Rapport de Stage

-

Représentations vectorielles de distributions de probabilités pour la similarité musicale

Aloïs Gruson

Remerciements

Je souhaiterais remercier Arshia Cont pour son cours de Machine Learning qui m'a ouvert les yeux sur un nouveau domaine qui me passionne.

Je souhaiterais aussi remercier la start-up Niland pour leur accueil très sympathique dès les entretiens pré-stage, et pour la chance qu'ils m'ont donné de les rejoindre, en stage dans un premier temps, puis pour un CDI maintenant.

Plus particulièrement, je souhaiterais remercier Christophe pour le soutien qu'il m'a apporté durant tout ce stage. Je remercie également Damien, qui me pousse à apprendre de nouvelles choses (bash, Python...), Johan, pour son travail bien utile à la recherche, et Raff, pour ses conseils côté développement. Je les remercie aussi tous pour la bonne ambiance qui règne chez nous.

Table des matières

Notations

ACP (ou PCA)	Analyse en Composantes Principales
DCT	Transformée en cosinus discrets - Discrete Cosinus Transform
DBN	Deep Belief Network
EM	Algorithme Expectation Maximisation - Espérance Maximisation
EMD	Earth Mover Distance
GMM	Gaussian Mixture Model - Modèle de mixtures gaussiennes
GMM-SV	Méthode Gaussian Mixture Model SuperVector - Méthode Mélange de gaussiennes et Supervecteur
KL	Kullback Leibler
MFCC	Mel Frequency Cepstrum Coefficients
MIR	Music Information Retrieval
RBM	Restricted Boltzmann Machine - Machine de Boltzmann Restreinte
UBM	Universal Background Model - Modèle du monde
Norme UCS	Norme UBM Centered Spherical
VQ	Vector Quantization - Quantification vectorielle
VLAD	Vector of Aggregated Local Descriptors
ZCA	Zero Component Analysis

Introduction

Le développement nouveau et rapide de la musique digitale et des plateformes d'écoute (Deezer, Spotify, Soundcloud, Pandora...) ou d'achats en ligne (Itunes, Beatport...) a créé de nouveaux besoins. La taille des catalogues ainsi que leur renouvellement quotidien (jusqu'à quelques dizaines de milliers de nouveaux titres par jour) ne permettent plus le recours à l'homme dans l'étiquetage ('tagging') des morceaux. Le recours à la similarité automatique devient alors nécessaire afin d'aiguiller les utilisateurs vers des propositions qui pourraient les intéresser.

Il existe deux grandes techniques de recommandation : la recommandation basée sur les utilisateurs et celle basée sur le signal.

La première méthode ('user-based') est utilisée par des sites comme Last.fm ou des systèmes comme Itunes Genius. Ceux ci proposent des recommandations musicales grâce au filtrage collaboratif, c'est à dire grâce à l'utilisation des retours utilisateurs ('feedbacks') explicites ou implicites. L'inconvénient de cette approche est qu'elle tend à favoriser les titres populaires, et ne permet pas la recommandation de titres peu populaires ou récemment ajoutés, sur lesquels on ne dispose donc d'aucune donnée utilisateur ('cold start problem').

D'autres encore, comme le site Pandora.com, font appel à de nombreux experts musicaux afin d'étiqueter les morceaux pour permettre la recherche par genre ou par similarité. Le site Pandora.com a lancé le Music Genome Project : bien que d'une efficacité évidente, cette méthode possède un très fort coût. On estime en effet à 20 ou 30 minutes la durée nécessaire pour annoter un son, et seuls 1 million de sons ont été annotés depuis le début du projet en 1999. C'est pourquoi la start-up Niland (http ://niland.io), propose une étude automatique basée sur le signal, avec notamment une API de recherche par similarité dans d'importantes bases de titres, API nommée Siilar.

J'ai été chargé durant ce stage d'évaluer la méthode d'apprentissage machine actuellement utilisée dans Siilar, et d'étudier les alternatives possibles à cette approche.

État de l'art

Principe général d'un système de recherche par similarité

L'approche permettant la comparaison par similarité de 2 objets est relativement commune aux différents domaines de recherche, c'est à dire l'image, la reconnaissance du locuteur, la musique... La première étape consiste à extraire des descripteurs à partir des objets à comparer. La seconde se charge de modéliser les distributions de ces descripteurs afin de construire un vecteur signature. Enfin, ces vecteurs signature sont comparés entre eux pour déterminer la similarité entre les objets considérés.

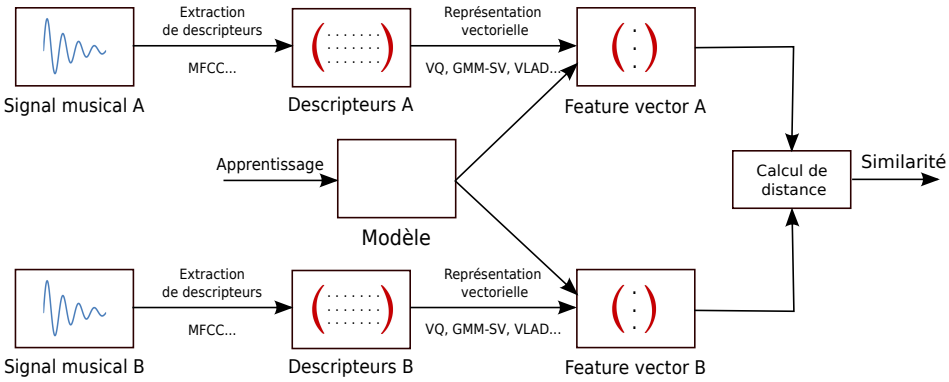


FIGURE 1 – Structure d'un système de calcul de similarité

Extraction de descripteurs

La première étape consiste à extraire des descripteurs ('features') de notre morceau. On découpe donc notre signal en trames, et on extrait des descripteurs sur chacune d'elles. Parmi les plus utilisés des descripteurs locaux, on peut citer les Mel Frequency Cepstrum Coefficients (MFCC). Utilisés depuis les années 80

dans le domaine de la reconnaissance de locuteur ou de parole, ces descripteurs ont aussi fait leurs preuves dans le domaine MIR [4] [10].

Des chercheurs ont proposé leurs propres descripteurs [23], dont certains sont basés sur la psycho-acoustique [18], ou sur des analyses à plusieurs échelles temporelles [12] [15].

Il existe aussi de nombreuses techniques destinées à l'apprentissage automatique de descripteurs à partir du spectrogramme, ce qui permettrait d'obtenir les meilleurs descripteurs pour la tâche choisie [20] [7] [13]. Ces approches appartiennent au domaine de l'apprentissage profond ('deep learning') et plus précisément à celui de l'apprentissage de représentations.

Une fois ces descripteurs extraits, on dispose d'un nuage de points représentatif de chaque morceau. L'objectif est alors de comparer ces nuages de points entre eux afin de déterminer leur similarité et donc la similarité entre les morceaux auxquelles ils sont associés.

Modèles de distribution

Une première idée serait de représenter le nuage de points représentant les descripteurs à l'aide d'un modèle gaussien (modèle Single Gaussian [17]), ou comme un modèle de distribution à plusieurs gaussiennes (Gaussian Mixture Model - GMM). Il suffit ensuite de calculer la distance entre ces modèles, notamment grâce à la divergence de Kullback Leibler (KL divergence) ou l'Earth Mover Distance (EMD).

Vers des modèles vectoriels

Le défaut de cette approche est qu'elle oblige à approximer la distribution des descripteurs de chaque morceau, et qu'elle impose de coûteux calculs de distances entre modèles.

Sur de grandes bases, il est donc inenvisageable de procéder de la sorte. L'idée d'une représentation de chaque nuage de points par un vecteur de taille fixe est alors envisagée. Ce vecteur sera appelé vecteur signature. Ainsi, une comparaison entre nuages de points se résumera à une mesure de distance entre vecteurs signature, c'est à dire entre deux vecteurs de taille identique.

L'approche GMM-Supervecteur

A l'inverse de la technique précédente, cette approche n'oblige pas à la modélisation de la distribution des descripteurs de chaque morceau [4]. On construit une seule fois un modèle de mixtures gaussiennes (Gaussian Mixture Model) représentant la distribution de tous les descripteurs de l'ensemble des morceaux musicaux grâce à l'algorithme Expectation-Maximisation (EM). Ce modèle est appelé Universal Background Model (UBM) ou modèle du monde.

Ce modèle GMM-UBM est ensuite adapté à chaque morceau (par l'approche Maximum A Posteriori), sur un nombre limité d'itérations, et chaque morceau dispose donc un modèle GMM proche du modèle commun UBM. Durant cette phase, seules les centroïdes peuvent varier, les covariances étant fixées.

On extrait enfin pour chaque morceau son supervecteur, vecteur signature regroupant les vecteurs déplacement des centroïdes au cours de l'étape adaptation de chaque morceau.

L'approche GMM-UBM-Supervecteur est l'une des plus utilisées en reconnaissance du locuteur et est présente dans le système Siilar actuel.

L'approche Quantification Vectorielle

Une autre approche est l'approche Vector Quantization (VQ) [10] [22]. Tirée notamment de l'approche Bag-Of-Words appliquée aux textes, elle consiste à créer un dictionnaire de descripteurs puis construire les vecteur signatures en codant les descripteurs des morceaux avec l'aide du dictionnaire ainsi créé. Celui ci est souvent construit par l'algorithme des K-moyennes ('K-means') [22] [6].

La création des vecteur signatures peut se faire en remplaçant chaque descripteur par l'élément du dictionnaire le plus proche [19] ou par les N plus proches voisins [22], puis en agrégeant temporellement ces éléments, que ce soit par agrégation moyenne [10] (c'est à dire en faisant la moyenne des utilisations des éléments du dictionnaire au cours du temps, ce qui revient à faire l'histogramme des apparitions des mots) ou par agrégation maximale [24] [20].

D'autres approches possibles

D'autres chercheurs ont proposées des techniques de recherche par similarité compétitives sur de grandes bases de données. A l'interface entre GMM-SV et VQ, l'approche Vector of Aggregated Local Descriptors (VLAD) [8] [1] utilise un dictionnaire (construit de la même façon que VQ par exemple, ou grâce à l'algorithme EM), mais le vecteur signature d'une morceau est composé par la concaténation des sommes des résiduels descripteur-centroïde le plus proche. Sur le même principe, on peut citer l'approche Bossa Nova [2]. Des techniques comme la Factorisation en Matrices Non Négatives (NNMF) ont aussi été testées pour la classification musicale [3].

Les approches par apprentissage profonds

Les recherches récentes en image (reconnaissance de formes notamment) ou en audio (en reconnaissance du locuteur) ouvrent la porte à une nouvelle branche de recherche pour la musique : les techniques dites d'apprentissage profond ('deep learning') semblent très prometteuses.

Ces méthodes sont issues des recherches sur les réseaux de neurones ('neural networks') artificiels, réseaux supposés apprendre eux même les représentations de données les plus pertinentes à partir d'exemples annotés ou non.

