Université de Montréal

Apprentissage statistique pour l'étiquetage de musique et la recommandation

par
Thierry Bertin-Mahieux

Département d'informatique et de recherche opérationnelle
Faculté des arts et des sciences

Mémoire présenté à la Faculté des études supérieures
en vue de l'obtention du grade de Maître ès sciences (M.Sc.)
en informatique

août, 2009

Université de Montréal
Faculté des études supérieures

Ce mémoire intitulé:

Apprentissage statistique pour l'étiquetage de musique et la recommandation

présenté par:

Thierry Bertin-Mahieux

a été évalué par un jury composé des personnes suivantes:

Yoshua Bengio
président-rapporteur

Douglas Eck
directeur de recherche

Balazs Kégl
codirecteur

Pascal Vincent
membre du jury

Mémoire accepté le

RÉSUMÉ

Nous appliquons des techniques d'apprentissage statistique à des *features* audio afin d'étiqueter automatiquement de la musique. Nous faisons cet apprentissage à partir d'une grande collection de musique et d'étiquettes disponibles en ligne, principalement sur le site `www.lastfm.com`.

Nous utilisons l'algorithme de boosting sur des *decision stumps* afin d'apprendre une fonction de classification. Cet algorithme a l'avantage d'avoir une complexité linéaire en fonction du nombre d'étiquettes et du nombre de chansons, en plus d'avoir fait ses preuves sur des données audio.

Nous montrons le mérite de notre méthode en utilisant les étiquettes automatiques pour mesurer la similarité entre chansons. Nous commençons par créer une mesure de similarité de référence à partir des données sur les utilisateurs de `www.lastfm.com`. Nous essayons ensuite de reproduire cette similarité à partir des tags appliqués aux chansons par des utilisateurs, puis à partir des tags appliqués automatiquement. Nous concluons que, bien que la qualité de ces étiquettes automatiques soit inférieure, elles peuvent compléter les "vrais" étiquettes dans le cas de chansons peu populaires.

Mots clés: machine learning, traitement de signal, étiquetage automatique de musique, similarité musicale.

ABSTRACT

We apply machine learning techniques to audio features in order to automatically tagging music. We learn from a large collection of music and tags available online, mostly from the website `www.lastfm.com`.

We use the boosting algorithm on decision stumps to learn a classification function. An advantage of this algorithm is to have a linear complexity in the number of tags and in the number of songs, and it also proved to be efficient on audio data.

We show that our method has merit by using our automaticlly applied tags for meauring similarity among songs. We start by computing a reference similarity measure based on the user data on `www.lastfm.com`. We then try to reproduce that similarity from the tags applied to songs by user, than from the automatic tags. We conclude that the quality of the latter is inferior, but that they can still complete the "real" tags for unpopular songs.

Keywords:machine learning, signal processing, automatic tagging of music, music similarity.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES SIGLES

AIC	Akaike Information Criterion
AMG	All Music Guide
CMU	Carnegie Mellon University
DFT	Transformée de Fourier discrète
MIREX	Music Information Retrieval Evaluation Exchange
MFCC	Mel-scale Frequency Cepstral Coefficients
MRT	Moteur de recherche pour le texte
PCM	Pulse Code Modulation
SITM	Search Inside the Music
TF $\times$ IDF	Term frequency-inverse term frequency
UCSD	University of California in San Diego

NOTATION

$\|\|A\|\|$	norme euclidienne de A
I_A	vaut 1 si A vrai, 0 sinon
Hz	Hertz

"All models are wrong, but some are useful"

George E. P. Box

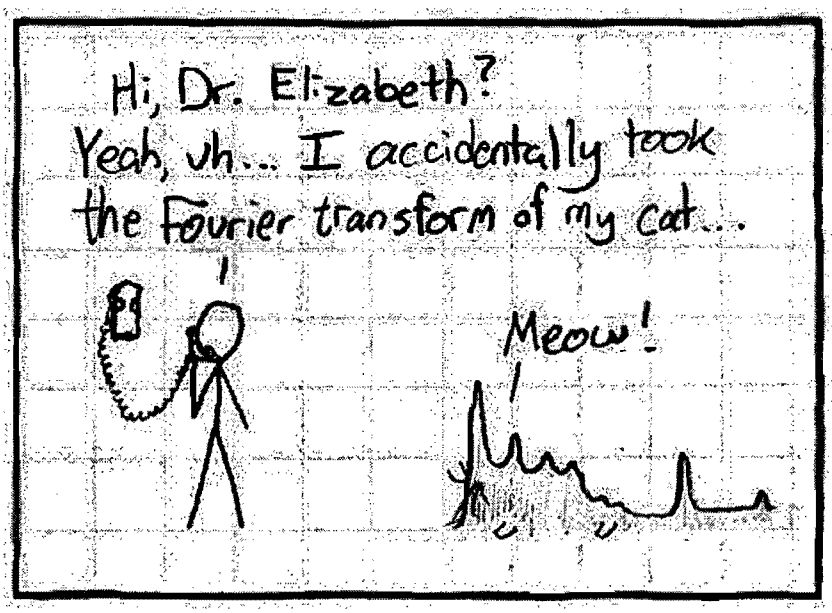


FIG. 1 – xkcd : http ://xkcd.com/26/

REMERCIEMENTS

Merci tout d'abord à mon directeur de recherche, Douglas Eck, ainsi qu'aux autres professeurs du laboratoire LISA : Yoshua Bengio, Balaźs Kégl et Pascal Vincent.

Merci à mes nombreux collègues du LISA, en particulier J. Bergstra, A. Lacoste, P. Lamblin, S. Lauly et S. Wood.

Merci à Anne Broadbent pour son modèle Latex de thèse.

Merci à Paul Lamere, pour m'avoir donné ma chance.

A PhD comics, lapin.org, Family Guy, The Simpsons, Friends, Hell's Kitchen et Lance et Compte.

A mes parents, pour leur soutien inconditionnel.

A Joannie. A Cristine. Pour m'avoir enduré.

AVANT-PROPOS

Le travail mené au laboratoire GAMME se résume, selon moi, à un concept : comprendre la structure de la musique.

Est-ce que par là on entend faire un travail de musicologue à grande échelle, et appliquer des théories musicales à des milliers de chansons grâce à la rapidité de l'informatique ? Pas nécessairement.

Est-ce que, dans ce cas, nous espérons augmenter la connaissance musicale par de nouvelles théories ? Peut-être.

Mais alors, que faisons-nous pour comprendre la structure de la musique ? Nous l'exploitons à différentes fins.

Et ce faisant, nous faisons un important postulat : il y a une structure à la musique. C'est ce qui permet à un humain de l'apprécier, de différencier des oeuvres, de comprendre les émotions qu'elle transporte, de jouer avec la musique et de la varier à souhait, de s'en servir comme d'un langage.

Nous croyons que cette structure est cachée dans un complexe enrobage, mais nous croyons aussi qu'avec suffisamment d'exemples, cette structure apparaîtra au moyen de la statistique. Cela ne signifiera pas que l'on comprenne ce qui apparaîtra. Mais à chaque fois qu'un algorithme sera capable d'exploiter la structure de la musique, nous nous en rapprocherons.

Et, en nous rapprochant de cette structure, ne nous nous rapprochons-nous pas de la seule chose connue qui la comprenne, L'Humain et sa pensée ? Soyons fous, et croyons-y le temps de ce mémoire. Il ne résoud rien d'aussi grandiose que la pensée humaine, mais la motivation demeure la même.

CHAPITRE 1

INTRODUCTION

L'ère du Web 2.0 a révolutionné la manière dont nous utilisons les médias (vidéos, musique, documents écrits, etc). Il serait redondant de vouloir décrire et expliquer ce que chacun vit dans son quotidien sur Internet : nous avons accès à une quantité astronomique d'information, et cette quantité ne cesse de croître. Nous nous concentrons ici sur l'un de ces médias, à savoir la musique. Ce média est intéressant pour plusieurs raisons :

- le nombre de chansons accessibles est quasi-infini ;
- la production de la musique n'est plus une exclusivité des grandes maisons de disques, un simple ordinateur peut suffir de nos jours à en produire ;
- l'intérêt que l'on porte à une chanson dépend non seulement de la chanson en elle-même, mais également de son contexte social.

Dans ce travail nous nous intéressons à la manière dont l'on gère et découvre la musique à l'ère du Web 2.0. Nous utilisons l'apprentissage statisque pour automatiquement étiqueter une chanson, simplement à partir du fichier audio. Les étiquettes (ou *tags*) peuvent servir à mieux organiser une collection personnelle, ainsi qu'à faire de la recommandation. Nous développons une méthode complète, de l'entraînement jusqu'à l'étiquetage et la recommandation, et nous comparons les étiquettes prédites avec celles appliquées par des humains.

1.1 Contribution

Ce projet s'inscrit dans une volonté d'améliorer la classification musicale et s'appuie sur les travaux de plusieurs collègues du laboratoire LISA[1]. En particulier, Bergstra et al. ont développé la reconnaissance de genres au moyen d'AdaBoost [BCE+06]. Ce travail est approfondi dans les mémoires de maîtrise de Bergstra [Ber06] et Casagrande [Cas05]. L'utilisation de *tags* est aussi un sujet d'intérêt du laboratoire, tel que le montre l'article de Bergstra et al. [BLE06].

[1] www.iro.umontreal.ca/~lisa

La majorité de ce travail a aussi été faite en collaboration avec Paul Lamere de Sun Microsystems, dans le cadre du projet *Search Inside the Music*[2] (SITM).

Au regard de ces travaux, la contribution de ce mémoire est double. Premièrement, il généralise l'utilisation d'AdaBoost aux cas des *tags* qui sont appris indépendamment. Nous travaillons avec un plus grand nombre de données et nous offrons une solution à ce problème.

Deuxièmement, nous présentons une méthode pour mesurer la performance d'un système d'étiquetage automatique de musique axé sur la similarité. Cette mesure contourne le problème du mauvais étiquetage des données de test tout en étant reproductible.

Finalement, la majorité des travaux présentés dans ce mémoire a été diffusée dans la communauté par les articles suivants (en ordre chronologique) : [EBML07, ELBMG08, BMEML08, KBME08].

