Chapitre 6

Manipulations du Master 1 - SIA

6.1 Présentation des manipulations

6.1.1 Introduction

Les travaux pratiques de classification sont répartis sur deux séances de trois heures. Leur but est d'illustrer toutes les étapes rencontrées lors de la résolution d'un problème de classification. La mise en œuvre est prévue sous Python 3. Le séquencement sera le suivant :

TP no1 :
- Etude temporelle et fréquentielle des signaux.
- L'extraction de paramètres pertinents pour la classification envisagée.
- Prétraitement des paramètres en vue de la classification.

TP no2 :
- La mise en œuvre de méthodes de classification :
  * Algorithme des centres mobiles.
  * Perceptron multicouches.
  * Séparateurs à vastes marges - SVM.
  * Forêt d'arbres décisionnels - Random Forest
- L'évaluation de la qualité de la classification obtenue.

Avant d'arriver en séance, vous devez impérativement :
- Avoir assimilé les notions présentées en cours et en TD.
- Avoir préparé les programmes demandés pour consacrer l'essentiel de la séance à l'obtention et l'interprétation des résultats.

Les fichiers nécessaires sont accessibles à l'adresse :
`http://userpages.irap.omp.eu/~jtrouilhet/TP_Classification/M1SIA/`

6.2 Séance n^o1 : Analyse des signaux et prétraitement des paramètres

6.2.1 Présentation de la manipulation

Pour la manipulation, vous aurez à votre disposition un ordinateur ave le langage de programmation Python qui vous permettra d'effectuer les traitements envisagés. Les signaux sont générés par le programme `genere.py` et stockés dans le fichier `signaux.txt`. Le programme de lecture est fourni dans le fichier `TPClassif2022.py`.

Détail des données fournies :

- Nom du fichier contenant les données : `signaux.txt`
- Nom de la variable contenant les individus : `MatriceDonnees`
- Dimensions de `MatriceDonnees` : NbrIndividus x NbrParametres
- Nom de la variable contenant le numero des classes : `NoClasse`
- Dimensions de `NoClasse` : NbrIndividus x 1

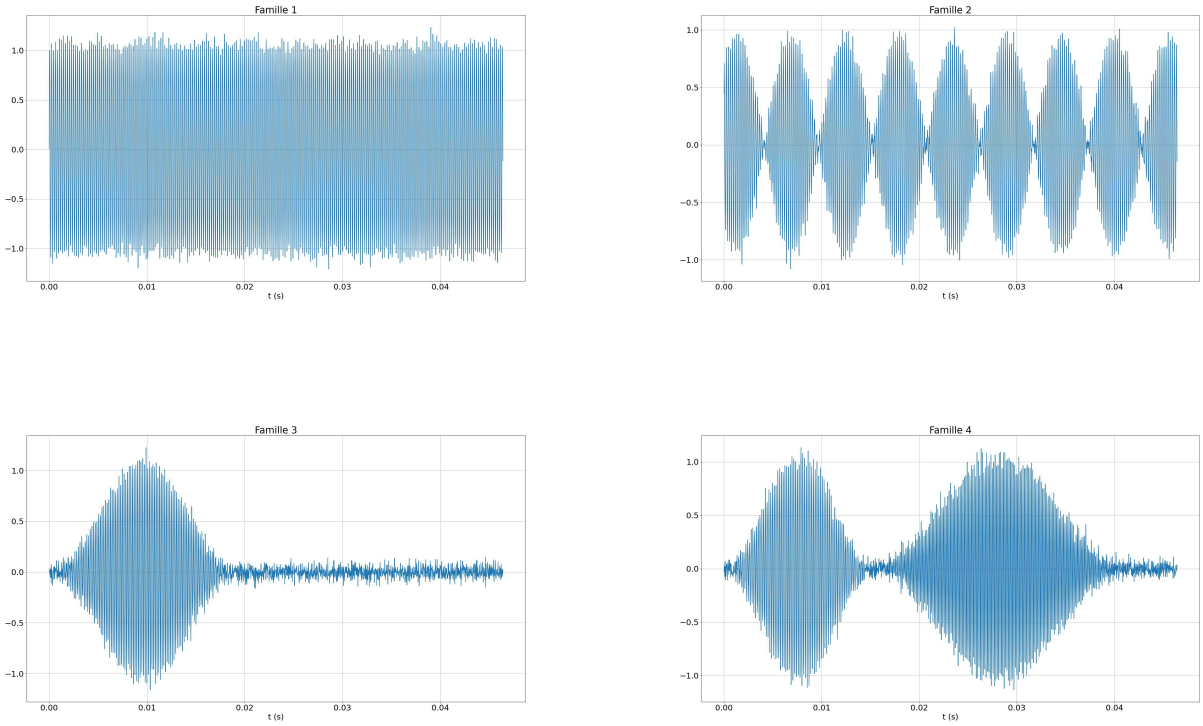

FIGURE 6.1 – Différents types de signaux à reconnaître

Les signaux à reconnaître se répartissent en quatre familles qui sont présentées à la figure n° 6.1. Le fichier `signaux.txt` contient les enregistrements de chaque famille, La fréquence d'échantillonnage utilisée est 44100 Hz.