Ces approches ont déjà été testées, que ce soit à l'aide d'une machine de Boltzmann restreinte (RBM) [20], de 'Deep Belief Networks' (DBN) appliqués sur le spectre du signal avec une approche conjointe supervisée et non supervisée [11], avec un réseau de neurones convolutionnel pré-entraîné [9] ou entraîné en apprentissage non supervisé uniquement [16]. Un autre type de réseau composé de machines de Boltzmann Moyenne-Covariance ('Mean-Covariance Restricted Boltzmann Machine', MC-RBM) est utilisé pour des mesures de similarité musicale dans l'article [21].

Bien que les résultats de ces méthodes n'aient pas encore remis en question les approches traditionnelles, elles semblent très prometteuses sur le plan musical pour de nombreuses raisons développées dans [13] [14].

Chapitre 1

Méthodes d'évaluation de la similarité musicale

1.1 Critère d'évaluation sur les genres

1.1.1 Présentation du critère

Notre premier critère d'évaluation utilisera l'hypothèse qu'une musique d'un même genre à la requête est une bonne proposition pour la similarité. En comptant le nombre titres du même genre dans le top K de tous les titres, on peut obtenir le pourcentage de 'bonnes recommandations' du top 10 de chacun des titres. La moyenne de ces pourcentages nous donne le critère qu'on nommera Average Ratio of Genre Matches. Comme souvent dans la littérature, nos résultats seront artist-filtered, ce qui signifie qu'on élimine les recommandations de l'artiste de la requête dans le top de cette requête. On s'intéressera dans la suite aux valeurs de K suivantes : $[1, 3, 5, 10, 20]$.

Requête	Rock	Classique	Metal
Proposition 1	Pop	Classique	Rock
Proposition 2	Rock	Electro	Metal
Proposition 3	Rock	Pop	Electro
Proposition 4	Metal	Classique	Metal
Proposition 5	Electro	Classique	Metal
Scores	2/5 =0.4	3/5 =0.6	3/5 =0.6

Score final : (0.4+0.6+0.6)/3 = 53%

FIGURE 1.1 – Calcul du critère Average Ratio of Genre Matches pour le Top 5

1.1.2 Bases d'évaluation du critère

Nous disposons d'une base de 15031 titres annotés sur 10 genres : Electro Ambient, Electro Techno, Pop, Rock, Reggae, Metal, Classique, Country, Folk, et Soul. Cette base est découpée en une base d'apprentissage, qui représente 2/3 de la base totale, et une base de test qui représente donc le tiers restant de la base. Ces bases sont construites afin que les ratios entre genres soit identiques dans les deux bases.

La base d'apprentissage sert à calculer les modèles nécessaires (modèle du monde UBM, centroïdes pour l'approche VQ ou Vlad...) et contient 10323 titres. Nous utilisons cependant une base d'entraînement réduite de 4880 titres (soit 488 titres par genre), équilibrée entre les genres, afin d'éviter une éventuelle sur-représentation d'un genre dans les modèles appris.

Nous avons choisi d'exclure le genre pop de la base de test, estimant qu'il était trop difficile à distinguer (notamment du genre Rock). La base de test est alors composée de 4706 titres et permet d'estimer la qualité des modèles appris grâce au critère Ratio Moyen de Genres Corrects. Sa répartition en genres est donnée par le tableau ci dessous.

Genre	Nombre de morceaux
Classique	466
Ambient	90
Techno	404
Folk	232
HipHop/RnB	838
Metal	245
Reggae	404
Rock	1363
Soul	664

FIGURE 1.2 – Répartition en genres de la base de test

1.2 Second critère d'évaluation

1.2.1 Présentation du critère

Notre second critère se base sur une base de morceaux composés de compilations très homogènes en leur sein. On peut donc, à partir d'une requête issue d'une compilation donnée, espérer avoir des suggestions de similarité qui appartiennent à cette même compilation. Le critère choisi est alors très proche de celui défini précédemment pour les genres. On compte le nombre de suggestions dans

les top K de similarité qui font partie de la même compilation que la requête. On s'intéressera aux valeurs de K suivantes : $[1, 3, 5, 10, 20]$.

1.2.2 Bases d'évaluation du critère

La base de compilations contient 8329 titres répartis en 293 compilations. On ne sépare pas cette base, car il s'agit uniquement d'une base de test, la base d'apprentissage utilisée étant la même que pour le premier critère.

1.3 Filtrage collaboratif

Le recours au filtrage collaboratif (et donc aux retours utilisateurs) peut permettre de se faire une idée sur la qualité d'un système de recommandation par similarité. Cette approche est décrite dans l'article [19].
Nous avons vu que le problème du filtrage collaboratif est l'absence de données sur les morceaux peu populaires. Ainsi, un système qui ne se baserait que sur le filtrage collaboratif ne pourrait pas juger de la qualité d'une recommandation sur ces morceaux peu écoutés. L'idée est donc de se concentrer sur les morceaux pour lesquels il existe des retours utilisateurs. On peut ainsi imaginer construire une bibliothèque qui ne contient que des morceaux avec des retours utilisateurs associés. Il existe des applications comme celle de Lastfm qui permettent de récupérer les titres favoris d'un auditeur. On estime la valeur de la similarité $S(A, B)$ entre un morceau A et un morceau B de la façon suivante :

$$S(A, B) = \frac{\textit{Nombre d'utilisateurs aimant A et B}}{\textit{Nombre d'utilisateurs aimant A ou B}} \tag{1.1}$$

Ainsi, plus la valeur de la similarité entre deux sons est forte, plus ces sons seront jugés similaires.

L'évaluation se fait alors avec des critères classiques, en observant par exemple la position des morceaux similaires dans les propositions d'un morceau requête.

1.4 Evaluation perceptive

La dernière étape qui nous permet de valider un nouveau système de comparaison par similarité est l'étape de validation subjective. C'est la seule qui évalue directement la similarité, bien qu'elle soit subjective. Nous disposons d'une interface web qui propose, à partir d'un morceau, d'utiliser les différents systèmes à comparer et de proposer à l'écoute, sans indiquer le système utilisé, le K-ème morceau retourné par chaque système. Chaque membre de l'équipe est ensuite chargé de noter la similarité entre le morceau source et les morceaux proposés ('pas similaire' (valeur 0), 'moyennement similaire' (valeur 1), 'très similaire' (valeur 2)). On calcule ensuite le score moyen de chaque système, ce qui permet d'observer si un nouveau système apporte une amélioration significative perceptivement.

Chapitre 2

Extractions de descripteurs

La première étape d'un système de recherche par similarité consiste à extraire des descripteurs pour chaque morceau à comparer. Au cours de mon stage, je me suis concentré sur le descripteur le plus utilisé dans le domaine audio (traitement de la parole et musique) : les Mel Frequency Cepstrum Coefficients (MFCC). Je détaille dans un premier temps la méthode de calcul des MFCC, et j'expliquerai ensuite les pré-traitements subis par ces descripteurs avant la modélisation vectorielle. Enfin, je parlerai en troisième partie de l'ouverture de mon stage vers une partie de ma thèse à venir : l'apprentissage de représentation par apprentissage profond.

2.1 Principe de calcul des MFCC

Les Mel Frequency Cepstral Coefficients (MFCC) sont les descripteurs les plus utilisés depuis les années 80 dans le domaine de la reconnaissance du locuteur et dans le domaine MIR. Voici les étapes permettant d'obtenir les MFCC :

- On multiplie le signal original par une fenêtre glissant temporellement. Généralement, cette fenêtre est longue de 40 à 60 ms et a un pas d'avancement temporel de 20 à 30 ms (pour un recouvrement de 50%)
- On calcule la transformée de Fourier de chaque fenêtre
- On filtre le spectre par un banc de filtres Mel (les détails concernant les filtres Mel sont donnés ci dessous), généralement 40 filtres
- On applique une compression par la fonction log
- On applique une transformée en cosinus discrète (Discrete Cosinus Transform - DCT) pour réduire la dimension, la dimension d'arrivée étant généralement 13

2.1.1 Filtres Mel

Les filtres Mel sont des filtres répartis logarithmiquement en fréquence, et dont la largeur et l'amplitude varient afin de conserver une énergie constante. La

FIGURE 2.1 – Méthode de calcul des MFCC

figure ci dessous donne un exemple de filtres Mel. Ces filtres ont été créés dans les années 30 en psycho-acoustique et modélisait le fonctionnement de l'oreille humaine. Bien qu'ils ne soient plus utilisés en psycho-acoustique aujourd'hui, ils restent important dans le domaine de la parole ou de la musique.

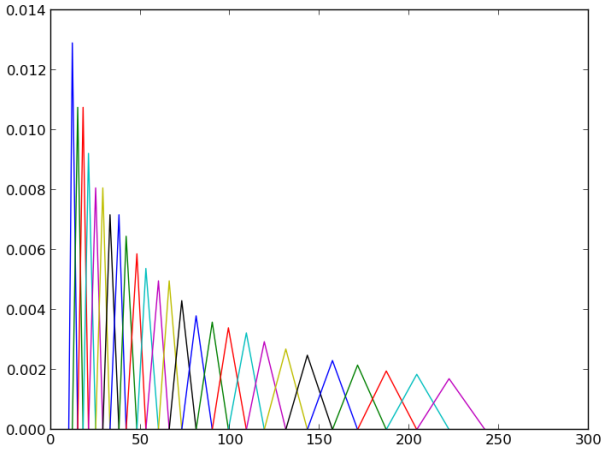


FIGURE 2.2 – Exemple de filtres Mel

La formule permettant de passer de l'échelle des fréquences à l'échelle Mel est

$$m = 2595 * log_{10}(1 + f/700) \tag{2.1}$$

et donc, celle permettant de repasser à l'échelle de fréquences classiques :

$$f = 700 * (10^{\frac{m}{2595}} - 1) \tag{2.2}$$

2.2 Blanchiment des données

Dans de nombreux cas d'apprentissage machine, les données doivent être "blanchies". Le blanchiment consiste à donner la même importance à chaque

composante des données afin qu'une composante qui serait de grande amplitude "n'étouffe" pas les autres.

2.2.1 Composantes centrées réduites

Un blanchiment classique consiste à calculer la moyenne et la variance de chaque composante sur l'ensemble des exemples, et à soustraire la moyenne sur chaque composante, puis à diviser par sa variance.

Ainsi, sur chaque composante

$$x_{CR}(j,i) = \frac{x(j,i) - \mu_i}{\sigma_i} \tag{2.3}$$

Chaque composante est alors une variable centrée réduite.

2.2.2 Blanchiment par ACP

Le blanchiment par Analyse en Composantes Principales (ACP) permet non seulement de normaliser l'amplitude de chaque composante, mais aussi de décorréler les composantes entre elles. Le but de l'ACP est de transformer la matrice de corrélation en la matrice identité. Ainsi, chaque composante est décorrélée des autres, et a une variance unitaire.

Les détails mathématiques concernant l'ACP et plus particulièrement le blanchiment par ACP peuvent être trouvés dans l'Appendix A.