1.2 Motivation

En premier lieu, notre motivation porte sur l'usage d'étiquettes. Prenons un exemple concret pour comprendre leur utilité. Disons que vous recherchez un texte à partir de quelques mots-clés. Vous allez sur Google[3], vous tapez *jazz* et *Miles Davis*, quelques dizaines de sites apparaissent avec leurs descriptions, et en quelques secondes vous savez si vous avez trouvé ce que vous cherchiez. Refaisons la même expérience avec de la musique. Vous avez accès à une immense base de données, vous cherchez en mettant les mots *jazz* et *Miles Davis*, et on vous donne accès à quelques dizaines de chansons que vous pouvez écouter, mais aucune information supplémentaire n'est affichée. Combien de temps allez vous passer à écouter toutes ces chansons pour savoir si il y en a une qui vous convient ?

La leçon à tirer est que nous sommes habitués à travailler avec du texte et plus efficaces. Nous scannons rapidement l'information, nous pouvons résumer à partir des éléments clés, et nous savons si l'on a ce que l'on cherchait. Faire cela sur de la musique est beaucoup moins évident. Il faut donc décrire cette musique au moyen de mots. La première intuition est d'utiliser le *metadata* disponible, par exemple le nom de l'artiste,

[2] `http://research.sun.com/projects/dashboard.php?id=153`
[3] `www.google.com`

le titre de la chanson, de l'album, l'année, etc. Une deuxième étape est l'utilisation de mots-clés qui décrivent directement la chanson. Ce sont ces étiquettes dont nous parlons ici. Si l'on repense à la mise en scène ci-dessus, choisir une chanson devient plus simple à l'aide de ces mots-clés. Non seulement on peut écouter une chanson recommandée, mais on peut y trouver accolé une description *Miles Davis - chanson bebop des années 40 avec trompette et tuba, énergique.* Nous savons beaucoup plus facilement si c'est le genre de musique qui nous intéresse.

Ensuite, notre motivation porte sur le *cold-start problem*, ou démarrage froid en traduction littérale. Cela désigne le cercle vicieux suivant : un nouveau produit qui apparaît dans un système de recommandation n'est pas recommandé, car personne ne le connaît encore. Puisqu'il n'est pas recommandé, personne ne le connaît. Ainsi, le produit peut facilement rester inconnu toute son existence simplement parce que le système ne sait pas quoi en faire en premier lieu. Les étiquettes peuvent remédier partiellement à ce problème dans le cas de la musique. Si une nouvelle chanson rentre dans un recommandeur de musique et que notre algorithme l'étiquette comme *jazz*, on peut sans aucun doute la faire écouter à quelques utilisateurs que l'on a identifié comme amateurs de jazz. Ainsi, on obtient rapidement du *feedback* des utilisateurs potentiellement intéressés par cette chanson. Actuellement, beaucoup de recommandeurs de musique en ligne ne font rien pour régler le *cold-start problem*, se disant que la chanson peut devenir populaire par d'autre moyens (télévision, radio, concerts, etc). En faisant cela, on ignore une partie grandissante de la culture musicale, celle qui se produit dans des studios maisons et qui se diffuse au moyen de l'internet.

1.3 Structure

En premier lieu nous ferons un survol des travaux précédents dans divers domaines (chapitre 2) : les genres musicaux, la physique du son, l'apprentissage statistique, la reconnaissance automatique de genre en musique et la similarité musicale.

Au chapitre suivant (chapitre 3), nous parlerons de différentes bases de données et sources d'information utiles à notre travail. En particulier, nous verrons comment l'on se peut se procurer des étiquettes appliquées par des humains.

Au chapitre 4, nous décrirons notre algorithme, étape par étape, du fichier audio

jusqu'à l'étiquetage prédit.

Par la suite, au chapite 5, nous mesurerons la performance de notre modèle en l'appliquant à la création d'une mesure de similarité entre artistes. Cette similarité est une composante essentielle d'un système de recommandation.

Finalement, nous discuterons des mesures d'évaluation utilisées dans ce travail et de différents travaux futurs qui pourraient nous permettre d'améliorer le modèle.

CHAPITRE 2

TRAVAUX PRÉCÉDENTS

Dans ce chapitre nous présentons différents travaux et concepts reliés à notre recherche. Tout d'abord, nous présentons les genres, les étiquettes et tout ce qu'on appelle meta-données (*metadata* en anglais). En particulier, nous discuterons de ce qui vient du contexte et de ce qui vient de l'audio en lui-même. Ensuite, nous ferons un rappel des méthodes classiques en traitement de signal, tout en mettant l'accent sur les *features* que nous utiliserons par la suite dans nos travaux. Dans la section 2.3, nous présenterons l'apprentissage statistique, en particulier les principaux classifieurs existants. Finalement, dans la section 2.5, nous expliquerons le concept de similarité musicale et présenterons différentes approches pour s'attaquer à ce problème.

2.1 Genres musicaux et étiquettes

2.1.1 Genres

Le concept le plus répandu nous servant à décrire la musique est certainement celui de *genre*. Cela dit, ce concept s'avère être très mal défini. Pachet et Cazary [PC] définissent une hiérarchie de genres basée sur l'instrumentation et des descripteurs historiques, qui se veut le plus objectif possible. D'autres tels que McKay et Fujinaga [MF05] ont voulu définir une taxonomie complète et consistante de genres, dans le but d'attribuer une catégorie précise à chaque chanson. Ces ensembles de genres peuvent être vus comme une extension des genres traditionnels tels que *classique*, *country*, *jazz*, *pop*, *rap*, *rock*. Suivant une structure développée dans Bergstra [Ber06], nous allons différencier les *genres traditionnels*, notamment développés par les maisons de disques, et les *genres collaboratifs*, développés par des communautés d'utilisateurs en ligne, où l'ensemble des genres possibles peut être considéré infini. Ce dernier type de genre sera développé dans la section suivante 2.1.2.

Un exemple d'ensemble de genres traditionnels est le standard ID3 utilisé par le format

MP3[1]. A l'origine, une chanson pouvait se faire attribuer l'un des 80 genres prédéfinis. Il n'y avait ni moyen d'appartenir à plusieurs genres, ni d'en définir de nouveaux. Un autre exemple est le service AllMusic Guide (AMG) qui sur sont site web[2] offre une hiérarchie à deux niveaux pour les genres. Au somment on retrouve *classical* et *popular*, puis en-dessous on retrouve respectivement des genres tels que *ballet*, *concerto*, *symphony*, et *blues*, *rap*, *latin*. De tels ensembles peuvent êtres utiles aux compagnies de disques qui essaie de décrire facilements leurs artistes (Perrot et Gjerdigen [PG99]). De la même façon on peut penser aux magasins de disque qui doivent organiser les oeuvres par catégories simples.

Ce systéme de classification comporte cependant de nombreux désavantages. Une analyse en est faite par Pachet et Cazaly [PC] et est résumé par Aucouturier et Pachet [AP03] :

- la plupart des taxonomies est pensée en fonction des albums et non des chansons elle-mêmes ;
- il n'y a pas de consensus, i.e. il n'existe pas un ensemble de genres traditionnels accepté de tous ;
- inconsistence sémantique : certains genres réfèrent à des périodes, d'autres à des sujets, des types de danses, etc ;
- ce qu'un genre particulier signifie n'est pas toujours clair. Le genre *world music* peut facilement changer d'un pays à un autre.

Il faut noter, qu'en théorie, ce système de genres traditionnels n'est pas une taxonomie, où chaque élément n'aurait qu'une seule et unique catégorie. Cela dit, c'est souvent le cas en pratique pour des raisons de commodité. Par exemple, le concours de reconnaissance de genre de MIREX [Mir] demande d'attribuer à chaque chanson le genre qui la représente le mieux (voir par exemple la soumission de Bergstra et al. [BCE05]). De la même façon, chaque enregistrement ne se retrouve qu'à un seul endroit chez un disquaire.

Tel que présenté plus haut, Pachet et Cazaly notait que le sens d'un genre n'est pas toujours bien défini et universel. Nous voulons rajouter à cela que le sens peut évoluer

[1]Liste des genres du standard ID3 : `www.id3.org/id3v2-00`
[2]AMG : `www.allmusic.com`

avec le temps, e.g. la définition de *rock* à l'époque d'Elvis Presley [3] n'englobait sans doute pas un groupe actuel comme Radiohead [4].

Comme nous le verrons dans la sous-section suivante, les étiquettes offrent une solution naturelle à ces problèmes. Une chanson peut se faire attribuer plusieurs étiquettes, comme si elle appartenait à plusieurs genres et leur sens est défini par la communauté (d'internautes généralement), il peut donc évoluer et s'adapter.

2.1.2 Etiquettes

En opposition avec les genres traditionnels décrits à la section 2.1.1, un nouvel ensemble de descripteurs est apparu avec le Web 2.0 dans les communautés d'internautes. Ceux-ci peuvent décrire différents produits au moyen d'étiquettes (ou *tags*) (Marlow et al. [MNBD06]). Par exemple, sur le site d'Amazon[5] on peut laisser des commentaires sur les livres, films ou musiques que l'on connaît. Ceux-ci sont visibles à l'ensemble des utilisateurs et peuvent servir à différentes choses.

Dans le cas de la musique, un excellent exemple est le site de Last.fm[6]. Last.fm est une radio personnalisée en ligne, ce service permet d'écouter de la musique en continu dont le contenu s'adapte aux goûts d'un utilisateur. Ceux-ci peuvent étiqueter (ou *tagger*) les artistes, albums ou chansons. Par exemple, le tableau 2.1 présente les tags les plus appliqués au groupe The Shins.

