Analyse de données audionumériques
Fiche de Travaux Pratiques numéro 1
Reconnaissance du chanteur

Mathieu Lagrange & Grégoire Lafay

25 février 2019

Ce document, les scripts Matlab ainsi que les fichiers audionumériques sont disponibles ici : `http://pagesperso.ls2n.fr/~lagrange-m/teaching/sigma/tp/sigmaTP1mir.zip`

Par convention, les fonctions directement disponibles via Matlab sont en italique. Une liste des commandes Matlab utiles pour ce TP est données en appendice.

Le but de ce TP est d'étudier des méthodes simples permettant de classifier des morceaux de usique en fonction de l'indentité du chanteur. Les enregistrements considérés ici sont des morceaux de musiques composés de deux types de sources : "voix" et "musique".

Dans cette première séance, nous allons étudier différentes techniques de descriptions de fichiers sonores. Nous mesurerons la "pertinence" de ces descripteurs à l'aide de méthodes d'évaluation simples. Par "pertinence", nous comprenons des descriptions qui permettent de bien discriminer les sources sonores que l'on cherche à isoler.

Nous commencerons par calculer des descripteurs spectraux pour chacun des morceaux de musiques. Enfin nous évaluerons la "pertinence" des différents descripteurs utilisés.

1 Description des données audio

On dispose d'un corpus de 40 chansons chantées par 10 chanteurs différents (4 par chanteurs). A chaque chanteur est attribué un label allant de a à j. Chacune de ces chansons a été segmentée en deux parties *music* et *voice*. La première correspond aux 15 premières secondes de la chanson et la seconde à 15 secondes après la première minute. Au total nous avons donc 80 extraits musicaux (x_i, $i \in [1:80]$), 40 avec une partie chantée, 40 sans partie chantée, répartie sur 10 classes labellisées de a à j en fonction du chanteur.

Notez que, d'après le style musical, l'accompagnement (*music*) a de fortes chances d'être présent dans les deux extraits d'un même morceaux, tandis que le chanteur a de fortes chances de n'être présent que dans le second extrait. Cette segmentation est arbitraire, mais fonctionne bien dans le cas de morceaux populaires.

1.1 Calcul des descripteurs

Implémenter la fonction **[V,Label2,Label10]=computefeatures(dataPath)** qui fournit en sortie :

- la matrice V comprenant les 80 vecteurs caractéristiques v_i à n dimensions de nos morceaux de musique (ligne : morceaux de musique ; colonne : coefficient)
- le vecteur label2 indiquant pour chaque morceau si il s'agit d'une partie chantée (label2(i)=1) ou d'une partie musicale (label2(i)=2)

— le vecteur label10 indiquant pour chaque morceau si il s'agit du chanteur a (label10(i)=1), du chanteur b (label10(i)=2), et ainsi de suite jusqu'au chanteur j (label10(i)=10)

Nous commencerons par représenter nos données en utilisant le spectre de puissance. Pour ce faire implémenter la fonction X_i = **pSpec(x_i, nfft, nwin, hopsize)** qui calcule le spectrogramme de puissance X_i (freq×time) du signal x_i en réalisant une transformé de Fourier à court terme à l'aide d'une fenêtre de hanning.

Paramètres d'entrée de **pSpec** :

— dataPath : chemin du répertoire contenant les fichiers sonores (cf fonction *dir*)
— nfft : Nombre de points de la FFT
— nwin : taille de la fenêtre d'analyse. (zero-padding si nwin < nfft, nfft doit être une puissance de 2)
— hopsize : pas d'avancement de la fenêtre d'analyse (typiquement nwin/4 ou nwin/2)

Les vecteurs caractéristiques v_i seront obtenus en réalisant la moyenne temporelle des représentations X_i correspondantes.

1.2 Visualisation

Afin de visualiser les positions relatives de nos points décrits par les vecteurs v_i, nous allons devoir réduire la dimension de notre espace de représentation. Pour ce faire nous utiliserons l'analyse en composante principale.

Implémenter la fonction **dataVisu(V,label)** qui permet d'afficher sur un espace à 2 dimensions les points représentant nos morceaux de musique. Cette fonction devra :

1. utiliser la fonction *princomp* (cf. appendice) afin d'extraire les 2 axes expliquant le mieux la variance des données
2. utiliser ces axes pour représenter nos morceau de musique sur un espace en deux dimensions
3. utiliser des points de couleurs différentes en fonction des labels des morceaux

Commenter la visualisation.