2.2.3 Blanchiment par ZCA

Le blanchiment par Zero Component Analysis (ZCA) est une technique de blanchiment des données souvent utilisée, qui correspond au blanchiment par ACP auquel on ajoute une rotation qui repasse les données dans l'espace initial par rotation inverse. La figure ci dessous montre un exemple de blanchiment par ACP et un blanchiment par ZCA.

Les détails mathématiques concernant le blanchiment par ZCA peuvent être trouvés dans l'Appendix B.

2.3 Ouverture : Vers l'apprentissage de représentation

D'autres descripteurs sont utilisables afin de représenter un morceau musical. Le problème principal est de savoir lesquels. De nombreux chercheurs ont proposé de nouveaux descripteurs censés apporter une meilleure représentation de la musique, mais aucun n'est parvenu à détrôner réellement les MFCC dans la communauté MIR. Le développement récent des techniques dites d'apprentissage de représentation, issues du domaine de l'apprentissage profond ('deep learning') ouvre la voie à la création automatique de descripteurs appropriés

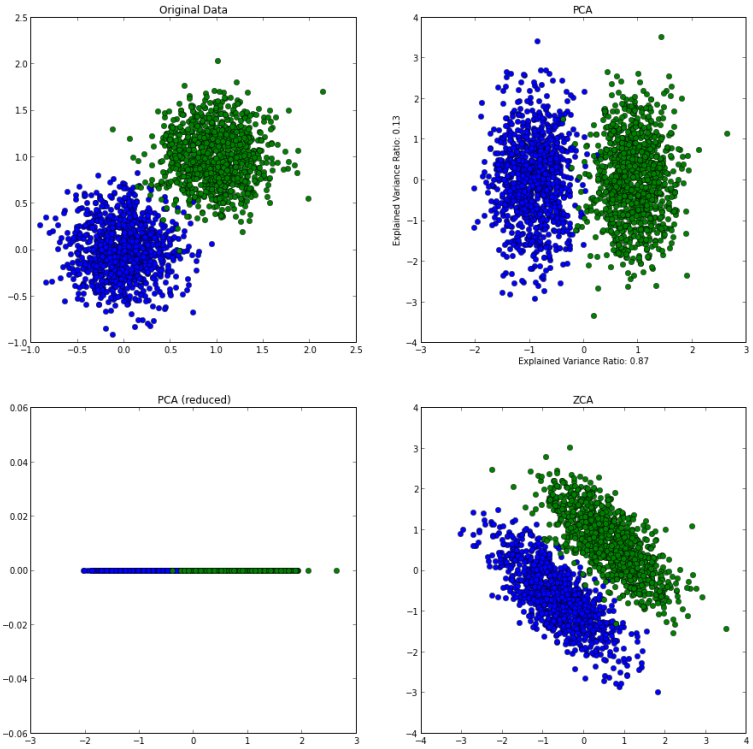


FIGURE 2.3 – Exemple d'applications du blanchiment par ACP et par ZCA

pour la tâche choisie. Ces techniques ont fait leur preuve dans les domaines de l'image ou de la reconnaissance du locuteur.

Chapitre 3

La Quantification Vectorielle : VQ

3.1 Présentation de l'approche

La méthode Quantification Vectorielle (Vector Quantization, VQ) consiste à créer un dictionnaire de descripteurs puis à associer à chacune des pistes un vecteur représentant les apparitions des éléments du dictionnaire. Nous allons dans un premier temps détailler les différentes façons possibles de créer ce dictionnaire puis nous étudierons dans un deuxième temps les codages possibles des descripteurs à l'aide du dictionnaire ainsi créé. Nous comparerons ensuite les différentes agrégations temporelles de ces codages ('pooling') afin former le vecteur signature. Enfin, nous parlerons des différentes mesures possibles de distance entre vecteurs signature pour déterminer la similarité entre les pistes.

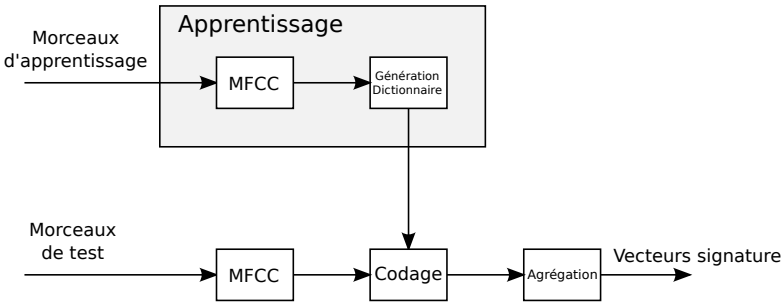


FIGURE 3.1 – Système de calcul des vecteurs signature par VQ

3.2 Création d'un dictionnaire

3.2.1 Avec l'algorithme K-moyennes

L'algorithme K-moyennes est le premier qui vient à l'esprit : il permet en effet de définir K centroïdes représentant un nuage de points. A chaque centroïde est associé un ensemble de points. Le principe de ce algorithme est de minimiser les distances centroïdes-points associés, c'est à dire de trouver l'ensemble $S = (S_1, ..., S_n)$ qui vérifie

$$\arg\min_{S} \sum_{i=1}^{k} \sum_{x_j \in S_i} ||x_j - \mu_i||^2 \text{ avec } \mu_i \text{ la moyenne des points dans } S_i \quad (3.1)$$

avec $||.||$ la norme L_2.

Voici le déroulement de l'algorithme :

- Initialiser K centroïdes $m_1^{(1)}, ..., m_K^{(1)}$ (en sélectionnant par exemple K descripteurs pris au hasard, ou en les générant à l'aide d'une distribution normale)
- Répéter jusqu'à convergence :
  - assigner chaque point à sa partition la plus proche

$$S_i^{(t)} = \left\{ x_j : ||x_j - m_i^{(t)}|| \leq ||x_j - m_{i^*}^{(t)}|| \ \forall \ i^* = 1, ..., k \right\} \quad (3.2)$$

  - Mettre à jour la moyenne de chaque partition

$$m_i^{(t+1)} = \frac{1}{|S_i^{(t)}|} \sum_{x_j \in S_i^{(t)}} x_j \quad (3.3)$$

Le critère de convergence peut être la non-modification des points associés à chaque partition entre deux itérations ou un nombre maximal d'itérations.

3.2.2 Sélection aléatoire de descripteurs

En allant à la simplicité, on peut penser à sélectionner aléatoirement des MFCC pour en faire un dictionnaire. Cette approche donne de très bons résultats en plus d'accélérer le temps de calcul, comme constaté dans [5] et dans nos calculs. De plus, la création du dictionnaire devient instantanée, ce qui est intéressant sans pour autant être indispensable.

3.3 Codage des descripteurs

Vient ensuite l'étape de codage de chaque morceau : il s'agit de représenter l'ensemble des descripteurs du morceau par un seul vecteur, le vecteur signature. Il faut donc coder chacun des descripteurs avec le dictionnaire défini précédemment.

3.3.1 Codage sur les plus proches voisins

Ce codage consiste à trouver l'élément du dictionnaire le plus proche du descripteur considéré et d'incrémenter de 1 la composante de l'élément considéré :

$$f_k(x) = \begin{cases} 1 & \text{if } k = \arg\min_j ||c_j - x||_2^2 \\ 0 & \text{otherwise} \end{cases} \tag{3.4}$$

On peut aussi considérer les N plus proches voisins et incrémenter le vecteur de 1 sur leur composante. Notons que cela revient à remplacer chaque descripteur par son élément de dictionnaire le plus proche, puis à construire l'histogramme de ces éléments.

Le schéma 3.2 permet de résumer la création du dictionnaire et le codage d'un descripteur avec ce dictionnaire.

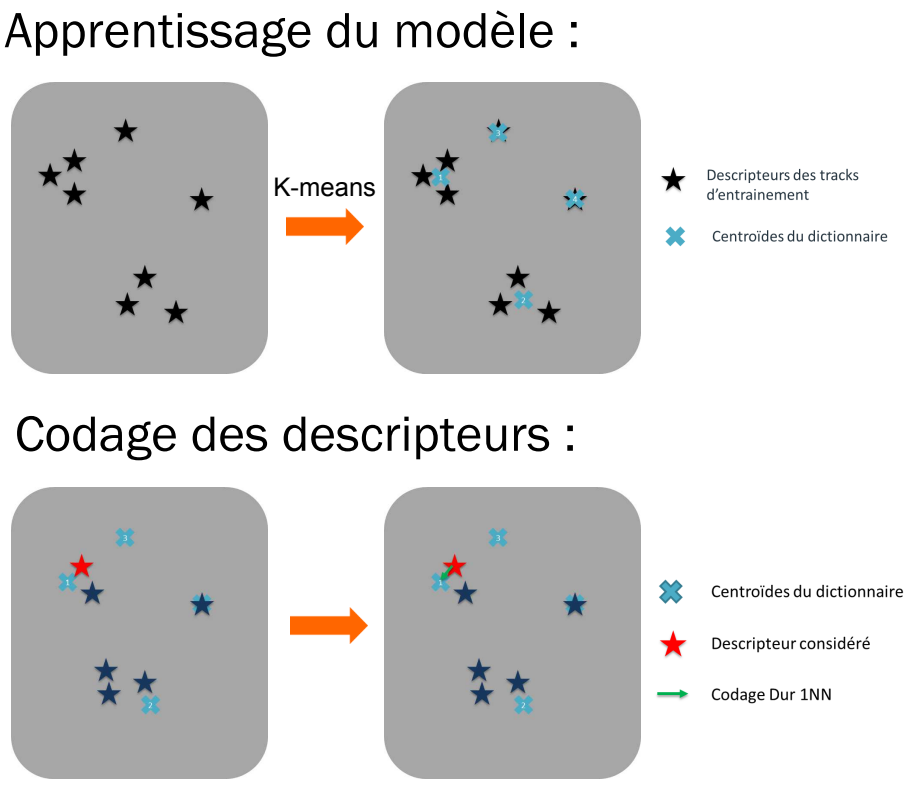


FIGURE 3.2 – Schéma explicatif VQ - Codage dur

3.3.2 Codage doux

[6] propose un codage doux ('soft assignment') défini de la manière suivante :

$$f_k(x) = \max\{0, \mu(z) - z_k\} \tag{3.5}$$

Ce codage est censé améliorer la précision (les centroïdes les plus proches ont plus de poids, et ils sont incrémentés plus ou moins en fonction de leur distance au point considéré) tout en conservant une certaine parcimonie (on ne prend pas en compte les centroïdes plus loin que la moyenne des distances).