Last.fm décrit elle-même les tags comme des "mots-clés". Lamere [Lam08] résume différentes raisons pourquoi les gens les utilisent :

- mémoire et contexte : les tags servent à aider l'utilisateur dans sa recherche de musique. Par exemple, il peut étiqueter une partie de sa collection comme étant *techno*, ou *party* ;
- organisation : les tags peuvent aider à la découverte de nouvelles chansons. Par exemple, un utilisateur peut étiqueter une chanson *check out* pour se rappeler d'aller se renseigner sur l'artiste ;
- signalisation sociale : un utilisateur peut se servir de tags pour exprimer ses pré-

[3] Site web officiel : `www.elvis.com`
[4] Site web officiel : `www.radiohead.com`
[5] `www.amazon.com`
[6] `www.last.fm`

TAB. 2.1 – Tags pour *The Shins*

Tag	Freq	Tag	Freq	Tag	Freq
Indie	2375	The Shins	190	Punk	49
Indie rock	1138	Favorites	138	Chill	45
Indie pop	841	Emo	113	Singer-songwriter	41
Alternative	653	Mellow	85	Garden State	39
Rock	512	Folk	85	Favorite	37
Seen Live	298	Alternative rock	83	Electronic	36
Pop	231	Acoustic	54	Love	35

Top 21 des tags appliqués au groupe *The Shins*

férences aux autre utilisateurs. Par exemple, il peut utiliser le tag *seen live* pour dire qu'il a vu un artiste en concert ;

- contribution sociale : l'utilisateur aide le reste de la communauté en étiquetant correctement les chansons qu'il connaît bien ;
- Jeu et compétition : certains tags sont appliqués dans le cadre de jeux, voir section 3.7 ;
- expression d'une opinion : cela concerne les tags tels que *awesome* et *worst song ever*.

Malgré qu'il n'y ait aucune restriction sur les tags pouvant être utilisés, il semble qu'une communauté, i.e. un ensemble d'utilisateurs d'un même site, finissent par converger sur un ensemble de mots plus ou moins fini (Guy et Tonkin [GT]). Ces étiquettes peuvent faire référence à différents concepts. Le tableau 2.2 catégorise les tags de Last.fm et donne leur fréquence approximative.

Par rapport aux genres traditionnels décrits à la section 2.1.1, les tags offrent plusieurs avantages. Ils peuvent s'adapter à la communauté si la définition d'un genre change avec le temps. Ils sont faits par et pour les utilisateurs. Enfin un artiste, un album ou une chanson, peuvent être décrits par un ensemble d'étiquettes, celles-ci n'étant en aucun cas mutuellement exclusives.

2.2 Signal audio et features

Nous présentons ici certaines bases de traitement de signal nécessaires à la compréhension des charactéristiques audio (*features* audio) que nous utilisons par la suite.

TAB. 2.2 – Distribution des types de tag

Type de Tag	Fréquence	Exemples
Genre	68%	heavy metal, punk
Lieu	12%	French, Seattle, NYC
Ambiance	5%	chill, party
Opinion	4%	love, favorite
Instrumentation	4%	piano, female vocal
Style	3%	political, humor
Divers	3%	Coldplay, composers
Personel	1%	seen live, I own it

Ces features, ou attributs, permettent de décrire un signal en totalité ou en partie. De très nombreux features ont été développés, et en aucun cas nous n'essaierons d'en faire un tour complet. Le lecteur curieux peut se référer à l'article d'Aucouturier et Pachet [AP03] ou au livre de Gold et Morgan [GM99] pour un meilleur tour d'horizon. Cette section s'inspire largement du résumé offert par Bergstra [Ber06]. Nous présenterons tout d'abord les caractéristiques physiques de base d'un signal audio. Par la suite, nous verrons la transformée de Fourier qui extrait les fréquences présentes dans un son. Enfin, nous verrons les cepstra, une deuxième transformée qui permet de créer plusieurs features classiques.

2.2.1 Physique

On considère qu'un son est une superposition d'ondes de différentes fréquences, phases et amplitudes. La fréquence peut être reliée aux notes de musique, par exemple un des La au piano a généralement une fréquence de $440Hz$, c'est à dire que l'onde a une période de $1/440s$. La phase donne une indication temporelle. Deux ondes peuvent avoir la même fréquence et amplitude, mais être décalées dans le temps. Elles ont alors une phase différente. Enfin, l'amplitude est proportionnelle au volume.

Dans l'air ambient, un son est transmis par des variations de pression qui se déplacent avec le temps. Il peut être créé ou capté par un courant électrique avec un haut-parleur et un microphone. Pour sa création, le signal passe dans une bobine de fil attachée à un diaphragme et entourée d'un champs magnétique. Ce fil se met alors en mouvement en entraînant le diaphragme, et cela fait varier la pression de l'air. Pour capter un son, on

utilise le même dispositif. Cette fois-ci, le son fait vibrer la membrane et le fil, ce qui crée un courant électrique.

On produit un signal audio numérique en échantillonant souvent et régulièrement le voltage du courant sortant du microphone. L'encodage Pulse Code Modulation (PCM) est tout simplement l'ensemble de ces valeurs. Cet encodage est utilisé notamment par les fichier .wav et les CD audio. Dans le cas du CD, l'échantillonnage est fait à une fréquence de $44100Hz$.

2.2.2 Transformée de Fourier

Le théorème de Fourier stipule que n'importe quelle fonction qui est raisonnablement continue et périodique peut être exprimée comme une somme (possiblement infinie) de fonctions cosinus et sinus. Dans le cas d'un signal audio, cela revient à dire que l'on peut exprimer n'importe quel signal par un ensemble de sinusoïdes de différentes fréquences, amplitudes et phases. La transformée de Fourier est la transformée qui nous donne l'amplitude et la phase pour chaque composante fréquentielle. Cette transformée fonctionne sur des fonctions continues, ce qui n'est pas le cas d'un signal audio numérique. Il en existe cependant une version pour le cas discret, la transformée de Fourier discrète (DFT), présentée dans l'équation 2.1. Ici, k est l'index d'une fréquence, t est l'index d'un échantillon (en suivant l'axe du temps) allant de 0 à T, et $\hat{s}$ est le signal échantillonné.

$$\mathcal{F}_s[k] = \sum_{t=0}^{T-1} \hat{s}[t] \left(\cos(2\pi \frac{kt}{T}) - i \sin(2\pi \frac{kt}{T}) \right) \tag{2.1}$$

Avec la DFT, on ne mesure que les fréquences indexées par k, qui peuvent en théorie aller de 0 à l'infini. Il existe cependant une limite dans la précision que l'on obtient qui dépend de la fréquence d'échantillonage. Le plus simple est d'observer le cas particulier illustré par la figure 2.1. Les valeurs échantillonnées sont les trois points, et les deux sinusoïdes ont exactement la même phase, mais la fréquence de celle hachurée vaut 3 fois la fréquence de la sinusoïde continue. A partir des points échantillonés, il est impossible de savoir de laquelle des deux courbes ils proviennent. Soit r la fréquence d'échantillonage, on appelle $r/2$ la fréquence de Nyquist et on considère que l'on ne peut représenter correctement des fréquences supérieure à celle de Nyquist. De plus, plus la fréquence est proche de celle de Nyquist, moins sa représentation sera précise. On peut se réferer à

Gold et Morgan [GM99] pour une justification mathématique de ce concept.

FIG. 2.1 – Fréquence de Nyquist

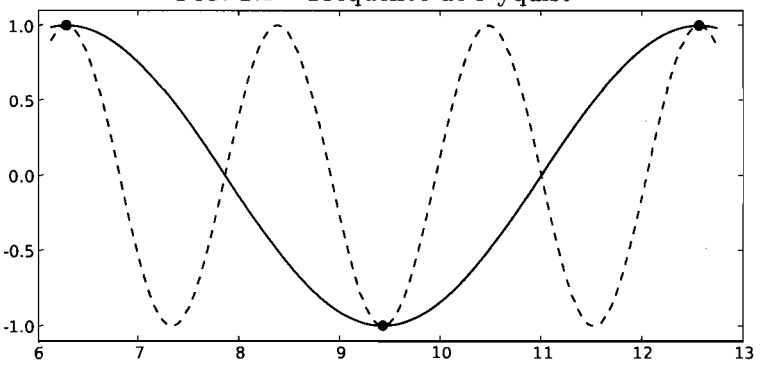


Illustration de la limite de précision d'un échantillonnage. Si nos valeurs échantillonnées sont les points, on ne peut déterminer s'ils proviennent de la courbe continue ou hachurée. La courbe continue à pour fréquence la fréquence de Nyquist, qui est la moitié de la fréquence d'échantillonage.

La DCT retourne un vecteur complexe f_k qui représente la quantité et la phase de l'énergie (ou intensité) pour chacune des $T/2$ fréquences indexées. La majorité des features audio sont basés sur les amplitudes $|f_k|$ représentant l'énergie présente pour la $k$ième fréquence. Un exemple en est le spectrogramme. On calcule la DCT de segments d'une chanson (par exemple des segments de 100 ms) et on concatène les vecteurs d'amplitude. Un exemple en est donné à la figure 4.2 (feature du bas).

2.2.3 Cepstra

Revenons au spectrogramme, il est classique de transformer l'échelle des fréquences par une fonction logarithmique. Notre oreille fait quelque chose de similaire. Les cepstra sont le résultat de la transformée de Fourier appliquée à un signal transformé une première fois par Fourier auquel on a appliqué une échelle logarithmique sur les valeurs absolues des fréquences. En plus clair :

$$signal \rightarrow DFT \rightarrow abs() \rightarrow logarithme \rightarrow DFT \rightarrow cepstra \tag{2.2}$$

Il n'est pas trivial de comprendre la signification physique des cepstra, l'auteur de ce mémoire y travaille encore. Ce qui est certain, c'est qu'il s'agit d'une manière efficace

de comprimer l'information contenue dans un signal audio. Les cepstra sont largement utilisé dans la reconnaissance de parole par exemple, et dans la plupart des travaux de la communauté MIR (*Music Information Retrieval*, voir section 2.4). Le plus simple est de s'en faire une idée par la figure 4.2 (image du haut). Le type de cepstra le plus utilisé est les MFCC (*Mel-Scale Cepstra Coefficients*) qui correspondent à une échelle logarithmique particulière.

2.3 Apprentissage statistique

Nous présentons ici les bases de l'apprentissage statistique (de l'anglais *machine learning* et aussi appelé *apprentissage automatique* ou *apprentissage machine*). Nous mettrons l'accent sur l'apprentissage de classifieurs multiclasses, avec une brève explication de la régression. Nous verrons ensuite le boosting, l'algorithme (ou plutôt *méta-algorithme*) principalement utilisé dans ce travail. Enfin, nous verrons les méthodes d'évaluations de l'erreur de classification. Bien sur, cette section n'est qu'un minuscule survol d'un pan entier de recherche. Le lecteur curieux peut se rapporter aux livres de Duda et al. [DHS00] ou de Bishop [Bis06] pour une introduction plus complète. La structure du résumé ci-dessous est largement inspirée de celle de Bergstra [Ber06].