1.3 Évaluation du système de description

Pour évaluer notre système de description, nous allons utiliser la métrique dites de precision au rang K. Il s'agit de calculer, pour chaque morceau de musique représenté par le vecteur v_i, le nombre de K plus proches morceaux v_j ayant le même label que que v_i. La précision au rang K est calculée en comptant pour l'ensemble des vecteur v_i, le nombre de K plus proches morceaux v_j ayant le label de v_i et en normalisant par l'ensemble des données testées (K*80).

Implémentez la fonction **[score]=precAtK(K,V,Gt)** qui calcule la précision au rang K pour la matrice V, en fonction de la vérité terrain Gt (label2 ou label10). Utiliser la fonction **myDist(V)** pour calculer les distances euclidiennes entre vos vecteurs v_i.

Calculer les précisions pour différentes valeurs de K en considérant comme vérité terrain les chanteurs (label10) ou le type d'extrait (label2).

Commenter les résultats pour les différentes valeurs de K. Quelle serait la borne inférieure d'une telle méthode d'évaluation (baseline) ?

1.4 Recherche par similarité

Nous allons maintenant utiliser notre système de représentation afin de détecter automatiquement si un morceaux comporte de la voix.

Implémenter la fonction [label]=detectVoice(x,K,V,label2) qui classifie un signal d'entrée x en utilisant la méthode des K plus proches voisins, en fonction de la matrice V contenant nos vecteurs de descriptions v_i et du vecteur label2 contenant les labels (1=voix, 2=musique) des vecteurs v_i.

Tester cette fonction sur les fichiers *bent_music* et *bent_voice*. Vous pouvez aussi la tester sur n'importe quel morceau à votre disposition. Commentez les résultats.

Attention, les éléments de la base sont, pour des raisons de taille de la base de donnée, échantillonés a 5 kHz. Si vos fichiers tests sont à une fréquence d'échantillonage différente, veillez à le prendre en compte.

2 Comparaison de plusieurs descripteurs

Refaire les parties 1.1 et 1.3 en modifiant la fonction **computefeatures** afin de faire varier les paramètres suivants :

1. descripteurs :
   (a) projection sur bandes Mel (Mel_i = **audspec(**X_i**, fe, 40, 'mel', 0, 4000, 1, 1)**)
   (b) logarithme du spectrogramme et logarithme de la représentation en bandes Mel
   (c) MFCC (fonction [$MFCC_i$]=spec2cep(Mel_i, 13, 2))
2. vecteurs caractéristiques :
   (a) représentation de l'évolution temporelle : utiliser ou non les dérivées des représentations X_i (fonction [d]=deltas(X_i)).
   (b) abstraction : concaténer à chaque vecteur moyen v_i les variances de X_i.
   (c) normalisation : normaliser ou non les coefficients des vecteurs v_i avant de les comparer (obligatoire si vous utilisez les variances ou les dérivées).
3. distance : modifier la fonction **mydist** afin de calculer non plus des distances euclidiennes mais cosinusoidales. Quel est l'avantage de cette distance ? La distance cosinusoidale entre deux vecteurs v_i et v_j est calculée par :

$$dist = 1 - \frac{\sum_{m=1}^{n} v_i(m)v_j(m)}{\sqrt{\sum_{m=1}^{n} v_i(m)^2}\sqrt{\sum_{m=1}^{n} v_j(m)^2}} \quad (1)$$

Identifier le système de description le plus performant pour identifier les chanteurs et/ou détecter s'il y a un chanteur ou non. Essayer de donner du sens à vos résultats.

A Commandes Matlab utiles

A.1 Divers

— load : permet de charger des données à partir d'un fichier ".mat"
— dir : charge la liste des fichiers présents dans un répertoire
— hist : permet de calculer un histogramme des valeurs d'une matrice
— repmat : permet la réplication de matrices
— imagesc : permet de visualiser les valeurs d'une matrice
— audioplayer : permet d'écouter un signal
— unique : sélectionne les éléments uniques d'un vecteur

A.2 Regroupement

— pdist : produit un vecteur de similarité
— squareform : convertit un vecteur de similarité en une matrice
— kmeans : implantation native de l'algorithme "k-means"

A.3 Analyse en composantes principales

— (pc,score) = princomp(V)
— score est une matrice de la même taille que V contenant les nouvelles coordonnées des données dans le système défini par les composantes principales
— plot(score(:,1),score(:,2),'+') affichera les données en fonction des deux axes expliquant le mieux la variance de ces dernières.

A.4 Documentation

— doc <command> : affiche des informations à propos de la commande <commande>.
— lookfor <keyword> : liste les fonctions ayant le mot-clé <keyword> au sein de son nom ou de sa description