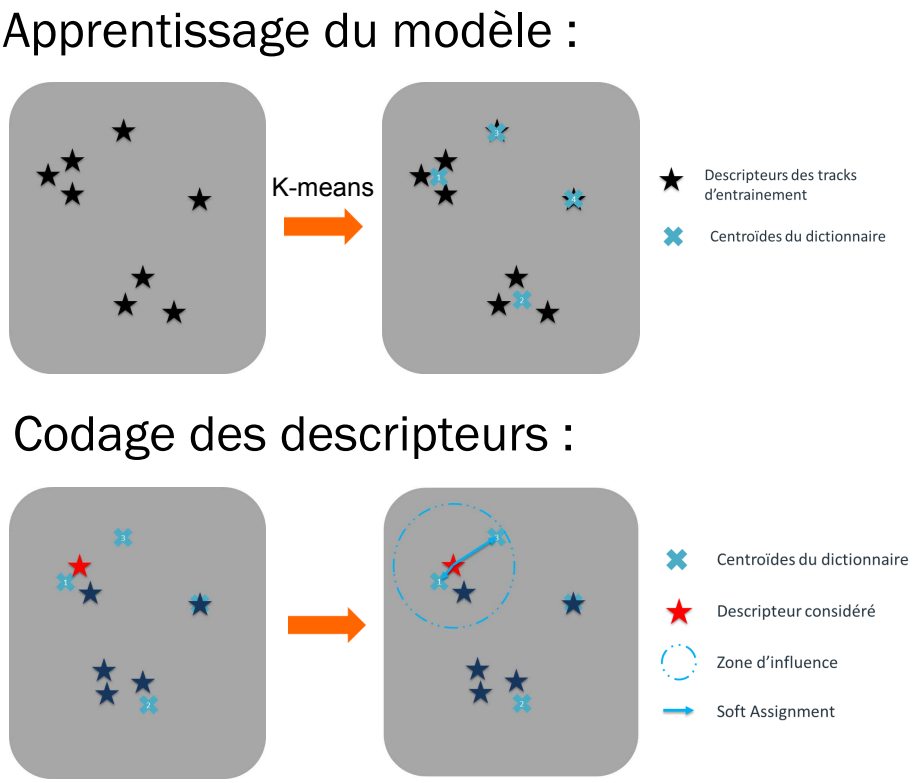


FIGURE 3.3 – Schéma explicatif VQ - Codage doux

3.4 Agrégation temporelle des codages

Il existe différents types d'agrégation des codages locaux ('pooling'). On peut faire la moyenne des codages de chacun des descripteurs ('mean-pooling'), la somme ('sum-pooling') ou prendre le maximum sur chacune des composantes ('max-pooling') afin de former le vecteur signature du morceau.

Les résultats présentés ci dessous donnent clairement l'avantage au max-pooling. Mes expériences préliminaires consistant à utiliser plusieurs types d'agrégation en parallèle (le vecteur signature est alors la concaténation du mean-

pooling et du max-pooling par exemple) n'ont pas donné de résultats suffisamment intéressants pour poursuivre sur cette piste.

J'ai donc choisi de poursuivre avec le max-pooling uniquement.

3.5 Mesures de distance entre vecteurs signature

Il faut enfin calculer la distance entre deux vecteurs signature afin de déterminer les similarités entre pistes.

Notons que cette étape est la seule étape dont le temps de calcul doit être optimisé. En effet, ce calcul de distances entre vecteurs signature est exécuté à chaque requête utilisateur et doit avoir lieu dans des délais inférieurs à la seconde sur des bases de plusieurs millions de titres.

3.5.1 Distance euclidienne

La distance la plus naturelle et la plus rapide à calculer est la distance euclidienne. Il suffit d'utiliser le produit scalaire entre la différence des vecteurs signature v_i et v_j et son conjugué.

$$d(v_i, v_j) = ||v_i - v_j||^2 \tag{3.6}$$

avec $||.||$ la norme L_2.

3.5.2 D'autres distances possibles

J'ai fait quelques tests avec des distances χ^2 et intersection d'histogrammes, les résultats n'étaient pas améliorés, et le temps de calcul se trouvait fortement augmenté. Ce calcul étant exécuté à chaque requête utilisateur et étant donc critique en terme de temps de calcul, j'ai décidé de me concentrer sur la distance euclidienne dans la suite de mes expérimentations.

La distance χ^2 est donnée par :

$$d_{\chi^2}(v_i, v_j) = \sum_k \frac{(v_i(k) - v_j(k))^2}{v_i(k) + v_j(k) + \epsilon} \tag{3.7}$$

avec ϵ une petite valeur qu'on ajoute pour éviter une éventuelle instabilité numérique.

La distance par intersection d'histogrammes est quant à elle définie par :

$$d_{\mathrm{HI}}(v_i, v_j) = \sum_k max(v_i(k), v_j(k)) \tag{3.8}$$

3.6 Résultats obtenus par la méthode VQ

* Retrait de la moyenne par composantes et par morceau
* Concaténation temporelle 6 trames (dimension 6*17 =102)
* Blanchiment par PCA -> dimension 40
* VQ avec codage dur

Nombre de centroïdes	Scores				
	1	3	5	10	20
512	51.9	49.8	48.6	46.9	44.8
1024	56.1	53.6	52.0	50.3	48.0
2048	51.5	49.0	48.0	46.2	44.1

FIGURE 3.4 – Résultats obtenus pour différents nombres de centroides avec codage dur

* Retrait de la moyenne par composantes et par morceau
* Concaténation temporelle 6 trames (dimension 6*17 =102)
* Blanchiment par PCA -> dimension 40
* VQ avec codage doux

Nombre de centroïdes	Scores				
	1	3	5	10	20
512	55.2	52.2	50.6	47.9	44.8
1024	52.8	50.1	49.1	46.9	44.4
2048	53.4	50.8	49.4	47.5	44.9

FIGURE 3.5 – Résultats obtenus pour différents nombres de centroides avec codage doux

Chapitre 4

Des propositions d'amélioration de VQ

4.1 Retrait de la moyenne

Le retrait sur chacune des composantes MFCC de sa moyenne au long du morceau améliore sensiblement les résultats. Le tableau suivant montre les améliorations apportées par cette opération simple.

Le principal intéret de la représentation cepstral est trouvé en reconnaissance de la parole, ou dans tous les domaines qui pratiquent la séparation source/filtre. Ainsi, sur le cepstre, l'influence du filtre est représentée par sa composante continue tandis que la source représente la composante temporelle. On pourrait donc supposer que le retrait de la moyenne des composants MFCC (ces moyennes étant calculées indépendamment sur chaque morceau) permet de se libérer des effets dus à la masterisation par exemple.

* Retrait de la moyenne par composantes et par morceau
* VQ 2048 avec codage dur sur le plus proche voisin

Retrait de la moyenne	Scores				
	1	3	5	10	20
Non	46.3	43.8	42.4	40.1	37.2
Oui	46.6	43.8	42.6	40.9	38.6

FIGURE 4.1 – Résultats obtenus avec ou sans retrait de la moyenne avec VQ codage dur

* Retrait de la moyenne par composantes et par morceau
* VQ 2048 avec codage doux

Retrait de la moyenne	Scores				
	1	3	5	10	20
Non	47.1	43.2	41.8	39.7	37.3
Oui	45.6	42.9	41.5	39.6	37.1

FIGURE 4.2 – Résultats obtenus avec ou sans retrait de la moyenne avec VQ codage doux

4.2 Agrégation temporelle et blanchiment par ACP

L'idée de cette amélioration est de prendre en compte la corrélation temporelle qui existe entre trames successives, selon la méthode présentée dans [20]. On vectorise plusieurs trames successives (de 2 à 20 généralement) et on applique un blanchiment par ACP sur les nouveaux descripteurs, tout en ne sélectionnant qu'une partie des composantes. On applique ensuite normalement la méthode VQ sur ces descripteurs modifiés.

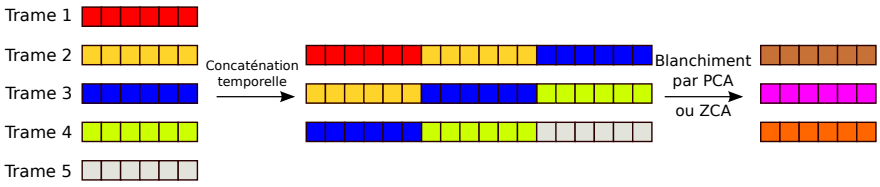


FIGURE 4.3 – Exemple Agrégation temporelle sur 3 trames + blanchiment par ACP

* Retrait de la moyenne par composantes et par morceau
* Concaténation temporelle 6 trames (dimension 6*17 =102)
* Blanchiment par PCA -> dimension 40
* VQ 2048 avec codage dur

Concaténation + PCAw	Scores				
	1	3	5	10	20
Non	46.6	43.8	42.6	40.9	38.6
Oui	51.5	49.0	48.0	46.2	44.1

FIGURE 4.4 – Résultats obtenus avec ou sans concaténation temporelle avec VQ codage dur

On observe comme attendu une forte augmentation des résultats grâce à la dimension temporelle capturée lors de cette amélioration.

* Retrait de la moyenne par composantes et par morceau
* Concaténation temporelle 6 trames (dimension 6*17 =102)
* Blanchiment par PCA -> dimension 40
* VQ 2048 avec codage doux

Concaténation + PCAw	Scores				
	1	3	5	10	20
Non	45.6	42.9	41.5	39.6	37.1
Oui	53.4	50.8	49.4	47.5	44.9

FIGURE 4.5 – Résultats obtenus avec ou sans concaténation temporelle avec VQ codage doux

4.3 ACP post-codage

Nous avons vu précédemment que l'utilisation de dictionnaires de taille plus importante permet d'améliorer les résultats. Cependant, cette augmentation de taille entraîne une augmentation de la taille des vecteurs signature, ce qui ralentit les calculs de distance entre vecteurs signature. Or, ces calculs sont critiques car exécutés à chaque requête utilisateur, et il est donc impossible de dépasser une certaine taille pour les vecteurs signature. On cherche donc à garder l'avantage apporté par de grands dictionnaires tout en conservant une taille limitée pour les vecteurs signature. L'idée est alors d'utiliser l'Analyse en Composantes Principales (ACP) afin de réduire la taille des vecteurs signature en essayant de conserver les informations qu'ils contiennent.

L'Analyse en Composantes Principales détermine les axes de variance maximale, et projette les données dans ce nouvel espace. La procédure mathématique de l'ACP est décrite en Appendix A.

Le nouveau système global est représenté figure 4.6.