Nous parlerons ici de x un échantillon d'attributs (ou *features*) parmi X l'ensemble des attributs possibles et de c une classe parmi un ensemble de classes possibles C. L'objectif de l'apprentissage machine est de trouver une fonction $f : X \rightarrow C$ telle que la probabilité conditionnelle $P(f(x)|X = x)$ est maximisée sur X. La fonction f est inférée à partir d'un échantillon S d'exemples (x_i, c_i). En utilisant un anglicisme, nous faisons référence à l'ensemble de données S par le terme *dataset*. La fonction se doît d'être la meilleure approximation possible, et ce sur tout *dataset* et non seulement sur celui utilisé pour l'apprentissage. On parle dans ce cas du *problème de généralisation*.

2.3.1 Évaluation et sélection de modèle

Avant de pouvoir parler de l'apprentissage de la fonction f ou de la comparaison entre différentes fonctions f apprises, il faut pouvoir mesurer la qualité de la classification. Etant donné le problème de généralisation, il faut mesurer la performance de f sur un dataset non vu durant l'apprentissage. Dans le cas contraire, on risquerait d'apprendre

par coeur un seul dataset et de performer de façon catastrophique sur les autres. Nous devons donc définir une *fonction de coût* qui donne une valeur liée à l'erreur de f. La fonction de coût la plus simple en classification est la L_{0-1} définie telle que suit :

$$L_{0-1}(f) := P(C \neq f(X)) \tag{2.3}$$

$$\hat{L}_{0-1}(f|T) := \frac{1}{|T|} \sum_{(x_i,c_i)\in T} \mathrm{I}_{c_i \neq f(x_i)} \tag{2.4}$$

A l'équation 2.4, on approxime L_{0-1} en mesurant le nombre d'éléments mals classifiés par f dans un dataset T, dit dataset de test. Tel que mentionné plus haut, les exemples de T n'ont pas été vus durant l'apprentissage de f.

Précisons la terminologie, un modèle d'apprentissage (*learner*) est un algorithme qui transforme f et, en pratique, qui cherche à minimiser une fonction de coût $\hat{L}(f|T)$. Un classifieur est tout simplement f, une fois appris. La sélection de modèle compare plusieurs classifieurs $\{f_i\}$ et choisit celui qui minimise une fonction de coût $\hat{L}(f|T)$ dans l'espoir de minimiser $L(f|T)$.

Il faut préciser que l'on peut faire de la sélection de modèle sur d'autre critères, par exemple liés à la théorie de l'information. Entre autre, le AIC (*Akaike Information Criterion*, voir [Aka74]) essaie de mesurer l'adéquation entre un modèle et un problème de classification. L'intuition est qu'un problème simple à résoudre ne nécessite qu'une fonction f "simple". Cela dit, étant donné que l'on s'intéresse à la performance en classification de notre modèle, nous ferons de la sélection de modèles basée sur des fonctions de coût liées à l'erreur de classification, telle L_{0-1}.

2.3.2 Ensembles d'entraînement, de validation, de test, minibatch

On précise ici le noms de certains datasets particuliers. L'ensemble d'entraînement est le dataset utilisé pour l'apprentissage de f.

L'ensemble de validation est un ensemble d'exemples ne faisant pas partie de l'ensemble d'entraînement. On peut le voir comme un premier ensemble de test servant à choisir certains paramêtres ou certains modèles.

2.3.3 Classification et régression

Jusqu'à présent nous avons parlé de classification, c'est à dire trouver la bonne classe à partir d'un ensemble de features. Il existe un autre type de prédiction où l'on ne cherche pas une classe mais une valeur. Plus précisément, on cherche à évaluer $E(V = v|X = x)$, où V est un ensemble de valeurs possibles, normalement continues. De la même façon qu'à la section 2.3.1, on peut définir une fonction de coût. Par exemple, l'erreur au carré L_2 se définit par :

$$L_2(f) := E([f(X) - v]^2) \tag{2.5}$$

$$\hat{L}_2(f|T) := \frac{1}{|T|} \sum_{(x_i,v_i)\in T} [f(x_i) - v_i]^2 \tag{2.6}$$

2.3.4 Boosting

Nous décrivons ici AdaBoost et AdaBoost.MH que nous utiliserons pour notre étiquetage automatique. Nous décrirons également FilterBoost, une extension récente, à la section 4.5.

AdaBoost [FS96] est une *méta-méthode* qui construit un classifieur de façon itérative. À l'origine, l'algorithme est conçu pour la classification binaire, mais il a depuis été étendu aux cas multi-classe de plusieurs manières. Nous décrivons ici AdaBoost.MH [SS99], une extension *one-versus-all* permettant de travailler avec k classes. À chaque itération t, l'algorithme appelle un algorithme d'apprentissage (généralement simple) appelé *weak learner*. Celui-ci retourne un *classifieur faible* (ou *weak classifier*) $\mathbf{h}^{(t)}$ et son coefficient $\alpha^{(t)}$. Par abus de langage, on appellera parfois le *weak classifier* le *weak learner*. En pratique, si l'on sait que le modèle d'apprentissage est la descente de gradient dans un réseau de neurones, le classifieur faible est forcément un réseau de neurones et il n'y a donc pas d'ambiguïté. L'entrée du classifieur faible $\mathbf{h^{(t)}}$ est un vecteur $\mathbf{x}$ de dimension d et la sortie désirée est un vecteur binaire $\mathbf{y} \in \{-\mathbf{1}, \mathbf{1}\}^{\mathbf{k}}$ (k classes présentes). Si $h_\ell^{(t)} = 1$, le weak learner "vote pour" la classe ℓ et si $h_\ell^{(t)} = -1$, le weak learner "vote contre". Après j itérations, l'algorithme produit une fonction discriminante à valeur vectorielle

$$f_j(x) = \sum_{t=1}^{j} \alpha^{(t)} \mathbf{h^{(t)}}(\mathbf{x}) \tag{2.7}$$

En supposant un weak learner dont le coût d'apprentissage est linéaire en fonction de n (nombre d'exemples), le coût de l'apprentissage est $O(nkd)$ et l'ensemble de l'algorithme est $O\big(nd(kT+\log n)\big)$ en temps. Il est possible que le boosting ait besoin de plus de weak learners pour un plus grand nombre d'exemple, mais le temps d'apprentissage linéaire en fonction du nombre d'exemples. Ceci est une propriété intéressante si l'on compare avec les SVM à noyaux gaussiens par exemple, qui sont $O(n^2)$

Nous verrons également à la section 4.5 FilterBoost, une extension d'AdaBoost qui utilise le concept de *rejection sampling* et permet de travailler efficacement avec de très grandes bases de données.

2.4 Classification de musique

Nous nous intéressons ici à la classification de musique à partir des attributs audio. L'une des tâches les plus étudiées dans la communauté de *Music Information Retrieval* (MIR) est la reconnaissance de genres. Un concours international sur cette tâche existe depuis 2005 à MIREX [Mir], une série de concours organisée en marge de la conférence annuelle ISMIR[7]. Le but est d'attribuer à une chanson le genre auquel elle appartient parmi un ensemble de genres candidats. La seule information que l'on a est la chanson elle-même (voir la section 2.2 pour un rappel des features que l'on peut extraire d'un signal audio). La reconnaissance de genre sous-entend la simplification suivante, à savoir qu'une chanson ne peut appartenir qu'à un seul genre.

De nombreux algorithmes ont été utilisés pour la reconnaissance de genre. Lambrou et al. [LKS+98] ont utilisé des algorithmes de plus proche voisin. Tzanetakis et Cook [TC02] ont employé des mixtures de gaussiennes. West et Cox [WC04] classifient les segments d'une chanson par mixture de gaussiennes, Linear Discriminant Analysis (LDA) et arbre de régression. Ahrendt et Meng [AM05] classifient eux-aussi des segments au moyen d'une régression logistique multiclasse. Bergstra [Ber06] utilise des réseaux de neurones. Les machines à support de vecteurs (SVM) ont aussi été très populaires. Voir par exemple Li et al. [LOL03] ou Mandel et Ellis [ME05]. L'approche que nous utilisons dans la suite de ce travail utilise AdaBoost qui fut utilisé par Bergstra et al. [BCE05, BCE+06].

À savoir si la classification doit se faire sur des segments d'une chanson ou sur des

[7] www.ismir.org

features représentant l'ensemble de cette chanson, Bergstra et al. [BCE+06] montrent de façon convaincante que l'approche par segment améliore les résultats de manière significative.

Bien que nous n'ayons parlé que de classification à partir de features audio, il vaut la peine de mentioner d'autres approches, en particulier la reconnaissance de genre à partir de l'Internet. Par exemple, Bergstra et al [BLE06] utilisent FreeDB[8], une base de données dont le but est d'étiqueter correctement les CD personnels. À partir de l'information recueillie pour un artiste, ils arrivent à prédire le genre défini par AMG[9]. Un autre exemple se trouve dans Knees et al. [KPW04]. D'une façon similiare, Celma et al. [CCH06] développent un algorithme de recherche de musique basé sur les blogues en ligne.

La recherche de musique est l'une des applications où la classification de musique devrait s'avérer utile. On aimerait pouvoir chercher une chanson à partir de mots-clés. Par exemple, une chanson *énergique* interprétée par une *chanteuse* dans les années *90* avec beaucoup de *guitare*. Malheureusement, cela va à l'encontre de la reconnaissance de genre qui n'associe qu'une catégorie à une chanson. Il faut donc généraliser ce concept de genre pour le rendre plus flexible. Une telle idée est développée par McKay et Fujinaga [MF06]. La clé de voûte en est qu'un artiste, un album ou une chanson peuvent appartenir à plusieurs genres ou catégories, quitte à l'être avec un certain degré d'appartenance.

2.5 Similarité musicale

2.5.1 Description de la tâche

La similarité musicale revient à déterminer quelles chansons ou artistes sont *proches* selon un certain critère. Ce critère peut prendre différentes formes. Par exemple, des artistes peuvent êtres considérés proches s'ils sont écoutés par les même personnes, s'ils ont été produits à la même époque ou par la même équipe, ou encore s'ils abordent des sujets semblables.

