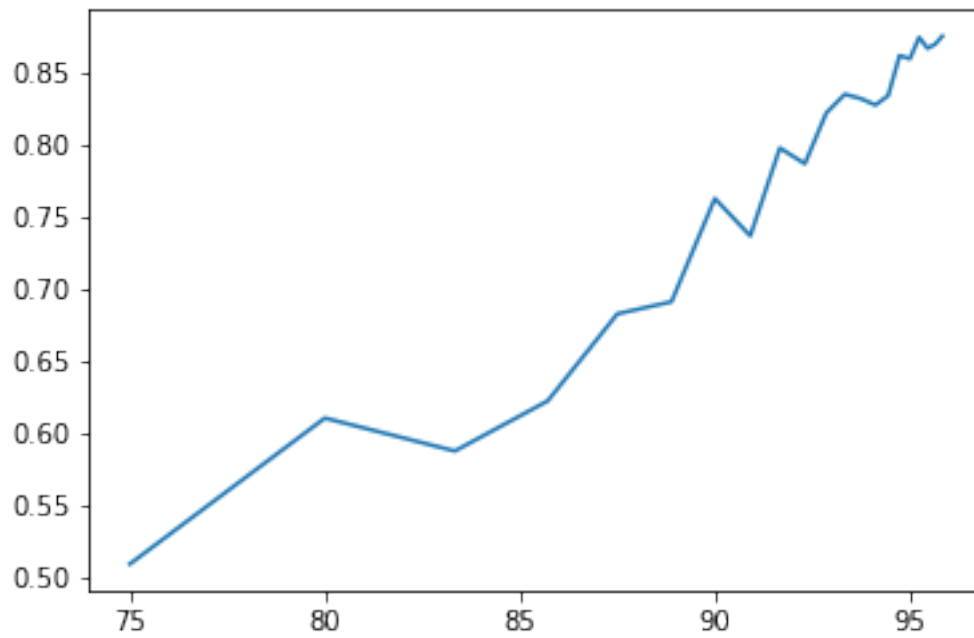


Figure 3: F1 Score(en cross validation) du knn en fonction du pourcentage des données utilisé pour le train

neigh = KNeighborsClassifier($n_{neighbors} = 10$)

3 Riemann Cov MDM

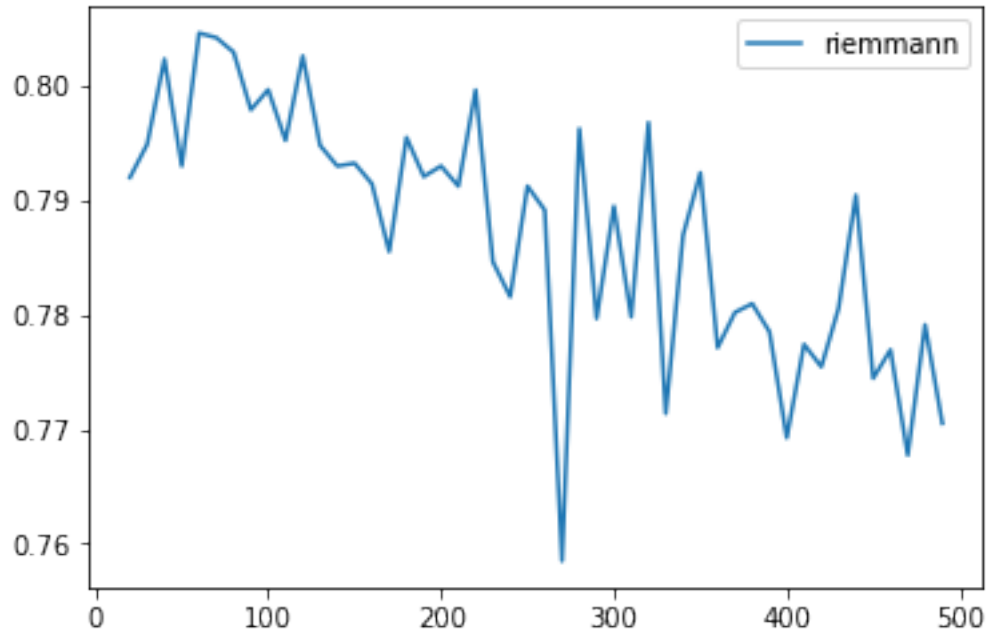


Figure 4: F1 Score(en cross validation) du knn en fonction du pourcentage des données utilisé pour le train

