**Compte-rendu de TP**

**D’Apprentissage**

Par Loan BERNAT



TP1 : Etude des voyelles

Une image contenant table

Description générée automatiquement

***Partie n°1 : Analyse des signaux et prétraitement des paramètres***

Vous pouvez retrouver les différentes fonctions réalisées dans cette partie dans le fichier de code main.py.

J’ai choisit personnellement de récupérer 10 mfcc par signal de [1 à 11]. Et de répartir initialement mes bases de données et de test de manière à avoir 910 signaux d’apprentissage et 90 de tests. C’est avec ces coefficients sceptraux que j’ai obtenu les meilleurs résultats même sans prétraitement.

Une image contenant texte

Description générée automatiquement

Une image contenant texte

Description générée automatiquement

Une image contenant texte

Description générée automatiquement

Une image contenant texte

Description générée automatiquement

Pour les options de mes modèles, j’ai essayé de faciliter les performances en temps, en mettant des nombres max de répétitions, une profondeur max pour le RandomForest. Ce sont des paramètres adaptés dans notre cas mais qui pourrait être intéressant de modifier voir de retirer si les performances deviennent insuffisante.

***Partie n°2 : Evaluation des performances et recherche des performances optimales***

On sépare cette partie en 2 phases. La première se pose dans les conditions initiales du sujet et on projette via les 10 premiers vecteurs propres pour l’AFD et l’ACP. Cependant, j’obtient ici des performances « trop » excellente, je vais donc essayer dans partie 2 de réduire le nombre de projection à 4. En prenant [2 : 6].

|  |
| --- |
| **Remarques :**  Pour KMeans, l’accuracy score n’est pas intéressant à étudier. On se basera uniquement sur la matrice de confusion pour ce modèle. |

**PHASE DE TEST SANS PRETRAITEMENT :**

* RandomForestClassifier

Le modèle fait une erreur et classe un élément de la 8ème classe dans la 9ème

Une image contenant texte

Description générée automatiquement

* MLPClassifier

Le modèle fonctionne parfaitement.

Une image contenant flèche

Description générée automatiquement

* Kmeans

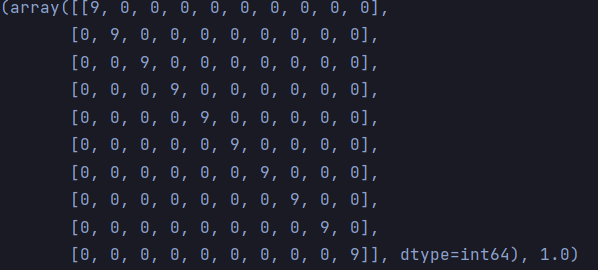
*On remarque que le modèle se trompe sur la 9ème ligne, attribuant à 2 signaux la mauvaise classe. On a donc 2 erreurs sur 90.*

Une image contenant texte, clavier, équipement électronique

Description générée automatiquement

* SVC

Le modèle fonctionne parfaitement



**PHASE DE TEST 1 :**

**AFD :**

* RandomForestClassifier

Le modèle fonctionne parfaitement

Une image contenant flèche

Description générée automatiquement

* MLPClassifier

Le modèle fonctionne parfaitement.

Une image contenant flèche

Description générée automatiquement

* KMeans

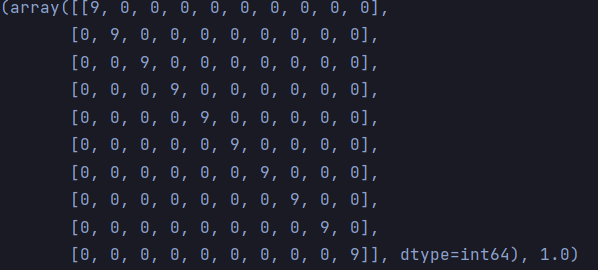
Le modèle classe parfaitement nos données.

Une image contenant texte, clavier

Description générée automatiquement

* SVC

Le modèle fonctionne parfaitement



**ACP :**

* RandomForestClassifier

Le modèle fait une erreur et classe un élément de la 8ème classe dans la 9ème

Une image contenant texte

Description générée automatiquement

* MLPClassifier

Le modèle fonctionne parfaitement.

Une image contenant flèche

Description générée automatiquement

* KMeans

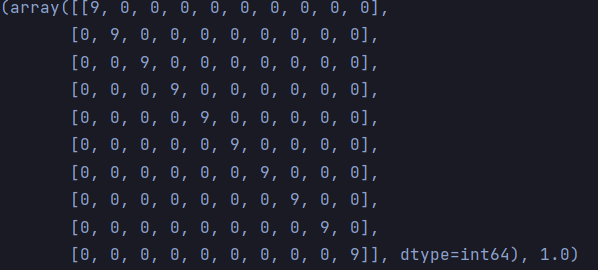
Le modèle fait une erreur à la 9ème ligne. Le résultat est tout de même meilleur que sans traitement, mais reste moins performant que avec AFD.

Une image contenant flèche

Description générée automatiquement

* SVC

Le modèle fonctionne parfaitement



|  |
| --- |
| **Remarques :**  Le prétraitement par AFD amène tous les modèles à la perfection mais seulement SVC et MLPCLassifier sont parfait dans tous les cas. |

**PHASE DE TEST 2**

AFD :

|  |  |
| --- | --- |
| RandomForestClassifier | MLPClassifier |
|  |  |
| KMeans | SVC |
|  | Une image contenant texte, équipement électronique, clavier  Description générée automatiquement |

On voit que l’AFD reste « performante » dans une certaine mesure ici. KMeans ne fonctionne plus très bien, on voit un grand nombre d’erreur dans sa matrice de confusion tandis que les autres modèles conserve un score autour de 90% de réussite.

ACP :

|  |  |
| --- | --- |
| RandomForestClassifier | MLPClassifier |
|  |  |
| KMeans | SVC |
|  | Une image contenant texte, équipement électronique, clavier  Description générée automatiquement |

On voit ici que l’ACP ne fonctionne plus. Elle a besoin des paramètres que l’on a enlevé pour fonctionner correctement et obtenir des performances intéressante.

|  |
| --- |
| **CONCLUSION :**  Après de nombreux tests effectués, que je ne détaillerai pas dans ce compte rendu, il semble idéal d’utiliser un **modèle MLPClassifier ou SVC** avec un **prétraitement par AFD** avec une **projection sur les 6 vecteurs** (dans mon cas, les premiers, car les 4 autres sont trop minimes). On allie donc performance en temps d’exécution et résultat optimal en toutes circonstances.  Même si dans notre cas, les prétraitements n’ont pas présenté un grand intérêt d’amélioration. Dans d’autres contextes, avec sans doutes, des bases de tests plus importantes et donc plus  « variés », procéder à une AFD nous permettra de palier à ça pour maintenir des performances élevées.  On discerne également qu’il peut arriver que les modèles fassent des erreurs lors de la prédiction entre les classes 7 et 8 qui sont les deux type de « o ». |

*Code main :*

Une image contenant texte

Description générée automatiquement