Un des problèmes fondamentaux de cette tâche est évidemment de trouver une similarité de référence, ou *ground truth* (Ellis et al. [EWBL02]). Cette tâche a quand même été trés étudiée par la communauté MIR. En particulier, la similarité musicale est l'un

[8] www.freedb.org
[9] www.allmusic.com

TAB. 2.3 – MIREX 2007 - Similarité

Position	Équipe	Score
1	T. Pohle & D. Schnitzer [PS]	0.568
2	G. Tzanetakis [Tza]	0.554
3	L. Barrignton et al. [BTTL]	0.541
4	C. Bastuck [Bas]	0.539

Meilleurs résultats à l'épreuve de similarité musicale (MIREX 2007) [Mir] basée sur la *mesure fine*.

des concours annuels à MIREX [Mir]. Le tableau 2.3 présente les équipes gagnantes au concours de 2007. Il est à noter que l'équipe de Barrington et al. [BTTL] utilise une approche par étiquetage similaire à celle présentée dans ce travail.

2.5.2 Méthodes de mesure

Un autre défi de la similarité musicale est la mesure de la performance d'un modèle. Nous présentons ici trois mesures qui comparent des listes ordonnées et dont nous nous servirons au chapitre 5 pour mesurer notre modèle.

La première, *TopN*, compare deux listes ordonnées : une liste cible A et une liste prédite B. Cette mesure a été introduite par Berenzweig et al. [BLEW03] et met l'accent sur la qualité avec laquelle la liste candidate positionne les éléments au sommet de la liste cible. Soit k_j la position dans la liste B du $j$ième élément de la liste A. On pose $\alpha_r = 0.5^{1/3}$ et $\alpha_c = 0.5^{2/3}$ comme dans [BLEW03]. Le résultat est une valeur entre 0 (listes différentes) et 1 (top N des listes identiques).

$$s_i = \frac{\sum_{j=1}^{N} \alpha_r^j \alpha_c^{k_j}}{\sum_{l=1}^{N} (\alpha_r \alpha_c)^l} \tag{2.8}$$

Pour les résultats présentés ici, nous prenons $N = 10$.

Notre seconde mesure est le Kendall's Tau, une mesure typique en filtrage collaboratif qui mesure le nombre de paires *discordantes* dans 2 listes. Soit $R_A(i)$ le rang d'un élément i dans une liste A, si i n'est pas explicitement présent, $R_A(i) = longueur(A)+1$. Soit C le nombre de paires *concordantes* (i, j), à savoir $R_A(i) > R_A(j)$ et $R_B(i) > R_B(j)$. De façon similaire, soit D le nombre de paires discordantes. Nous utilisons la définition de Kendall's

Tau de Herlocker et al. [HKTR04]. Nous définissons aussi T_A et T_B le nombre déléments ayant la même position (éléments ex aequo) dans la liste A et B respectivement. Dans notre cas, c'est le nombre d'artistes qui sont dans A mais pas dans B, parce qu'ils sont considérés avoir la même position $R_B(i) = longueur(B)+1$, et réciproquement. Kendall's *Tau* est défini de la façon suivante :

$$\tau = \frac{C - D}{sqrt((C + D + T_A)(C + D + T_B))} \tag{2.9}$$

Si rien d'autre n'est précisé, nous analysons le top 50 des deux listes.

Finalement, nous calculons ce que nous appelons *TopBucket*, qui est simplement le pourcentage d'éléments en commun dans le top N de deux listes ordonnées. Si rien d'autre n'est précisé, nous comparons le top 20 de deux listes.

CHAPITRE 3

BASES DE DONNÉES : MUSIQUE ET ÉTIQUETTES

Nous présentons ici les bases de données utilisées dans ce travail. Rassembler des données aussi diverses que de l'audio, des ensembles d'étiquettes (ou tags), ou encore des informations touchant à la similarité musicale n'est pas chose facile. Cela étant dit, à l'ère du Web 2.0, beaucoup de données sont disponibles et le défi est de mettre les différents sources d'information en lien l'une avec l'autre. Nous montrons ici comment certaines bases de données se complètent et peuvent être utilisées.

3.1 USPOP et MusicSeer

USPOP[1] regroupe un ensemble de chansons et d'informations relatives à celles-ci (fournies par AMG, voir section 3.3). La liste des chansons est disponible, mais pas l'audio. Dan Ellis propose cependant les MFCC (voir section 2.2.3) précalculés. USPOP veut représenter la musique populaire. Les artistes ont étés sélectionnés d'après leur popularité sur le réseau *peer-to-peer* OpenNap[2] en 2001. Au moins un album a par la suite été acheté pour chacun des 400 artistes retenus. Les statistiques de base de USPOP, telles que prises sur son site web, sont résumées dans le tableau 3.1. Un sous-ensemble de USPOP à déjà été utilisé comme ensemble d'évaluation à MIREX [Mir] pour le concours de reconnaissance de genre.

TAB. 3.1 -- USPOP

	# chansons	# albums	# artistes	# styles	# genres
USPOP	8764	706	400	251	10

Statistiques de la base de données USPOP.

Une autre application possible de USPOP est en similarité. D. Ellis et ses collègues ont développé et testé plusieurs métriques de similarité par rapport à cette base de données

[1] http://labrosa.ee.columbia.edu/projects/musicsim/uspop2002.html
[2] http://www.ee.columbia.edu/~dpwe/research/musicsim/opennap.html

et ont rendu les distances entre artistes disponibles[3]. En particulier, ils ont créé un jeu en ligne appelé MusicSeer[4]. Les visiteurs devaient explicitement choisir dans une liste la chanson la plus similaire à une chanson cible. Cela donne sans doute l'une des meilleure *ground truth* en similarité musicale, et il est dommage que ce projet se soit limité à un petit sous-ensemble d'artistes. Nous utilisons néanmoins cette *ground truth* dans Eck et al. [ELBMG08] pour valider nos résultats. Pour plus d'information sur les travaux de Dan Ellis en similarité, voir notemment Ellis et al. [EWBL02] et Berenzweig et al. [BLEW03]. Pour des projets similaires (et améliorés) à MusicSeer, voir la section 3.7.

3.2 Magnatune

Magnatune[5] est un site offrant de la musique sous différentes licenses. En deux mots, les artistes qui signent avec Magnatune gardent les droits sur leurs oeuvres, mais autorisent Magnatune à les distribuer. Le coût dépend du projet pour lesquelles ces chansons sont destinées et la moitié des profits revient aux artistes. C'est un incontournable pour obtenir des chansons de qualité professionnelle puisqu'il n'en coûte rien si on les utilise à des fins non-commerciales ou de recherche. Certaines de ces chansons ont d'ailleurs été utlisées pour le concours de similarité musicale à MIREX [Mir]. De notre côté, nous nous en sommes servi pour différentes démonstrations reliées à notre travail. Le seul problème de Magnatune est la faible popularité (voir nulle) des oeuvres qu'on y retrouve.

3.3 AllMusic

Aussi appelé *All Music Guide* ou *AMG*, c'est un service en ligne[6] professionnel qui fournit de l'information sur les artistes et leur musique. On y retrouve entre autres les noms d'artistes, des détails biographiques, leur discographie, des photos, des liens vers d'autres groupes similaires, etc. Le contenu est maintenu à jour par des experts. Pour ceux qui connaissent, on peut faire le parallèle avec IMDB[7] pour les films.

[3] http://www.ee.columbia.edu/~dpwe/research/musicsim/
[4] http://www.ee.columbia.edu/~dpwe/research/musicsim/musicseer.html
[5] http://magnatune.com/
[6] http://allmusic.com/
[7] www.imdb.com

TAB. 3.2 – MusicBrainz et *The Beatles*

Position	Artiste	Position	Artiste
1	The Beach Boys	7	Jimi Hendrix
2	Beatless	8	John Denver
3	Bob Dylan	9	Nirvana
4	Boston	10	Rod Stewart
5	U2	11	Sheryl Crow
6	Cat Stevens	12	The Kinks

Artistes similaires à *The Beatles* d'après *MusicBrainz.*

3.4 MusicBrainz et collection personnelle

MusicBrainz[8] peut être décrit comme une version communautaire de AMG (section 3.3), où l'information est produite et contrôlée par des utilisateurs bénévoles. On y retrouve sensiblement les mêmes informations, par exemple des artistes similaires (table 3.2) et des étiquettes ou mots-clés. Par exemples, les tags appliqués à *The Beatles* sont *british*, *english*, *liverpool*, *parlophone*, *pop*, *rock* et *uk*.

La base de données de MusicBrainz peut être téléchargée. Il faut noter que chaque artiste, album et chanson y reçoivent un identifiant unique et permanent. Nous utilisons ces identifiants pour gérer notre collection personnelle, et nous en encourageons l'adoption par les autres laboratoires. Ces identifiants peuvent régler une fois pour toute le problème des titres et noms mal orthographiés lorsque l'on veut partager une liste de chansons.

3.5 AudioScrobbler

Audioscrobbler [Aud] est un service web qui donne accès à de nombreuses données provenant du site Last.fm. Les donnée sont gratuites tant qu'elles sont utilisées à des fins non lucratives, et donc parfaites pour la recherche. Le plus simple pour présenter AudioScrobbler est de donner des exemples de ce qui y est accessible.

Les données sont divisées en huit catégories :

- User Profile Data
- Artist Data

[8] `http://musicbrainz.org/`

- Album Data
- Track Data
- Tag Data
- Group Data
- Forum Data
- Geo-aware Data

Dans ce qui nous intéresse, dans la catégorie *User Profile Data*, nous pouvons obtenir les données appelées *Top Artists*, les artistes les plus joués par un utilisateur particulier. Cela est utilisé dans la section 5.2 pour construire une similarité de référence. Relativement aux tags, nous pouvons obtenir dans la catégorie *Tag Data* les données *Overall Top Tags* qui nous donnent les tags les plus populaires sur Last.fm et les données *Top Artists* qui sont les artistes les plus étiquetés avec un tag particulier. Cela nous permet de créer nos exemples d'entraînement au chapitre 4 et de mesurer la similarité induite par les tags de Last.fm au chapitre 5.

3.6 CAL500

Afin d'obtenir des étiquettes, nous avons vu AudioScrobbler [Aud] à la section 3.5. Ces tags, très nombreux, sont aussi très bruités puisqu'il n'y a aucun contrôle sur la manière dont ils sont attribués. CAL500[9] offre une alternative à cela. L'équipe de G. Lanckriet de UCSD[10] a payé des étudiants pour annoter 500 chansons. Cela forme la base de donné CAL500 introduite dans Turnbull et al. [TBTL07]. Les étudiants devaient écouter des extraits de chansons et répondre aux questions d'un formulaire prédéfini. Le vocabulaire (l'ensemble des tags) a une taille de 174.