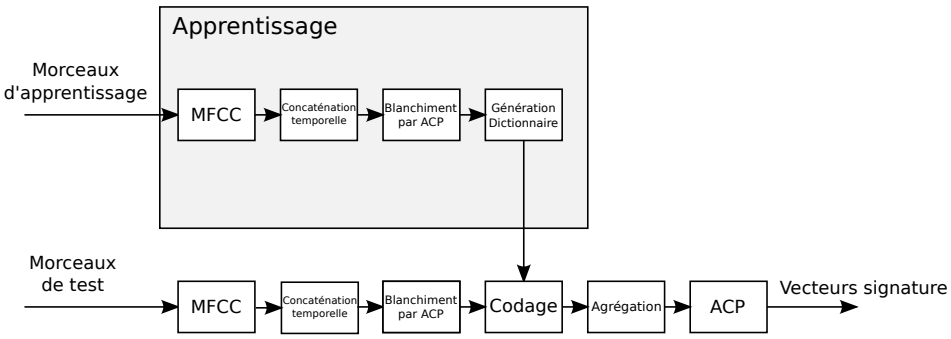


FIGURE 4.6 – Organisation du système modifié de calcul des vecteurs signature par VQ

* Retrait de la moyenne par composantes et par morceau
* Concaténation temporelle 6 trames (dimension 6*17 =102)
* Blanchiment par PCA -> dimension 40
* VQ 2048, codage dur
* PCA

Variance conservée	Dimension de sortie	Scores				
		1	3	5	10	20
1	2048	51.5	49.0	48.0	46.2	44.1
0.9	1003	51.9	49.6	48.2	46.7	44.8
0.5	247	52.8	51.3	50.0	48.3	46.2
0.3	132	53.9	51.3	50.4	48.3	46.6

FIGURE 4.7 – Résultats obtenus pour différentes réductions de dimension par ACP post-VQ avec codage dur

* Retrait de la moyenne par composantes et par morceau
* Concaténation temporelle 6 trames (dimension 6*17 =102)
* Blanchiment par PCA -> dimension 40
* VQ 2048, codage doux
* PCA

Variance conservée	Dimension de sortie	Scores				
		1	3	5	10	20
1	2048	53.4	50.8	49.4	47.5	44.9
0.9	1051	53.5	51.5	50.3	48.8	45.4
0.5	70	53.4	50.9	49.8	47.5	44.9
0.3	12	36.4	36.5	35.7	35.6	34.7

FIGURE 4.8 – Résultats obtenus pour différentes réductions de dimension par ACP post-VQ avec codage doux

On observe sur ces résultats que l'ACP post-codage est très intéressante lorsqu'on utilise le codage dur. On réduit en effet la dimension des vecteurs signatures tout en améliorant les résultats. Les résultats lorsque le codage doux est utilisé sont bien plus mitigés, avec une baisse sensible des résultats avec la diminution de la variance conservée.

Chapitre 5

Approche GMM-Supervecteur

Dans ce chapitre, je présente d'abord une deuxième technique de représentation vectorielle, GMM-Supervecteur. Il s'agit de la méthode utilisée au moment de mon arrivée en stage. Dans un second temps, j'expliquerai brièvement en quoi consiste la norme UBM Centered Spherical (norme UCS) et pourquoi nous avons choisi de l'utiliser. Dans un troisième temps, je détaillerai les améliorations apportées au cours de mon stage à cette approche. Enfin, je présenterai les résultats obtenus.

5.1 Présentation de l'approche

5.1.1 Modèle GMM-UBM

On modélise la distribution des descripteurs de toutes les morceaux à l'aide d'un Gaussian Mixture Model (GMM). Ce modèle est construit grâce à l'algorithme Espérance-Maximisation ('Expectation-Maximisation' - EM), et donne ce qu'on appelle le modèle du monde ou Universal Background Model (UBM).

Pour des raisons de coût de calcul, la matrice de covariance du modèle GMM-UBM est fixée diagonale. Il est observé qu'une augmentation du nombre de gaussiennes permet de compenser la perte d'informations due à cette limitation.

5.1.2 Obtention du supervecteur

Afin de déterminer le vecteur signature de chaque morceau, on adapte le modèle UBM grâce à une approche Maximum A Posteriori (MAP) pour chaque morceau. Seules les moyennes des gaussiennes peuvent varier, les variances sont fixées.

Ainsi, on dispose de N vecteurs déplacements des centroïdes (avec N le nombre de gaussiennes dans l'UBM). On concatène ces vecteurs déplacements

pour former le supervecteur. Ce supervecteur est donc de dimension ($N * M$,1) avec M la dimension de l'espace des descripteurs.

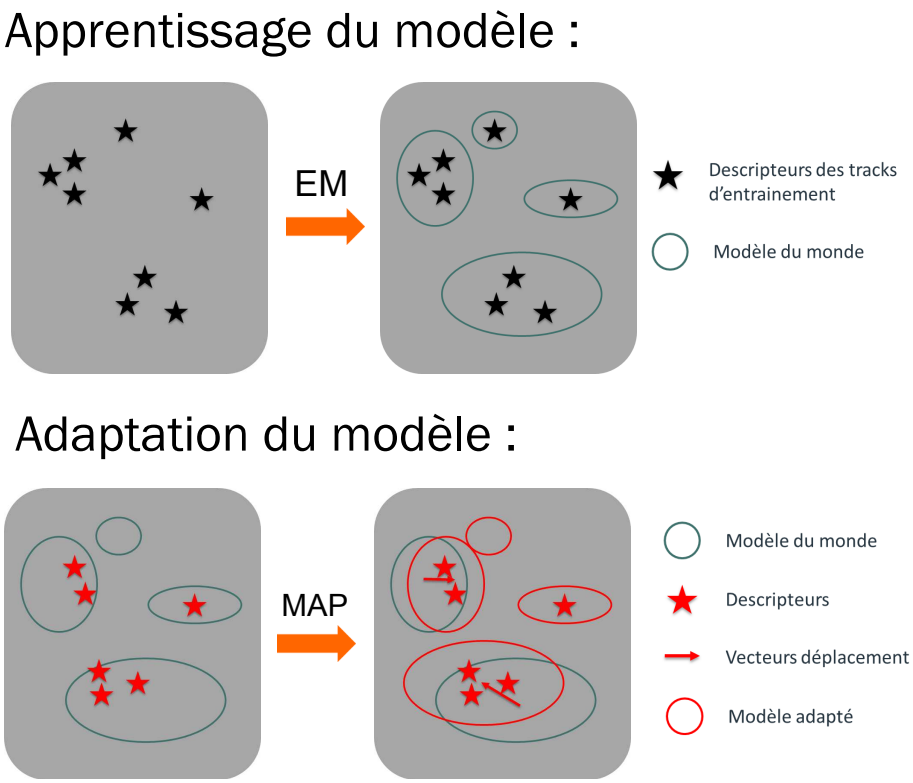


FIGURE 5.1 – Schéma explicatif GMM-SV

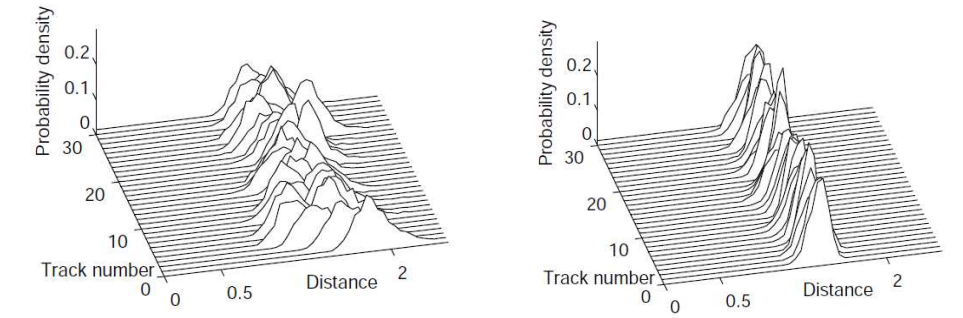
5.2 Hubs, orphelins et norme UCS

Un problème connu des systèmes de recommandation est l'existence de titres qui reviennent trop souvent ('hubs') et celle de titres qui ne reviennent jamais ('orphans'). Une solution a été proposée dans [4]. Elle consiste à appliquer une norme qui permet à chaque morceau d'avoir une distribution de distances avec les autres morceaux de moyenne 1 et d'écart type fixé. Cette norme est nommée UBM Centered Spherical Normalization car elle consiste à projeter les supervecteurs sur une sphère unitaire centrée sur le modèle du monde. On représente son effet figure 5.2.

D'un point de vue mathématique, cela consiste à appliquer les opérations suivantes :

$$\mu^a_{UCS} = \frac{\mu^a - \mu^{\text{UBM}}}{||\mu^a - \mu^{\text{UBM}}||} \tag{5.1}$$

avec μ^a le supervecteur de la morceau a à normaliser et μ^{UBM} le supervecteur correspondant au modèle du monde. $||.||$ correspond à la norme L_2.



Distribution de distances entre supervecteurs : chaque courbe représente l'histogramme des distances entre un supervecteur et le reste de la base

Distribution de distances entre supervecteurs après application de la UCS-Norm

FIGURE 5.2 – Représentation de l'effet de la norme UCS

Ceci a pour effet de rapprocher les morceaux les plus lointaines de l'ensemble des autres morceaux, et d'en éloigner les plus proches. On évite ainsi l'apparition trop fréquente de certaines morceaux au dépend d'autres. On observe que l'histogramme des apparitions de chaque morceau dans le top 10 des autres morceaux est plus équilibré.

Les résultats de l'évaluation Average Ratio of Genre Matches sont aussi légèrement améliorés par l'application de cette norme.

5.3 Améliorations apportées

Le système final est décrit par la figure 5.3.

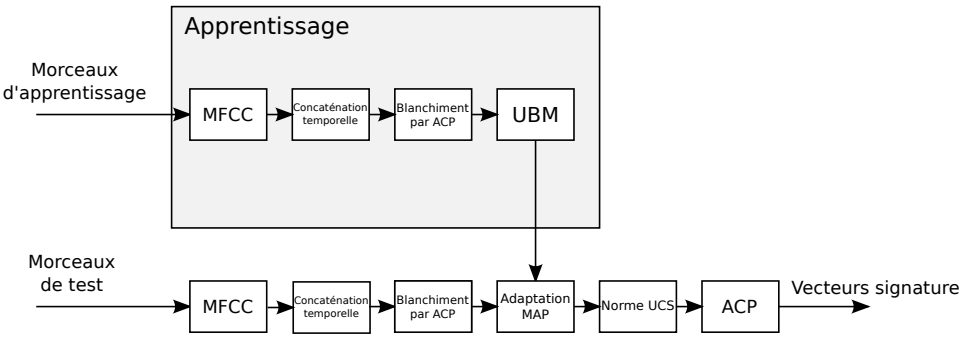


FIGURE 5.3 – Système final de calcul des vecteurs signature par GMM-SV

5.4 Résultats obtenus

5.4.1 Influence du retrait de la moyenne

* Retrait de la moyenne par composantes et par morceau
* SV 128

Retrait de la moyenne	Scores				
	1	3	5	10	20
Non	48.2	46.7	45.8	43.7	40.9
Oui	53.7	49.5	48.0	45.9	43.4

FIGURE 5.4 – Résultats obtenus avec ou sans retrait de la moyenne

On observe une forte augmentation des résultats lorsqu'on retire la moyenne des MFCC par composantes et par morceau.

5.4.2 Influence de la concaténation temporelle

* Retrait de la moyenne par composantes et par morceau
* Concaténation temporelle 6 trames (dimension 6*17 =102)
* Blanchiment par PCA -> dimension 40
* GMM-SV 128

Concaténation + PCAw	Scores				
	1	3	5	10	20
Non	53.7	49.5	48.0	45.9	43.4
Oui	57.7	54.2	52.4	50.0	47.0

FIGURE 5.5 – Résultats obtenus avec ou sans concaténation temporelle + Blanchiment par ACP

Comme pour les approches VQ, la concaténation temporelle apporte un vrai plus dans les résultats.