estimer la matrice de covariance

```
cov = pyriemann.estimation.Covariances().fit_transform(X)
```

validation croisée

```
mdm = pyriemann.classification.MDM()
```

4 Riemann Cov KNN

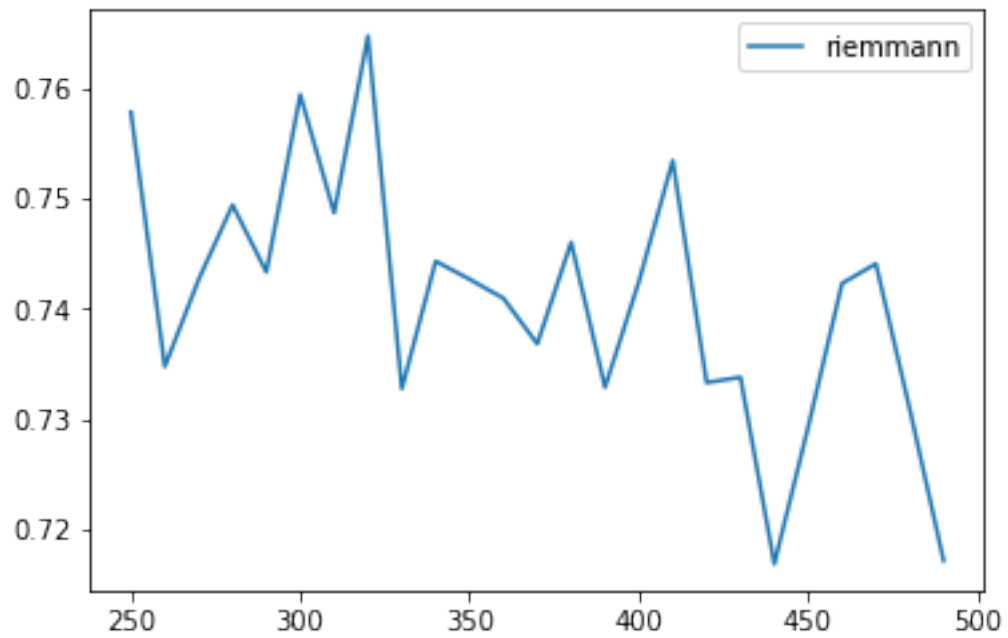


Figure 5: F1 Score(en cross validation) du knn en fonction du pourcentage des données utilisé pour le train

estimer la matrice de covariance

```
cov = pyriemann.estimation.Covariances().fit_transform(X)
```

validation croisée

```
knn = pyriemann.classification.KNearestNeighbor(n_neighbors = 10)
```