De façon similaire à USPOP (section 3.1), les étiquettes apposées et certains features sont disponibles en ligne mais pas l'audio. F. Maillet[11] a utilisé CAL500 dans Bertin-Mahieux et al. [BMEML08] pour comparer notre algorithme à celui développé à UCSD. Il y a cependant de nombreuses critiques à formuler concernant ces données, la principale étant le faible nombre d'annotations. La plupart des chansons n'ont été écoutées que par 3 personnes, et l'avis d'un si petit nombre peu difficilement être considéré fiable. Cela

[9] http://cosmal.ucsd.edu/cal/projects/AnnRet/AnnRet.php
[10] http://www.ece.ucsd.edu/~glanckriet/
[11] http://www-etud.iro.umontreal.ca/~mailletf/

est d'autant plus problématique lorsque l'on s'intéresse à des tags subjectifs, par exemple *original* ou *party*.

3.7 Jeux en lignes

Nous avons vu les tags fournis par les sytèmes en ligne tels que AudioScrobbler (section 3.5) et ceux obtenus en payant des gens pour annoter des chansons (section 3.6). Nous parlons ici d'une troisième idée, les jeux en ligne, qui veut compenser les lacunes des deux précédentes. Principalement, le coût pour obtenir des tags doit être gratuit (outre les frais de maintenance d'un site web) mais l'environnement dans lequel se fait l'étiquetage doit être mieux contrôlé.

Le premier de ces jeux, le ESP game [vAD04], a été créé pour annoter des images. Le principe en est le suivant : deux joueurs qui ne se connaissent pas annotent la même image en même temps. Ces joueurs gagnent des points s'ils utilisent les même mots pour décrire ce qu'ils voient et cela encourage ainsi l'utilisation de tags pertinents. Plusieurs mécanismes de contrôle peuvent être ajoutés. On peut régulièrement changer les partenaires pour les dissuader de prendre contact et de tricher. On peut aussi montrer plusieurs fois la même image à un joueur pour s'assurer qu'il soit constant dans ses annotations. On obtient ainsi avec des étiquettes obtenues gratuites mais généralement de très bonne qualité.

FIG. 3.1 – ListenGame

Aperçu du jeu *ListenGame* développé à UCSD.

Suivant ce modèle, de nombreuses équipes de recherche ont créé des jeux en ligne pour obtenir des étiquettes fiables. N'ayant pas utilisé ces données dans nos travaux, nous n'expliqueront pas les particularités de chacun. Le lecteur curieux, et qui de plus aimerait contribuer à la recherche, peut jouer aux jeux de CMU [LvADC07], de Columbia [ME07] ou de UCSD [TLBL07]. Un aperçu de ce dernier est montré à la figure 3.1. Geste à noter, à notre connaissance, les trois équipes mentionnées ci-dessus comptent rendre public les données obtenues par leur jeu[12].

Nous avons mentionné à la section 3.1 le jeu en ligne MusicSeer sans en avoir parlé dans cette section. La raison en est que MusicSeer n'offrait pas de récompense aux utilisateurs en fonction de leur performance, il n'y avait aucun système de point et les participants y jouaient de façon bénévole pour aider la recherche. Sachant cela, nous pouvons considérer MusicSeer comme la première génération de jeu en ligne sur les tags, et ceux présentés dans cette section comme la deuxième génération, plus accrocheurs pour l'utilisateur.

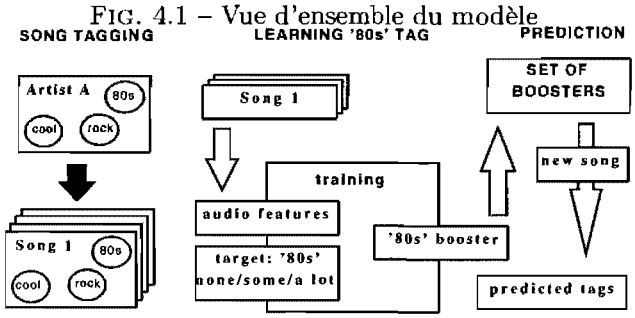
[12]Tout récemment, un premier concours d'étiquetage automatique a été introduit à Mirex 2008 [Mir] sous le nom de *Audio Tag Classification*. Les données utilisées proviennent du jeu de Columbia [ME07] et E. Law de CMU [LvADC07] a proposé d'utiliser son jeu pour "faire jouer" les algorithmes du concours avec des humains.

CHAPITRE 4

ÉTIQUETAGE

Ce chapitre présente notre modèle en lui-même. À la section 4.1 nous faisons un survol général et donnons des détails quand aux données utilisées. À la section 4.2, nous précisons les types de *features* audio utilisés. Ensuite, nous présentons une première méthode basée sur AdaBoost.MH tel que vu à la section 2.3.4. Cet algorithme comporte plusieurs problèmes que nous essayons de résoudre en introduisant FilterBoost à la section 4.5.

FIG. 4.1 – Vue d'ensemble du modèle



Vue d'ensemble de notre modèle d'étiquetage. Nous attribuons aux chansons les tags que les artistes ont reçus. Nous utilisons les charactéristiques audio (ou *features* audio) de ces chansons et des tags cibles pour entraîner un classifieur par tag. Ces classifieurs peuvent étiqueter automatiquement une nouvelle chanson.

4.1 Vue d'ensemble de notre modèle

Nous présentons ici notre modèle tel qu'illustré par la figure 4.1. L'idée principale est d'entraîner un classifieur par étiquette (ou *tag*). L'idéal serait d'avoir une grande base de données de tags par chanson, mais la répartition de ces tags dans le service AudioScrobbler [Aud] est très inégale. Pour obtenir un nombre raisonnable d'étiquettes, nous utilisons les tags appliqués aux artistes. Nous faisons ensuite la simplification suivante, à savoir qu'un tag appliqué à un artiste l'a été à toutes les chansons de cet artiste. Pour décrire mathématiquement les chansons, nous utilisons certains features audio présentés

à la section 2.2 et justifiés à la section 4.2. Une intuition en est donnée par la figure 4.2.

Le coeur de notre algorithme réside dans l'entraînement d'un classifieur par tag. Nous tenons à apprendre chaque tag séparemment, et non tous avec un seul classifieur. Il est évident qu'il existe des liens entre les différents tags : *electro* et *techno* sont souvent appliqués aux mêmes chansons, de même que *rock* et *classic rock*. Cela dit, nous voulons un système qui puisse s'adapter à de nouvelles et nombreuses données. En particulier, il serait dommage d'avoir un vocabulaire (ensemble d'étiquettes) prédéfini et fixe. Si un nouveau tag populaire apparaît parmi les utilisateurs, de nombreux algorithmes d'apprentissage statistique nécessiteraient de réentraîner tout le système pour pouvoir l'utiliser. En ayant un classifieur par étiquette, on peut facilement augmenter le vocabulaire et réentraîner un seul tag si l'on juge cela nécessaire.

Que l'on essaie de prédire la probabilité qu'un tag soit appliqué à une chanson ou bien le nombre de fois que ce tag y sera appliqué, on fait face à un problème de régression (Section 2.3.3). Malheureusement, l'ensemble d'étiquettes qui nous sert à l'apprentissage n'est pas assez grand et précis pour permettre un tel apprentissage. Nous simplifions donc l'apprentissage en en faisant un problème de classification. Le but sera d'apprendre un nombre fini de classes pour chaque tag. Par exemple, pour le tag *rock*, les classes pourront représenter les concepts *pas rock*, *un peu rock*, et *beaucoup rock*. Différentes façons de passer d'une régression à une classification sont présentées à la section 4.3.

4.2 Entrées : attributs audio

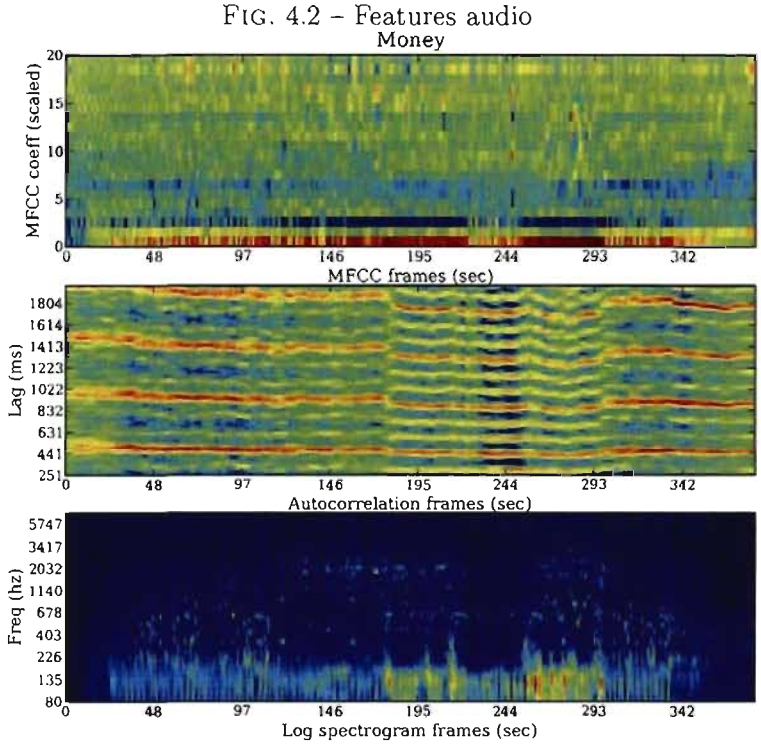
Nous représentons chaque segment de 5s d'audio par un ensemble de trois features. La figure 4.2 montre ces features extraits pour la chanson *Money* de Pink Floyd. Premièrement nous retrouvons les *autocorrelation coefficients* (coefficients d'autocorrélation). Ceux-ci donnent des informations sur le rythme, et ont d'ailleurs été utilisé pour du *beat tracking* [Eck07]. Ses propriétés son visibles dans la figure 4.2 (milieu). La chanson commence en 7/8 (7 temps de croche), une signature rythmique rare en musique. Après 190s environ, le solo commence et la signature rythmique revient en 4/4, ce qui est beaucoup plus commun. Ce changement de rythme est parfaitement visible dans la figure.