5.4.3 Influence du nombre de centroides

* Retrait de la moyenne par composantes et par morceau
* Concaténation temporelle 6 trames (dimension 6*17 =102)
* Blanchiment par PCA -> dimension 40
* GMM-SV

Nombre de centroïdes	Scores				
	1	3	5	10	20
64	56.5	54.1	52.6	49.7	46.9
128	57.7	54.2	52.4	50.0	47.0
256	58.2	55.2	53.4	51.0	48.0

FIGURE 5.6 – Résultats obtenus pour différents nombres de centroides

L'influence du nombre de centroides est celle attendue, avec une amélioration des résultats qui suit l'augmentation du nombre de centroides.

5.4.4 Influence de l'ACP post-SV

* Retrait de la moyenne par composantes et par morceau
* Concaténation temporelle 6 trames (dimension 6*17 =102)
* Blanchiment par PCA -> dimension 40
* GMM-SV 128
* PCA

Variance conservée	Dimension de sortie	Scores				
		1	3	5	10	20
1	5120	57.7	54.2	52.4	50.0	47.0
0.9	1810	57.2	54.2	52.5	50.1	47.5
0.5	332	56.9	54.3	53.0	50.5	47.9
0.3	52	57.1	54.8	52.9	50.7	48.3

FIGURE 5.7 – Résultats obtenus pour différentes réductions de dimension par ACP post-SV

Ces résultats sont très intéressants, car ils montrent qu'on peut réduire très fortement la dimension des vecteurs signature sans perdre l'information qu'ils contiennent. En particulier, conserver 30% de la variance nous permet de passer d'un vecteur de taille 5120 à 52 tout en conservant voire améliorant les résultats.

Chapitre 6

Approche VLAD

6.1 Présentation de l'approche

A l'interface entre GMM-SV et VQ se trouve la technique Vector Agregation of Local Descriptors (VLAD). Cette technique, appliquée uniquement à l'image dans la littérature, peut se transposer à l'audio. Comme nous allons le voir, elle possède un fonctionnement très proche de GMM-SV mais peut permettre une amélioration en terme de temps de calcul et de résultats, et possède quelques points particuliers.

6.1.1 Création d'un dictionnaire

La première étape de VLAD, tout comme VQ (et GMM-SV dans un certain sens), est de créer un dictionnaire qui permettra de représenter au mieux les descripteurs pouvant exister dans la musique. Comme pour VQ, l'algorithme K-moyennes est sûrement la solution la plus évidente.

Le fait que nous disposions d'un algorithme EM (qui permet la création du modèle du monde dans l'approche GMM-SV), et l'envie de comparaison directe avec GMM-SV nous poussent à tester cette méthode de création de dictionnaire. Il suffira en effet de ne sélectionner que les centroïdes de l'UBM pour obtenir le dictionnaire qui servira dans VLAD.

6.1.2 Calculs des résiduels

Tandis que l'étape suivant la création d'un modèle du monde dans GMM-SV est l'adaption MAP pour chaque morceau, la seconde étape de VLAD consiste à associer chaque descripteur à l'élément le plus proche du dictionnaire, et à calculer la distance entre ces deux points. On somme ensuite pour chaque centroïde les distances entre les points qui lui sont associés et lui même, ce qui donne pour chaque centroïde un vecteur résiduel, et on concatène tous les vecteurs résiduels pour former le vecteur signature.

On dispose donc d'une fonction de quantification, qui associe chaque point à l'élément le plus proche dans le dictionnaire :

$$q : \mathbf{R}^d \to C \subset \mathbf{R}^d$$
$$x \mapsto q(x) = \arg\min_{\mu \in C} ||x - \mu||$$

avec $||.||_2$ la norme L_2.
Puis on calcule les résiduels de chaque centroïde. Ainsi, pour le centroïde i :

$$v^i = \sum_{x:q(x)=\mu_i} x - \mu_i \tag{6.1}$$

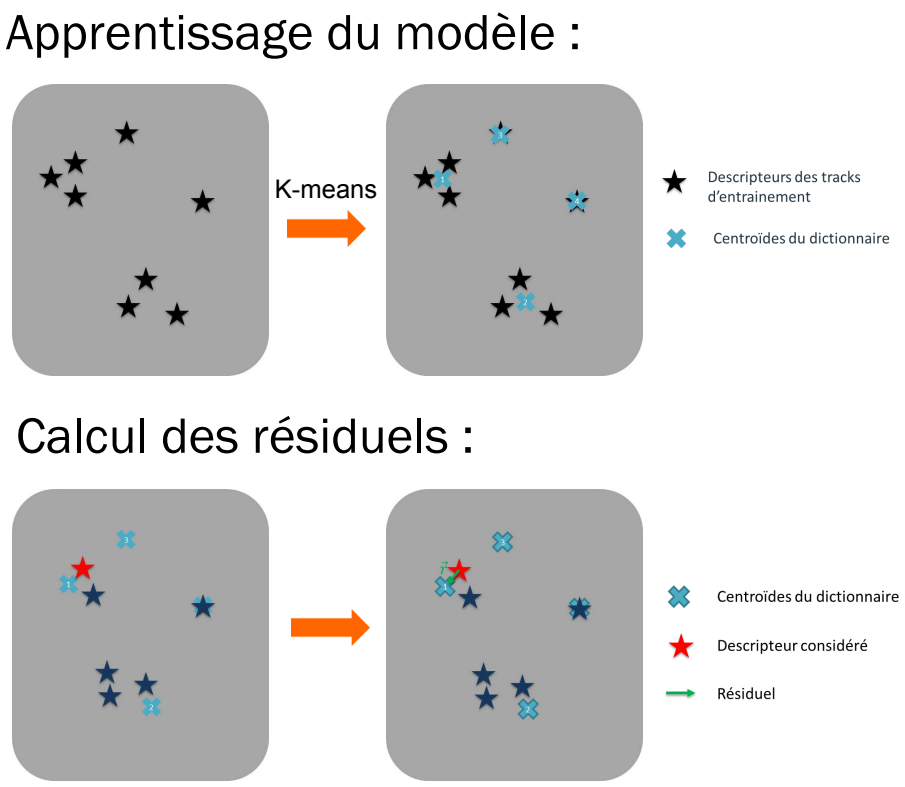


FIGURE 6.1 – Schéma explicatif Vlad

Enfin, on concatène ces résiduels pour former le vecteur signature :

$$v = [v^1 ... v^k] \tag{6.2}$$

Ce vecteur signature est donc de dimension $k \times d$, avec k la taille du dictionnaire et d la dimension de l'espace des descripteurs.

Il est intéressant de noter, que, contrairement à l'approche VQ, l'agrégation des descripteurs incorpore une contrainte sur la position du descripteur : en effet, seuls les descripteurs appartenant au même cluster sont agrégés ensemble.

6.1.3 Normalisation Puissance

Afin de réduire l'influence trop forte des éléments répétitifs, et notamment des textures en image, une norme nommée normalisation Puissance est proposée. Sa transposition à l'audio est questionnable, mais les expériences ont prouvées qu'un réglage approprié permettait effectivement une amélioration de la recherche par similarité.

La normalisation Puissance consiste à appliquer sur chaque composante du vecteur signature une opération non linéaire, c'est à dire élever la composante à une puissance α avec $\alpha \in [0; 1]$, tout en conservant son signe :

$$v_j = sign(v_j) \times |v_j|^{\alpha} \tag{6.3}$$

Mes expériences montrent que le choix de $\alpha = 0.2$ permet d'améliorer de manière importante les résultats.

6.2 Améliorations apportées

A nouveau, on applique l'agrégation temporelle préalable et le blanchiment par ZCA. On applique aussi la norme UCS. Le système complet est donc décrit figure 6.2.

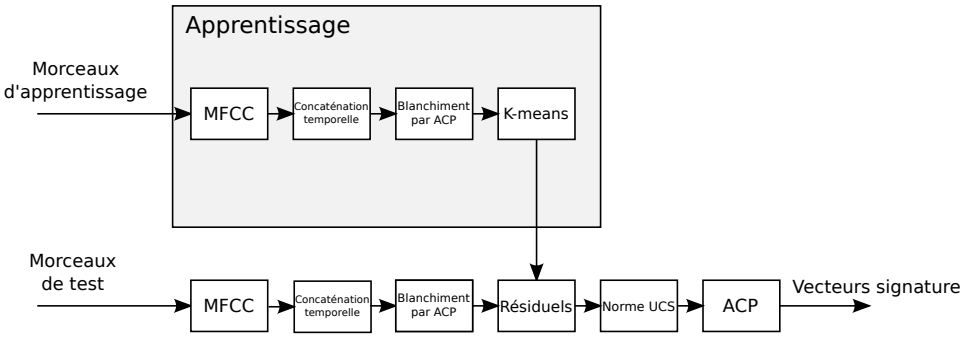


FIGURE 6.2 – Organisation du système final de calcul des vecteurs signature par VLAD

6.3 Résultats obtenus

6.3.1 Influence du retrait de la moyenne

* Retrait de la moyenne par composantes et par morceau
* Vlad 128

Retrait de la moyenne	Scores				
	1	3	5	10	20
Non	43.3	41.1	39.4	37.4	34.9
Oui	48.4	46.2	44.8	42.4	40.0

FIGURE 6.3 – Résultats obtenus avec ou sans retrait de la moyenne

Tout comme GMM-SV, Vlad profite nettement du retrait de la moyenne sur chaque composante et sur chaque morceau.

6.3.2 Influence de la concaténation temporelle

* Retrait de la moyenne par composantes et par morceau
* Concaténation temporelle 6 trames (dimension 6*17 =102)
* Blanchiment par PCA -> dimension 40
* Vlad 128

Concaténation + PCAw	Scores				
	1	3	5	10	20
Non	48.4	46.2	44.8	42.4	40.0
Oui	58.6	54.6	53.2	50.8	48.0

FIGURE 6.4 – Résultats obtenus avec ou sans concaténation temporelle + Blanchiment par ACP

Ce résultat est particulièrement intéressant : alors que les résultats de Vlad sont assez largement en dessous de GMM-SV sans concaténation temporelle, Vlad devient l'approche la plus intéressante lorsqu'on utilise la concaténation temporelle. Une explication pourrait être que Vlad fonctionne mieux dans de grands espaces (sans concaténation temporelle : espace de dimension 17, avec : espace de dimension 40).

6.3.3 Influence du nombre de centroides

* Retrait de la moyenne par composantes et par morceau
* Concaténation temporelle 6 trames (dimension 6*17 =102)
* Blanchiment par PCA -> dimension 40
* Vlad

Nombre de centroïdes	Scores				
	1	3	5	10	20
64	55.2	52.2	50.6	47.9	44.8
128	58.6	54.5	53.2	50.8	48.0
256	59.4	55.4	53.8	51.6	48.4

FIGURE 6.5 – Résultats obtenus pour différents nombres de centroides

L'influence du nombre de centroides est celle attendue, avec une amélioration des résultats qui suit l'augmentation du nombre de centroides.