6.2.2 Travail à effectuer

Les titres suivis d'une astérisque doivent être préparés pour la séance de TP. Compte tenu des procédures disponibles et détaillées plus loin, la programmation reste aisée car les programmes sont courts.

Analyse du signal* *partiellement*

L'objet de cette partie est de mettre en évidence des singularités permettant de discerner les familles de signaux entre elles. Pour cela, une étude temporelle puis fréquentielle des signaux sera

effectuée. L'examen temporel des signaux montre que l'enveloppe est porteuse d'information. Nous savons que le module de la transformée de Hibert d'un signal à bande étroite correspond à l'enveloppe de celui-ci. Aussi, pour le TP, vous pourrez utiliser les commandes suivantes :

```
import numpy as np
from scipy.signal import hilbert
Enveloppe = np.abs(hilbert(MatriceDonnees))
```

Extraction de paramètres pertinents

En fonction des résultats obtenus précédemment, écrire un programme Python qui permette d'extraire, au moins, trois paramètres représentatifs des signaux de façon automatique.

Prétraitement des données*

Afin d'améliorer les performances lors de la classification, vous réaliserez une Analyse Factorielle Discriminante AFD sur les données tout en réduisant l'espace de représentation à deux paramètres. Afin de pouvoir comparer les deux approches, vous réaliserez aussi une Analyse en Composantes Principales dans les mêmes conditions. Vous prendrez soin de justifier l'intérêt d'utiliser ou non des variables centrées-réduites.

6.2.3 Conclusions

Enumérez les points délicats que vous avez rencontrés tout au long de la manipulation. En regard des résultats obtenus par les deux méthodes (AFD/ACP) indiquez les avantages et les inconvénients de chacune.

6.3 Séance n^o2 : Mise en œuvre de méthodes de classification.

6.3.1 Introduction

Le but de cette séance est de mettre en évidence les performances des différentes méthodes de classification envisagées. Pour les trois sortes de classifieurs, vous évaluerez les performances de chacune des méthodes sur les données issues du prétraitement par Analyse Factorielle Discriminante et par Analyse en Composantes Principales.
Afin de tester les capacités de généralisation de chacune des méthodes, c'est à dire éviter l'apprentissage par coeur, vous effectuerez l'apprentissage sur un jeu de données créé par une première exécution de `genere.py` (Base d'apprentissage) et le test des performances sur un autre jeu de données obtenu par une seconde exécution de `genere.py` (Base de Test).

6.3.2 Travail à effectuer

Le but de cette partie est de mettre en œuvre différentes méthodes de classification disponibles dans le module **sklearn** :

— Algorithme des centres mobiles :

```
Classif=sklearn.cluster.KMeans( ... Options ... )
Classif.fit(X_Apprentissage, NoClasses)
y_predict=Classif.predict(X_Test)
```

— Perceptron multicouche :

```
Classif = sklearn.neural_network.MLPClassifier( ... Options ... )
Classif.fit(X_Apprentissage, NoClasses)
y_predict=Classif.predict(X_Test)
```

— Séparateurs à Vastes Marges :

```
Classif = sklearn.svm.SVC( ... options ... )
Classif.fit(X_Apprentissage, NoClasses)
y_predict=Classif.predict(X_Test)
```

— Forêt d'arbres décisionnels :

```
Classif = sklearn.ensemble.RandomForestClassifier( ... options ... )
Classif.fit(X_Apprentissage, NoClasses)
y_predict=Classif.predict(X_Test)
```

6.3.3 Conclusions

Enumérez les points délicats que vous avez rencontrés tout au long de la manipulation. En regard des résultats obtenus pour chaque des méthodes, indiquez les avantages et les inconvénients de chacune.

6.4 Présentation des procédures Python mises à disposition

Afin de faciliter votre travail, les procédures de base sont à votre disposition.