Deuxièmement, nous utilisons les *mel-scale frequency cepstra coefficients* (MFCC) déjà vus à la section 2.2.3. Ceux-ci donnent de l'information sur le timbre. Par exemple,

dans la figure 4.2 (haut), on voit les basses fréquences gagner en intensité (plus rouge) lorsqu'une trompette commence à jouer, un peu avant le solo.

Enfin, nous calculons le *constant-Q spectrogram*, qui est une variante de spectrogramme où les fréquences passent par une échelle logarithmique. Ce feature contient différentes informations sur la quantité d'énergie présente et sur le timbre, entre autres.

FIG. 4.2 – Features audio



Features audio pour "Money" de *Pink Floyd*.

Dans les trois cas, on ne garde que la valeur moyenne et la variance sur 5s, ce qu'on appelle les features aggrégués.

Pour chaque 5s, le feature d'autocorrélation a une taille de 176, celui des MFCC 40 (les 40 premiers coefficients), et celui du spectrogramme 170. Le total donne une taille de 386 pour les features d'un segment de 5s.

4.3 Régression et classification

L'étiquetage automatique est essentiellement un problème de régression, peu importe que l'on prédise la probabilité qu'un tag soit appliqué ou encore le nombre de fois qu'il le sera (en anglais son *tag count*). Cela dit, nos données ne nous permettent pas une telle précision du fait qu'elles soient bruitées et parcimonieuses (de l'anglais *sparse*). Nous transformons donc notre problème en classification.

Pour chaque tag, nous classons les artistes selon leur poids pour ce tag calculé par la formule suivante :

$$\text{poids} = \frac{\#\text{ fois ce tag appliqué à cet artiste}}{\#\text{ fois n'importe quel tag appliqué à cet artiste}} \tag{4.1}$$

Ceci peut être vu comme une approximation de la probabilité qu'un tag t soit appliqué à un artiste a : $P(tag = t|artist = a)$.

Par exemple, dans le tableau 4.1, nous voyons la position de certains artistes pour le tag *pop* par rapport au poids attribué par la formule 4.1. Il est difficile de mesurer l'efficacité de ce poids mais notre avis subjectif est que les artistes se retrouvant au sommet sont généralement peu discutables. On voit cependant apparaître les différents sens d'une même étiquette. Dans ce cas-ci, on serait tenté de dire que le *pop* de Madonna n'est pas le même que celui des Beatles (l'époque en particulier est très différente). C'est pourtant un tag qui est régulièrement attribué à ces deux artistes.

Nous créons nos classes à partir de ce classement. Une première idée développée dans Eck et al. [EBML07, ELBMG08] est de créer trois classes balancées représentant (pour le tag *pop*) *pas du tout pop*, *un peu pop*, et *très pop*. Cela impliquerait de mettre les premiers 500 artistes de la liste dans la classes *très pop*, les 500 suivants dans *un peu pop*, et le reste de notre base de donnée dans *pas du tout pop*. Afin de balancer les classes, on pourrait prendre davantage d'exemples (segments) de chaque chanson pour les 2 premières classes, ou mettre plusieurs fois le même segment. Il n'est pas certain qu'il faille utiliser des classes bien balancées, mais le choix semble justifié lorsque l'on essaie d'apprendre des tags très peu utilisés. Si notre classifieur ne voit que 10 chansons d'un genre, et $100K$ contre-exemples, il n'apprendra qu'à prédire que le tag n'est jamais approprié. Cela implique que dans l'apprentissage de notre classifieur, les classes populaires seront sous-pondérées, e.g. la probabilité d'un artiste d'être classé comme "pop" sera plus faible que dans la

TAB. 4.1 – Artistes *pop*

position	artiste	poids
1	Madonna	0.03225
2	The Beatles	0.03246
3	Black Eyed Peas	0.03071
4	The Beach Boys	0.03068
5	Kelly Clarkson	0.03047
	...	
9	Nelly Furtado	0.02989
10	Coldplay	0.02988
11	Britney Spears	0.02987
	...	
999	Phixx	0.01653
1000	Kristian Leontiou	0.01653

Echantillon d'artistes classés selon leur poids pour le tag *pop* calculé par l'équation 4.1.

réalité. Nous faisons consciemment ce choix afin que notre classifieur puisse correctement utiliser tous les tags existant, et même être plus versatile qu'un humain typique.

L'approche précédente essaie d'approximer la régression par 3 classes. Il n'est pas évident que la classe du milieu, *un peu pop*, soit très utile. On peut espérer que le classifieur soit ainsi amené à différencier les artistes *vraiment pop* des autres, mais en pratique ce n'est pas clair qu'il y parvient. De plus, notre ordonnancement des artistes selon l'équation 4.1 n'étant pas parfait, cela revient parfois à différencier entre deux artistes qui sont tous les deux *pop*. Pour la suite de nos travaux, nous modifions la classification pour en faire un cas binaire. Nous ne conservons que les artistes les plus représentatifs d'un tag, le top 10 dans nos expériences. Pour le tag *pop*, d'après le tableau 4.1, cela signifie les artistes allant de Madonna à Coldplay. Les artistes suivants de la liste (jusqu'à la position 1000) sont ignorés car nous ne sommes pas sûr qu'ils représentent correctement le tag. Ils ne seront jamais vu par le classifieur durant l'apprentissage. Enfin, le reste de la base de données formera les exemples négatifs. Cette option offre plusieurs avantages. Premièrement, en apprentissage statistique, plus de classifieurs ont été développés pour les problèmes binaires, ce qui offre plus de possibitités et de comparaisons à long terme. Ensuite, nous espérons apprendre des tags représentant un petit sous-ensemble de la musique. Cela est sans doute plus utile qu'un tag *pop* que l'on peut probablement appliquer à la majorité des oeuvres des dernières décennies. Enfin, les artistes ignorés

durant l'apprentissage forment un ensemble de test très intéressant pour voir ce qu'a appris notre classifieur.

4.4 Étiquetage par AdaBoost

Nous avons vu aux section précédentes comment obtenir des features audio et comment attribuer à chaque exemple une classe pour chaque tag. Nous donnons maintenant les détails sur l'utilisation d'AdaBoost pour prédire des autotags. Nous présentons aussi quelques résultats liés à cette méthode.

4.4.1 Description

Nous créons un ensemble d'apprentissage en balançant les classes. Soit l'on prend plusieurs copies des exemples de la classe sous-représentée (lourd en mémoire), soit l'on prend plus d'exemples des classes sous-représentées (méthode d'échantillonage) dans le cas où il y a trop d'exemples pour tous les prendre. Nous estimons au moyen d'un ensemble de validation le nombre de *weak learners* nécessaires. Dans nos expériences, l'apprentissage ralentissait rapidement puis stagnait. Nous avons choisi d'apprendre 2000 *weak learners*, ce qui semblait raisonnable et garantissait que l'apprentissage avait atteint son quasi-maximum. Nos *weak learners* sont des *single stumps*, à savoir un seuil (*threshold*) sur l'une des dimensions des features.

Une fois le classifieur fort appris, nous pouvons l'utiliser sur tout exemple, vu ou non au cours de l'apprentissage. Pour un exemple (un segment de 5s d'une chanson), nous prenons la valeur prédite par AdaBoost pour chacune des classes, et nous transformons ces valeurs en une seule, l'autotag. Transformer ces valeurs en une n'est pas trivial, surtout si l'on travaille avec trois classes. Une première idée est de prendre le barycentre des trois valeurs, mais cela peut entraîner des valeurs extrêmes absurdes. Une solution cohérente est de soustraires à la valeur de la classe positive (exemple, *très rock*), la valeur de la classe négative (exemple, *pas du tout rock*). Dans le cas à trois classes, cela implique qu'on n'utilise pas la valeur de la classe intermédiaire (exemple, *un peu rock*) mais il est possible que d'avoir cette classe durant l'apprentissage aide néanmoins en forçant le boosting à séparer davantage les classes.

4.4.2 Expériences

Nous avons utilisé l'étiquetage par AdaBoost avec les 60 tags les plus populaires sur Last.fm. Bien que le résultat qui compte réellement est la qualité de l'autotag créé (ce que nous testons au chapitre 5), nous donnons dans le tableau 4.2 l'erreur de classification. Ce tableau donne le résultat moyen de 60 tests de classifications, un par tag. Dans chacun de ces tests, chaque exemple appartenait à une classe sur trois, les classes représentant *pas du tout*, *un peu*, et *très* pour le tag en question. La classification se fait par segment, et l'on obtient la classe d'une chanson en prenant la classe la plus présente parmi les segments. Cela est similaire à ce qui a été fait par Bergstra et al. [BCE+06]. L'erreur de classification est caluclée selon la formule classique : $\sum_i I_{(\text{argmax}_k f(x,k))\neq y}$. Un algorithme aléatoire donnerait une erreur de 66%.

TAB. 4.2 – Erreur de classification

	Mean	Median	Min	Max
Segment	40.93	43.1	21.3	49.6
Song	37.61	39.69	17.8	46.6

Moyenne des erreurs de test de classification (%) de 60 tags lorsque l'on veut prédire la classe des exemples (segments ou chansons).

Ces chiffres ne montrent qu'une seule chose, c'est que l'on apprend effectivement quelque chose. Il faut noter que, pour ce travail, nous utilisons notre propre code mais la plupart des résultats sont reproductible à l'aide de MultiBoost [Mul], très rapide et disponible gratuitement.

4.5 Etiquetage par FilterBoost

Nous présentons ici une extension d'AdaBoost qui permet de mieux travailler avec de très grandes bases de données. FilterBoost [BS08] à été présenté à la conférence NIPS07[1], et peu de choses ont été écrites à son sujet. Il convient cependant parfaitement à notre problème de classification musicale : beaucoup de données, une base de données éventuellement distribuée et l'impossibilité de tout garder en mémoire. Nous

[1] www.nips.cc

décrivons FilterBoost à la section 4.5.1 et expliquons certaines modifications à la méthode d'étiquetage à la section 4.5.2. Enfin, nous présentons d'autres résultats.