6.3.4 Influence de la norme puissance

* Retrait de la moyenne par composantes et par morceau
* Concaténation temporelle 6 trames (dimension 6*17 =102)
* Blanchiment par PCA -> dimension 40
* Vlad 128

Puissance utilisée	Scores				
	1	3	5	10	20
0.1	59.2	56.0	53.9	51.2	47.9
0.2	60.2	56.0	54.4	51.9	48.4
0.5	58.6	54.5	53.2	50.8	47.9
1	50.1	47.0	45.5	43.7	41.7

FIGURE 6.6 – Résultats obtenus pour différentes puissances dans la norme Puissance

La norme puissance permet de réduire l'influence des poids importants par rapport aux poids faibles. Cette norme a tendance à s'opposer à la parcimonie. Etant donné que Vlad est une somme de résiduels, si un centroïde n'a dans son entourage que des points lointains, il risque d'avoir une grande importance qui peut ne pas être justifiée par rapport à son intérêt dans la représentation.

6.3.5 Influence de l'ACP post-Vlad

* Retrait de la moyenne par composantes et par morceau
* Concaténation temporelle 6 trames (dimension 6*17 =102)
* Blanchiment par PCA -> dimension 40
* Vlad 128
* PCA

Variance conservée	Dimension de sortie	Scores				
		1	3	5	10	20
1	5120	58.6	54.5	53.2	50.8	47.9
0.9	1824	58.7	55.3	53.3	51.2	48.3
0.5	361	58.3	55.4	53.8	51.4	48.4
0.3	65	58.6	55.7	53.9	51.3	48.7

FIGURE 6.7 – Résultats obtenus pour différentes réductions de dimension par ACP post-Vlad

De manière assez intéressante, la réduction de dimension par ACP couplée à la méthode Vlad améliore les résultats. On peut penser que l'ACP supprime la partie commune à tous les morceaux, et ne garde que les composantes qui différencient vraiment les morceaux entre eux. Ainsi, la mesure de distance entre vecteurs signature est moins perturbée par cette composante commune à tous les morceaux.

Conclusion

Lors de mon stage, j'ai pu étudier différentes techniques de modélisation vectorielle pour des distributions de descripteurs, et j'ai montré que des techniques issues de l'image (VQ et Vlad notamment) pouvaient concurrencer l'approche état de l'art en audio (GMM-SV). Il semble notamment que l'approche Vlad soit mieux adaptée à des espaces de grande dimension que GMM-SV. D'autre part, l'algorithme K-means étant moins coûteux que l'algorithme EM, et le calcul des résiduels Vlad étant plus aisé que le calcul de l'adaptation dans GMM-SV, ce changement nous permet d'économiser beaucoup de temps dans le calcul des vecteurs signature.

D'autre part, et comme on pouvait en partie le supposer, j'ai pu observer l'importance du caractère temporel pour la similarité et la classification musicales. L'approche mêlant agrégation temporelle puis blanchiment par ACP a contribué à améliorer fortement les résultats de toutes les approches, car elle permet de prendre en compte la dimension temporelle qui existe entre les trames.

Enfin, la réduction de dimension post-représentation vectorielle par ACP permet d'améliorer encore un peu plus les résultats en éliminant la partie bruitée de la représentation vectorielle de chaque morceau. Cette ACP est particulièrement efficace avec la méthode Vlad.

L'ouverture naturelle dans laquelle j'ai choisi de poursuivre est l'apprentissage automatique de représentations afin de remplacer les MFCC. Ces MFCC semblent en effet devenir le point faible de notre système, et les résultats état de l'art obtenus dans les autres domaines par l'apprentissage profond nous donnent de grandes espérances quant à cette direction de recherche.

Système initial :
* SV 128
Meilleure configuration obtenue :
* Retrait de la moyenne par composantes et par morceau
* Concaténation temporelle 6 trames (dimension 6*17 =102)
* Blanchiment par PCA -> dimension 40
* Vlad 128, norme puissance à 0.2
* PCA avec 50% de la variance conservée (dimension : 505)

Système	Scores				
	1	3	5	10	20
Initial	48.2	46.7	45.8	43.7	40.9
Nouveau	60.7	56.7	54.8	52.0	48.9

FIGURE 6.8 – Comparatif Système initial/Nouveau système

Annexe A

Analyse en Composantes Principales et Blanchiment par ACP

L'Analyse en Composantes Principales consiste à déterminer les axes de variance maximale d'un nuage de points, et à décorréler les données. Son but est de déterminer les k dimensions telles que la variance sur ces axes soit maximale.

On suppose qu'on dispose d'une matrice X contenant n échantillons dans un espace de dimension d. La dimension de la matrice X est donc $n \times d$ et les données sont supposées centrées. On calcule sa matrice de covariance :

$$C = X^T X \tag{A.1}$$

On cherche à rendre la matrice de covariance diagonale (afin de décorréler les données), on détermine alors les vecteurs propres et les valeurs propres de cette matrice de covariance.

$$C = V^T \Delta V \tag{A.2}$$

avec $\Delta = Diag(\lambda_1, ..., \lambda_n)$ où les λ correspondent aux valeurs propres de la matrice de covariance, rangées en ordre décroissant. La matrice V correspond aux vecteurs propres de la matrice de covariance. Ces vecteurs propres sont nommés axes principaux.

On cherche le sous espace de dimension k tel que la variance sur ce sous espace soit maximum. On garde donc la projection sur les k premiers vecteurs propres, car ce sont ceux qui possèdent la plus grande variance.

Ainsi,

$$Y = XV_k \tag{A.3}$$

avec V_k la matrice carrée de dimension $k \times k$ qui correspond aux k premières lignes et k premières colonnes de la matrice V.

Si on souhaite que les données possèdent une variance unitaire sur chacun des nouveaux axes, on peut blanchir les données, c'est à dire choisir

$$Y = \Delta_k^{-1} X V_k \tag{A.4}$$

où Δ_k^{-1} correspond à la matrice $\Delta_k^{-1} = Diag(\frac{1}{\lambda_1}, ..., \frac{1}{\lambda_k})$. On a alors effectué un blanchiment par ACP.

Annexe B

Blanchiment par Zero Component Analysis

On dispose d'une matrice X de dimension $n \times d$. On suppose que cette matrice est de moyenne nulle, sa covariance est alors donnée par :

$$\frac{1}{n-1} X X^T \tag{B.1}$$

On cherche à décorréler les différentes dimensions. On peut le faire grâce à une transformation linéaire W, qui transforme la matrice X en une matrice Y décorrélée :

$$Y = WX \tag{B.2}$$

Pour que W soit une matrice de décorrélation, et afin d'imposer une variance identique pour toutes les dimensions, la matrice de covariance de Y doit être égale à la matrice identité. Ainsi :

$$\begin{aligned}
YY^T &= (n-1)I \\
WXX^TW^T &= (n-1)I \\
W^TWXX^TW^T &= (n-1)W^T \\
W^2XX^TW^T &= (n-1)W^T \\
W^2XX^T &= (n-1)I \\
W^2 &= (n-1)(XX^T)^{-1} \\
W &= \sqrt{n-1}(XX^T)^{-\frac{1}{2}}
\end{aligned}$$

On trouve facilement $(XX^T)^{-\frac{1}{2}}$ car XX^T est symétrique et donc diagonalisable. Ainsi :

$$
\begin{aligned}
XX^T &= PDP^T \\
(XX^T)^{-\frac{1}{2}} &= ((XX^T)^{-1})^{\frac{1}{2}} \\
&= ((PDP^T)^{-1})^{\frac{1}{2}} \\
&= (PD^{-1}P^T)^{\frac{1}{2}} \\
&= PD^{-\frac{1}{2}}P^T
\end{aligned}
$$

avec $D^{-\frac{1}{2}}$ la matrice D avec les éléments pris à la puissance $-\frac{1}{2}$.

La matrice W transforme alors X pour que la matrice Y résultante possèdent des dimensions décorrélées et de variance unitaire. On peut voir le blanchiment par ZCA comme une rotation dans l'espace des composantes principales, une division de chacune des dimensions par son écart type et une rotation inverse dans l'espace de départ.

Notons que le blanchiment par ZCA diffère du blanchiment par ACP par l'ajout de la rotation inverse dans l'espace de départ.

Annexe C

Ensemble des résultats

C.1 VQ

* Retrait de la moyenne par composantes et par morceau
* Concaténation temporelle 6 trames (dimension 6*17 =102)
* Blanchiment par PCA -> dimension 40
* VQ avec codage dur

Nombre de centroïdes	Scores				
	1	3	5	10	20
512	51.9	49.8	48.6	46.9	44.8
1024	56.1	53.6	52.0	50.3	48.0
2048	51.5	49.0	48.0	46.2	44.1

FIGURE C.1 – Résultats obtenus pour différents nombres de centroides, codage dur

* Retrait de la moyenne par composantes et par morceau
* Concaténation temporelle 6 trames (dimension 6*17 =102)
* Blanchiment par PCA -> dimension 40
* VQ avec codage doux

Nombre de centroïdes	Scores				
	1	3	5	10	20
512	55.2	52.2	50.6	47.9	44.8
1024	52.8	50.1	49.1	46.9	44.4
2048	53.4	50.8	49.4	47.5	44.9

FIGURE C.2 – Résultats obtenus pour différents nombres de centroides, codage doux

* Retrait de la moyenne par composantes et par morceau
* VQ 2048 avec codage dur sur le plus proche voisin

Retrait de la moyenne	Scores				
	1	3	5	10	20
Non	46.3	43.8	42.4	40.1	37.2
Oui	46.6	43.8	42.6	40.9	38.6

FIGURE C.3 – Résultats obtenus avec ou sans retrait de la moyenne avec VQ codage dur

* Retrait de la moyenne par composantes et par morceau
* VQ 2048 avec codage doux

Retrait de la moyenne	Scores				
	1	3	5	10	20
Non	47.1	43.2	41.8	39.7	37.3
Oui	45.6	42.9	41.5	39.6	37.1

FIGURE C.4 – Résultats obtenus avec ou sans retrait de la moyenne avec VQ codage doux

* Retrait de la moyenne par composantes et par morceau
* Concaténation temporelle 6 trames (dimension 6*17 =102)
* Blanchiment par PCA -> dimension 40
* VQ 2048 avec codage dur

Concaténation + PCAw	Scores				
	1	3	5	10	20
Non	46.6	43.8	42.6	40.9	38.6
Oui	51.5	49.0	48.0	46.2	44.1

FIGURE C.5 – Résultats obtenus avec ou sans concaténation temporelle avec VQ codage dur

* Retrait de la moyenne par composantes et par morceau
* Concaténation temporelle 6 trames (dimension 6*17 =102)
* Blanchiment par PCA -> dimension 40
* VQ 2048 avec codage doux

Concaténation + PCAw	Scores				
	1	3	5	10	20
Non	45.6	42.9	41.5	39.6	37.1
Oui	53.4	50.8	49.4	47.5	44.9

FIGURE C.6 – Résultats obtenus avec ou sans concaténation temporelle avec VQ codage doux