5 Perceptron filtre passe bas

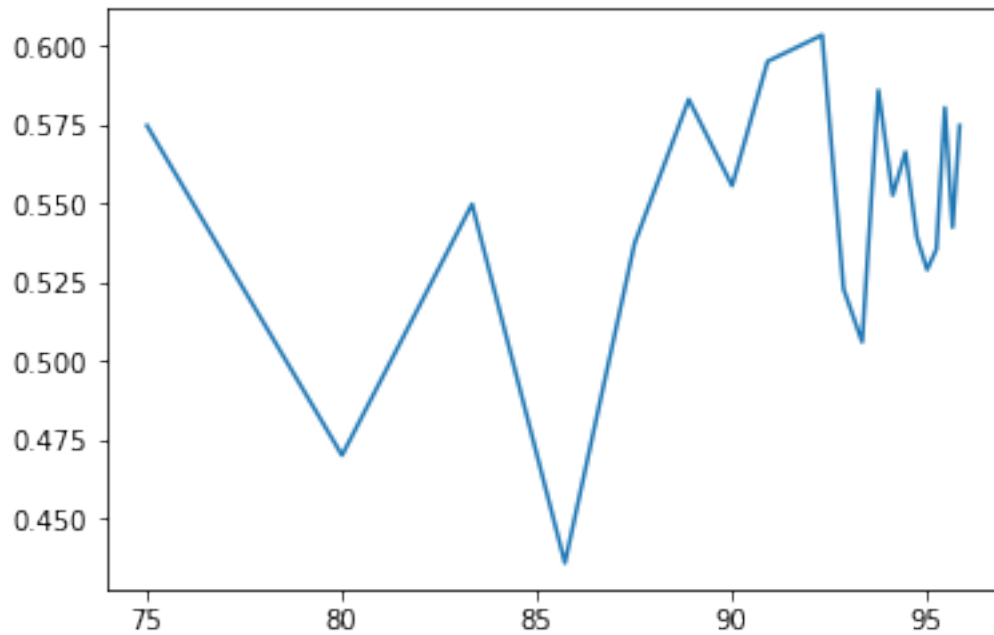


Figure 6: F1 Score(en cross validation) du knn en fonction du pourcentage des données utilisé pour le train

estimer la matrice de covariance

```
cov = pyriemann.estimation.Covariances().fit_transform(X)
```

validation croisée

```
knn = pyriemann.classification.KNearestNeighbor(n_neighbors = 10)
```

6 KNN filtre passe bas

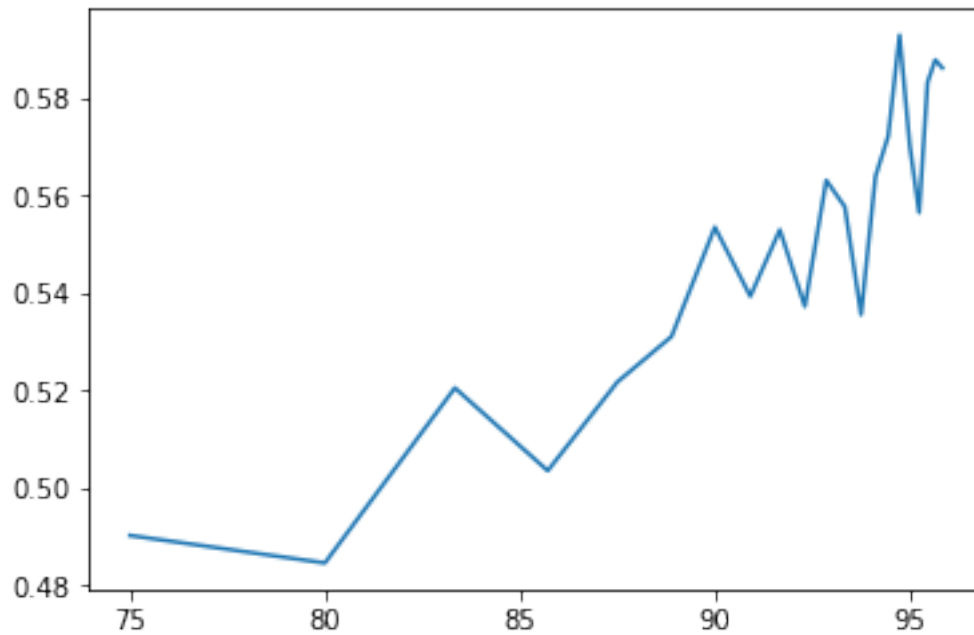


Figure 7: F1 Score(en cross validation) du knn en fonction du pourcentage des données utilisé pour le train

estimer la matrice de covariance

```
cov = pyriemann.estimation.Covariances().fit_transform(X)
```

validation croisée

```
knn = pyriemann.classification.KNearestNeighbor(n_neighbors = 10)
```