6.4.1 Génération des fichiers contenant les enregistrements

Le programme `genere.py` crée le fichier `signaux.txt` qui contient, les paramètres de chaque individu ainsi que la classe à laquelle il appartient.

```
#-------------------------------------------------------------------------
# Master 1 SIA                 Génération des signaux
#-------------------------------------------------------------------------
# Entree :  Néant
# Sortie :  Le fichier contenant les individus :  signaux.txt
#-------------------------------------------------------------------------
```

6.4.2 Fichier à compléter

Le fichier `TPClassif2022.py` contient la procédure de lecture du fichier `signaux.txt`, il ne vous restera qu'à compléter celui-ci avec votre programme.

A l'issue de la lecture du fichier `signaux.txt`, les variables suivantes seront disponibles :

— Nom de la variable contenant les individus : `MatriceDonnees`
de dimensions : NbrIndividus x NbrParametres
— Nom de la variable contenant le numero des classes : `NoClasse`
de dimensions : NbrIndividus x 1

6.4.3 Procédures de calcul pour l'ACP et l'AFD

Les procédures pour la mise en œuvre de l'AFD et l'ACP sont contenues dans le module du fichier `TPClassif.py`. Vous ferez appel à lui en ajoutant la ligne : `import TPClassif`.
Voici, ci-après, le détail de chacune d'elles.

Normalisation des variables

Cette fonction permet d'obtenir des variables centrées et réduites.

```
def CalculerIndividusCentresReduits(Individus, CentresGravite) :
#-------------------------------------------------------------------------
#
# Calcul des individus centrés réduits
#
#-------------------------------------------------------------------------
# Entree : Individus                [NbrIndividus x NbrParametres]
#          CentresGravite           [NbrClasses   x NbrParametres]
#
# Sortie : IndividusCentresReduits  [NbrIndividus x NbrParametres]
#-------------------------------------------------------------------------
```

Calcul des centres de gravité

Cette fonction permet de calculer les centres de gravité des classes telles quelles sont définies par la variable `classes`.

```
def CalculerCentresGravite(Individus, NoClasses) :
#-------------------------------------------------------------------------
#
# Calcul des centres de gravité de chaque classe.
#
#-------------------------------------------------------------------------
# Entree : Individus               [NbrIndividus x NbrParametres]
#          NoClasses               [NbrIndividus]
#
# Sortie : CentresGravite          [NbrClasses   x NbrParametres]
#          Nb : CentreGravite[0] =  Centre gravite global
#-------------------------------------------------------------------------
```

Calculer les variances

Cette fonction permet de calculer les valeurs des variances telles quelles ont été définies en cours.

```
def CalculerVariances(Individus, NoClasses, CentresGravite) :
#-------------------------------------------------------------------------
#
# Calcul des variances Totale, Intraclasses et Interclasses
#
#-------------------------------------------------------------------------
# Entree : Individus          [NbrIndividus x NbrParametres]
#          NoClasses          [NbrIndividus x 1 ]
#          CentresGravite     [NbrClasses   x NbrParametres]
#
```

```
# Sortie : VT = Variance totale       [1 x 1]
#          VA = Variance intraclasses [1 x 1]
#          VE = Variance interclasses [1 x 1]
#--------------------------------------------------------------------------
```

Calcul des matrices de covariance

Cette fonction permet de calculer les matrices de covariances telles qu'elles ont été définies en cours.

```
def CalculerMatricesCovariance(Individus, NoClasses, CentresGravite):
#--------------------------------------------------------------------------
#
# Calcul des matrices de covariance Totale, Intraclasses et Interclasses
#
#--------------------------------------------------------------------------
# Entree : Individus          [NbrIndividus x NbrParametres]
#          NoClasses          [NbrIndividus x 1 ]
#          CentresGravite     [NbrClasses   x NbrParametres]
#
# Sortie : CT = Matrice de covariance totale       [NbrParametres x NbrParametres]
#          CA = Matrice de covariance intraclasses [NbrParametres x NbrParametres]
#          CE = Matrice de covariance interclasses [NbrParametres x NbrParametres]
#--------------------------------------------------------------------------
```

6.4.4 Module scikit-learn - sklearn

Le module sklearn est incontournable pour l'apprentissage automatique sous Python. Cette librairie couvre la majeure partie de la discipline. Entre autres, les algorithmes nécessaires pour le déroulement du TP :

* Algorithme des centres mobiles - K-Means clustering.
* Perceptron multicouches - Multi-layer Perceptron classifier.
* Séparateurs à vastes marges - Support Vector Classifier.
* Forêt d'arbres décisionnels - Random Forest classifier.

Vous vous refèrerez aux nombreux tutoriaux qu'offre le site `scikit-learn.org`