* Retrait de la moyenne par composantes et par morceau
* Concaténation temporelle 6 trames (dimension 6*17 =102)
* Blanchiment par PCA -> dimension 40
* VQ 2048, codage dur
* PCA

Variance conservée	Dimension de sortie	Scores				
		1	3	5	10	20
1	2048	51.5	49.0	48.0	46.2	44.1
0.9	1003	51.9	49.6	48.2	46.7	44.8
0.5	247	52.8	51.3	50.0	48.3	46.2
0.3	132	53.9	51.3	50.4	48.3	46.6

FIGURE C.7 – Résultats obtenus pour différentes réductions de dimension par ACP post-VQ avec codage dur

* Retrait de la moyenne par composantes et par morceau
* Concaténation temporelle 6 trames (dimension 6*17 =102)
* Blanchiment par PCA -> dimension 40
* VQ 2048, codage doux
* PCA

Variance conservée	Dimension de sortie	Scores				
		1	3	5	10	20
1	2048	53.4	50.8	49.4	47.5	44.9
0.9	1051	53.5	51.5	50.3	48.8	45.4
0.5	70	53.4	50.9	49.8	47.5	44.9
0.3	12	36.4	36.5	35.7	35.6	34.7

FIGURE C.8 – Résultats obtenus pour différentes réductions de dimension par ACP post-VQ avec codage doux

C.2 GMM-SV

* Retrait de la moyenne par composantes et par morceau
* SV 128

Retrait de la moyenne	Scores				
	1	3	5	10	20
Non	48.2	46.7	45.8	43.7	40.9
Oui	53.7	49.5	48.0	45.9	43.4

FIGURE C.9 – Résultats obtenus avec ou sans retrait de la moyenne

* Retrait de la moyenne par composantes et par morceau
* Concaténation temporelle 6 trames (dimension 6*17 =102)
* Blanchiment par PCA -> dimension 40
* GMM-SV 128

Concaténation + PCAw	Scores				
	1	3	5	10	20
Non	53.7	49.5	48.0	45.9	43.4
Oui	57.7	54.2	52.4	50.0	47.0

FIGURE C.10 – Résultats obtenus avec ou sans concaténation temporelle + Blanchiment par ACP

* Retrait de la moyenne par composantes et par morceau
* Concaténation temporelle 6 trames (dimension 6*17 =102)
* Blanchiment par PCA -> dimension 40
* GMM-SV

Nombre de centroïdes	Scores				
	1	3	5	10	20
64	56.5	54.1	52.6	49.7	46.9
128	57.7	54.2	52.4	50.0	47.0
256	58.2	55.2	53.4	51.0	48.0

FIGURE C.11 – Résultats obtenus pour différents nombres de centroides

* Retrait de la moyenne par composantes et par morceau
* Concaténation temporelle 6 trames (dimension 6*17 =102)
* Blanchiment par PCA -> dimension 40
* GMM-SV 128
* PCA

Variance conservée	Dimension de sortie	Scores				
		1	3	5	10	20
1	5120	57.7	54.2	52.4	50.0	47.0
0.9	1810	57.2	54.2	52.5	50.1	47.5
0.5	332	56.9	54.3	53.0	50.5	47.9
0.3	52	57.1	54.8	52.9	50.7	48.3

FIGURE C.12 – Résultats obtenus pour différentes réductions de dimension par ACP post-SV

C.3 Vlad

* Retrait de la moyenne par composantes et par morceau
* Vlad 128

Retrait de la moyenne	Scores				
	1	3	5	10	20
Non	43.3	41.1	39.4	37.4	34.9
Oui	48.4	46.2	44.8	42.4	40.0

FIGURE C.13 – Résultats obtenus avec ou sans retrait de la moyenne

* Retrait de la moyenne par composantes et par morceau
* Concaténation temporelle 6 trames (dimension 6*17 =102)
* Blanchiment par PCA -> dimension 40
* Vlad 128

Concaténation + PCAw	Scores				
	1	3	5	10	20
Non	48.4	46.2	44.8	42.4	40.0
Oui	58.6	54.6	53.2	50.8	48.0

FIGURE C.14 – Résultats obtenus avec ou sans concaténation temporelle + Blanchiment par ACP

* Retrait de la moyenne par composantes et par morceau
* Concaténation temporelle 6 trames (dimension 6*17 =102)
* Blanchiment par PCA -> dimension 40
* Vlad

Nombre de centroïdes	Scores				
	1	3	5	10	20
64	55.2	52.2	50.6	47.9	44.8
128	58.6	54.5	53.2	50.8	48.0
256	59.4	55.4	53.8	51.6	48.4

FIGURE C.15 – Résultats obtenus pour différents nombres de centroides

* Retrait de la moyenne par composantes et par morceau
* Concaténation temporelle 6 trames (dimension 6*17 =102)
* Blanchiment par PCA -> dimension 40
* Vlad 128

Puissance utilisée	Scores				
	1	3	5	10	20
0.1	59.2	56.0	53.9	51.2	47.9
0.2	60.2	56.0	54.4	51.9	48.4
0.5	58.6	54.5	53.2	50.8	47.9
1	50.1	47.0	45.5	43.7	41.7

FIGURE C.16 – Résultats obtenus pour différentes puissances dans la norme Puissance

* Retrait de la moyenne par composantes et par morceau
* Concaténation temporelle 6 trames (dimension 6*17 =102)
* Blanchiment par PCA -> dimension 40
* Vlad 128
* PCA

Variance conservée	Dimension de sortie	Scores				
		1	3	5	10	20
1	5120	58.6	54.5	53.2	50.8	47.9
0.9	1824	58.7	55.3	53.3	51.2	48.3
0.5	361	58.3	55.4	53.8	51.4	48.4
0.3	65	58.6	55.7	53.9	51.3	48.7

FIGURE C.17 – Résultats obtenus pour différentes réductions de dimension par ACP post-Vlad

C.4 Nouveau système

Système initial :
* SV 128
Meilleure configuration obtenue :
* Retrait de la moyenne par composantes et par morceau
* Concaténation temporelle 6 trames (dimension 6*17 =102)
* Blanchiment par PCA -> dimension 40
* Vlad 128, norme puissance à 0.2
* PCA avec 50% de la variance conservée (dimension : 505)

Système	Scores				
	1	3	5	10	20
Initial	48.2	46.7	45.8	43.7	40.9
Nouveau	60.7	56.7	54.8	52.0	48.9

FIGURE C.18 – Comparatif Système initial/Nouveau système

Bibliographie

[1] Relja Arandjelovic and Andrew Zisserman. All about vlad. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 1578–1585. IEEE, 2013.

[2] Sandra Avila, Nicolas Thome, Matthieu Cord, Eduardo Valle, and A de A Araujo. Bossa : Extended bow formalism for image classification. In *Image Processing (ICIP), 2011 18th IEEE International Conference on*, pages 2909–2912. IEEE, 2011.

[3] Emmanouil Benetos, Margarita Kotti, and Constantine Kotropoulos. Musical instrument classification using non-negative matrix factorization algorithms. In *Circuits and Systems, 2006. ISCAS 2006. Proceedings. 2006 IEEE International Symposium on*, pages 4–pp. IEEE, 2006.

[4] Christophe Charbuillet, Damien Tardieu, and Geoffroy Peeters. Gmm supervector for content based music similarity. In *International Conference on Digital Audio Effects, Paris, France*, pages 425–428, 2011.

[5] Adam Coates and Andrew Y Ng. The importance of encoding versus training with sparse coding and vector quantization. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 921–928, 2011.

[6] Adam Coates and Andrew Y Ng. Learning feature representations with k-means. In *Neural Networks : Tricks of the Trade*, pages 561–580. Springer, 2012.

[7] Adam Coates, Andrew Y Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *International Conference on Artificial Intelligence and Statistics*, pages 215–223, 2011.

[8] Jonathan Delhumeau, Philippe-Henri Gosselin, Hervé Jégou, and Patrick Pérez. Revisiting the vlad image representation. In *ACM Multimedia*, Barcelona, Spain, October 2013.

[9] Sander Dieleman, Philémon Brakel, and Benjamin Schrauwen. Audio-based music classification with a pretrained convolutional network. In *12th International Society for Music Information Retrieval Conference (ISMIR-2011)*, pages 669–674. University of Miami, 2011.

[10] Zhouyu Fu, Guojun Lu, Kai Ming Ting, and Dengsheng Zhang. Music classification via the bag-of-features approach. *Pattern Recognition Letters*, 32(14) :1768–1777, 2011.

[11] Philippe Hamel and Douglas Eck. Learning features from music audio with deep belief networks. In *ISMIR*, pages 339–344. Utrecht, The Netherlands, 2010.

[12] Philippe Hamel, Simon Lemieux, Yoshua Bengio, and Douglas Eck. Temporal pooling and multiscale learning for automatic annotation and ranking of music audio. In *ISMIR*, pages 729–734, 2011.

[13] Eric J Humphrey, Juan P Bello, and Yann LeCun. Feature learning and deep architectures : new directions for music informatics. *Journal of Intelligent Information Systems*, 41(3) :461–481, 2013.

[14] Eric J Humphrey, Juan Pablo Bello, and Yann LeCun. Moving beyond feature design : Deep architectures and automatic feature learning in music informatics. In *ISMIR*, pages 403–408, 2012.

[15] Thomas Kelly. Music artist identification using linear temporal pyramid matching.

[16] Honglak Lee, Peter T Pham, Yan Largman, and Andrew Y Ng. Unsupervised feature learning for audio classification using convolutional deep belief networks. In *NIPS*, volume 9, pages 1096–1104, 2009.

[17] Mark Levy and Mark Sandler. Lightweight measures for timbral similarity of musical audio. In *Proceedings of the 1st ACM workshop on Audio and music computing multimedia*, pages 27–36. ACM, 2006.

[18] Thomas Lidy and Andreas Rauber. Evaluation of feature extractors and psycho-acoustic transformations for music genre classification. In *ISMIR*, pages 34–41, 2005.

[19] Brian McFee, Luke Barrington, and Gert Lanckriet. Learning content similarity for music recommendation. *Audio, Speech, and Language Processing, IEEE Transactions on*, 20(8) :2207–2218, 2012.

[20] Juhan Nam, Jorge Herrera, Malcolm Slaney, and Julius Smith. Learning sparse feature representations for music annotation and retrieval. 2013.

[21] Jan Schluter and Christian Osendorfer. Music similarity estimation with the mean-covariance restricted boltzmann machine. In *Machine Learning and Applications and Workshops (ICMLA), 2011 10th International Conference on*, volume 2, pages 118–123. IEEE, 2011.

[22] Yonatan Vaizman, Brian McFee, and Gert Lanckriet. Codebook based audio feature representation for music information retrieval. 2013.

[23] Chin-Chia Michael Yeh, Li Su, and Yi-Hsuan Yang. Dual-layer bag-of-frames model for music genre classification. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 246–250. IEEE, 2013.

[24] Syed Zubair, Fei Yan, and Wenwu Wang. Dictionary learning based sparse coefficients for audio classification with max and average pooling. *Digital Signal Processing*, 23(3) :960 – 970, 2013.