7 Perceptron transformée de fourier

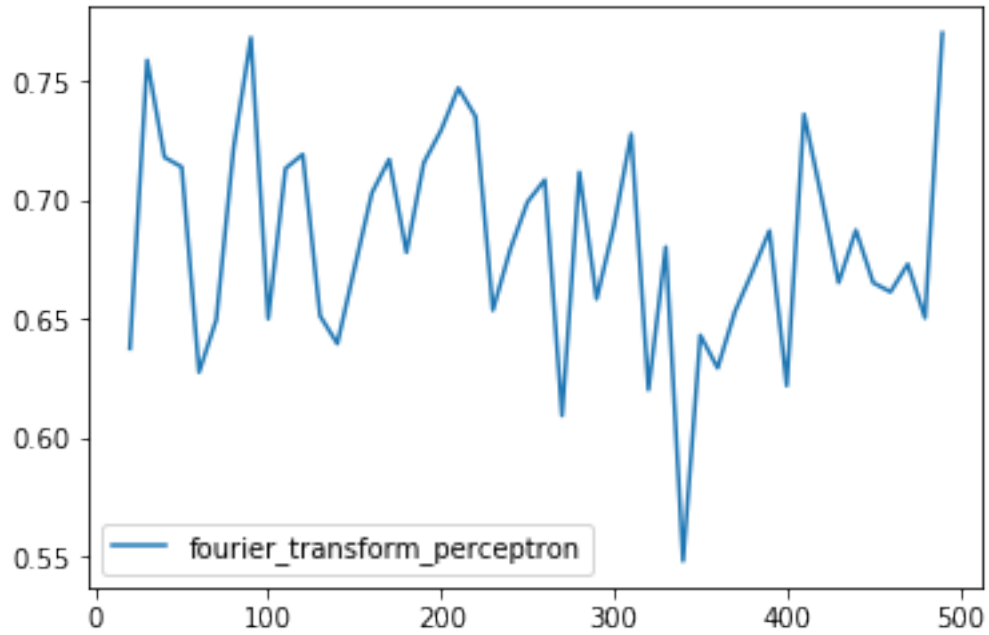


Figure 8: F1 Score(en cross validation) du perceptron en fonction

estimer la matrice de covariance

```
cov = pyriemann.estimation.Covariances(fit_transform(X))
```

validation croisée

```
knn = pyriemann.classification.KNearestNeighbor(n_neighbors = 10)
```

8 KNN transformée de fourier

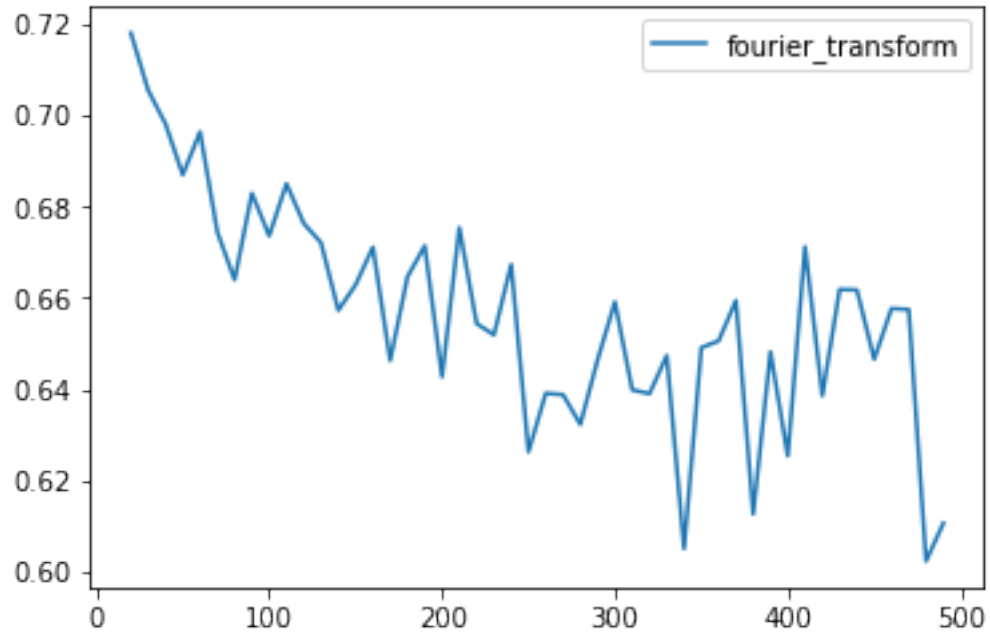


Figure 9: F1 Score(en cross validation) du knn en fonction

estimer la matrice de covariance

```
cov = pyriemann.estimation.Covariances().fit_transform(X)
```

validation croisée

```
knn = pyriemann.classification.KNearestNeighbor(n_neighbors = 10)
```