4.5.1 D'AdaBoost à FilterBoost, description de l'algorithme

Nous considérons ici connu et compris l'algorithme AdaBoost présenté à la section 2.3.4. FilterBoost est une adaptation de cet algorithme qui utilise le concept de échantillonage par rejet (*rejection sampling*). L'objectif principal est de travailler avec de très larges bases de données où il est impensable de garder tous les exemples en mémoire. Pour ce faire, nous ne prenons en mémoire que de petits ensembles d'exemples (*minibatch*) à la fois. AdaBoost doit repondérer tous les exemples à chaque itération en fonction de la difficulté avec laquelle ils sont classifiés. Cela devient évidemment impossible avec les minibatch. Au lieu de cela, nous ajoutons un oracle et un filtre à notre modèle. Le rôle de l'oracle est d'aller chercher des exemples dans la base de données et de les montrer au filtre. Celui-ci les accepte selon la difficulté avec laquelle le classifieur fort actuel les classifie. En résumé, au lieu d'augmenter le poids des exemples difficiles, on montre plus souvent ces exemples au booster en les incluant dans les minibatch. Aussi, le poids de chaque nouveau weak learner est évalué sur une nouvelle minibatch.

De façon plus détaillée, nous nous restreignons maintenant à deux classes. Théoriquement, rien n'empêche FilterBoost de fonctionner dans le cas multiclasse, mais cela n'a pas encore été étudié. À chaque fois qu'on le lui demande, l'oracle va chercher un échantillon positif avec probabilité $1/2$ et négatif avec la même probabilité. À l'itération $t+1$, le filtre reçoit un exemple (x, l) avec x features et l classe, et l'accepte avec probabilité

$$q_t(x, l) = \frac{1}{1 + e^{l f_t(x)}} = \text{sigmoid}(l f_t(x)) \tag{4.2}$$

Notons que cette probabilité utilise une fonction logistique, semblable à LogitBoost [FHT98], mais rien n'empêcherait d'utiliser une fonction exponentielle plus similaire à AdaBoost.

L'échantillonage continue jusqu'à ce qu'une minibatch (d'habitude 3000 exemples dans nos expériences) soit construite, et nous choisissons le meilleur weak learner $h^{(t+1)}(x)$ sur ces exemples de la même façon qu'avec AdaBoost. Pour obtenir le poids de ce nouveau classifieur faible, nous utilisons la fonction *getEdge(h(x))* (algorithme 1) où l'on mesure

le nombre de fois que le weak learners se trompe sur de nouveaux exemples. A titre de comparaison, l'*edge* dans AdaBoost est définit par : $\gamma = \sum_i w_i * h(x_i) * y_i$.

Algorithm 1 Pseudocode de getEdge(h(x))

$m \leftarrow 0$, $n \leftarrow 0$, $u \leftarrow 0$
for iter $\leftarrow$ 1 to batchsize **do**
  $(x, l) \leftarrow$ filter()
  $n \leftarrow n + 1$
  $m \leftarrow m + I(h(x) = l)$
  $u \leftarrow m/n - 1/2$
end for
return u

On obtient $\gamma =$ *getEdge(h(x))*, et le poids α_{t+1} est calculé par la formule 4.3.

$$\alpha_{t+1} = \frac{1}{2} \ln \left(\frac{1/2 + \gamma}{1/2 - \gamma} \right) \tag{4.3}$$

On peut aussi choisir plus d'un weak learner par itération en utilisant la méthode habituelle de pondération d'AdaBoost sur la minibatch. Conceptuellement, tous les weak learners choisis sont regroupés et considérés comme un seul weak learner au moment d'évaluer leur poids. Dans nos expériences, nous choisissons 50 nouveaux weak learners par itération.

4.5.2 Etiquetage

L'utilisation de FilterBoost nous pousse à une simplification majeure, à savoir se restreindre au cas binaire. Pour rendre les classifieurs plus comparable entre eux, nous normalisons la somme des poids des weak learners à 1. Ceux-ci font une prédiction entre -1 et 1, et leur poids est compris entre 0 et 1. Nous prenons cette valeur a et appliquons la transformation suivante : $1/2 + a/2$. On obtient ainsi un autotag entre 0 et 1, dont une valeur supérieure à 0.5 signifie que cet exemple est considéré comme positif.

4.5.3 Expériences

Nous avons entraînés 360 tags avec FilterBoost, 6 fois plus qu'avec AdaBoost, ce qui s'explique par la rapidité accrue de l'entraînement. Les exemples positifs sont les chansons des 10 artistes les plus représentatifs d'un tag, les chansons des artistes suivants sont

TAB. 4.3 – Résultats de classification - FilterBoost

Boosters	Positive (10 artists)	Ignore (100 songs)	Negative (200 songs)
main genres (rock, pop, Hip-Hop, metal, jazz, dance, Classical, country, blues, reggae)	89.1%	80.6%	80.0%
instruments (piano, guitar, saxophone, trumpet)	87.0%	61.0%	82.4%
mood (happy, sad, romantic, Mellow)	87.8%	67.4%	79.0%
all	88.2%	60.0%	81.4%

Résultat de classification par chanson, pourcentage des chansons considérés positives (pour les exemples positifs et ignorés) et négatifs pour les exemples négatifs. Un algorithme aléatoire aurait un résultat de 50%.

ignorés, et le reste de la base de donnée produit les exemples négatifs (voir section 4.3).

Nous présentons dans le tableau 4.3 le pourcentage d'éléments classifiés positifs (pour les examples positifs et ignorés) et négatifs (pour les exemples négatifs, le reste de la base de donnée). Ces deux derniers chiffres sont les plus significatifs, le premier étant clairement lié à l'erreur sur l'ensemble d'entraînement. Nous voyons encore une fois que les boosters apprennent quelque chose, et qu'ils semblent généraliser aux exemples ignorés. Aussi, certains types de tags sont appris plus facilement, les genres par exemple. Certains tags plus difficiles n'apprennent presque rien, comme *80s*. Dans le tableau 4.4, nous montrons les valeurs typiques que nous donne les boosters, à titre indicatif.

TAB. 4.4 – Prédiction moyenne par chanson

Boosters	Positive (10 artists)	Ignore (100 songs)	Negative (200 songs)
main genres (rock, pop, Hip-Hop, metal, jazz, dance, Classical, country, blues, reggae)	0.540	0.528	0.463
instruments (piano, guitar, saxophone, trumpet)	0.538	0.507	0.470
mood (happy, sad, romantic, Mellow)	0.532	0.511	0.463
all	0.536	0.508	0.466

Plus les nombres tendent vers 0 ou 1, plus la décision du classifieur est claire. Il est cependant difficile de dire si ce résultat est satisfaisant ou non.

4.6 Analyse des attributs (features)

Un intérêt du boosting, lorsque l'on utilise de simple weak learners, est qu'il effectue une sélection des features implicite. En clair, comme chaque weak learner ne voit qu'un type de valeur (provenant des coefficients d'autocorrelations, du spectrogramme ou des MFCC) et qu'à chaque itération on choisit le meilleur weak learner possible, on peut faire un lien entre l'utilité d'un feature et un tag donné. Par exemple, on peut sommer les poids des weaks learners utilisant une valeur provenant des MFCC. On compare ensuite cela avec la somme des poids des weak learners utilisant les coefficients d'autocorrelation et le spectrogramme.

Pour obtenir une intuition sur l'apprentissage, nous avons additionné les poids des weak learners par attribut pour les 360 tags appris. A la figure 4.3, nous montrons les tags qui ont obtenu le poids le plus élevé pour chacun des trois features. Il est intéressant de voir que les tags utilisant le plus les coeffecicients d'autocorrelation sont de type *électronique* et *dance* où l'on peut s'attendre à ce que le rythme soit très caractéristique. Tel qu'expliqué à la section 4.2, les coefficients d'autocorrelation caractérisent bien le rythme, cela est cohérent. De même, les MFCC qui caractérisent le timbre sont utilisés le plus par des tags de type *instrument*. Enfin, on retrouve beaucoup de tags de type *metal* dans ceux qui utilisent le plus le spectrogramme. Une interprétation est la présence de hautes frquences provenant des guitares électriques et de la distorsion, et ces fréquences sont clairement visibles dans le spectrogramme.

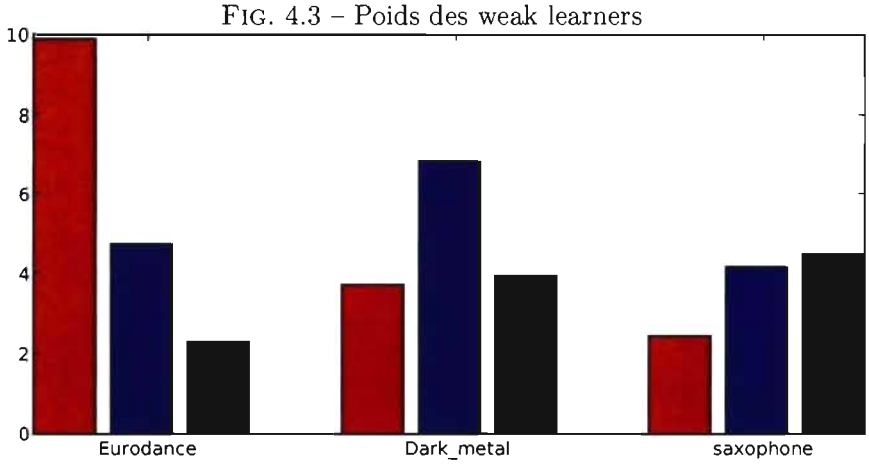
4.7 Approches concurrentes

L'étiquetage automatique est un sujet très actuel pour la communauté MIR, et plusieurs chercheurs travaillent activement dessus. Nous les mentionnons ici, et non au chapitre 2 car la plupart de leurs travaux ne sont pas encore publiés.

L'équipe la plus avancée est celle de UCSD[2] avec plusieurs publications disponibles sur leur page web. Ce sont eux qui ont développé la base de donnée CAL500 (section 3.6) et l'un des jeux en ligne (section 3.7 et figure 3.1). Leur algorithme est une mixture de gaussienne hiérarchique [TBTL08]. Dans notre article [BMEML08], F. Maillet tente une

[2] `http://cosmal.ucsd.edu/cal/projects/AnnRet/AnnRet.php`

FIG. 4.3 – Poids des weak learners



Somme des poids des weak learners sur différents tags pour les 3 types de features : 1) coefficients d'autocorrélation 2) constant-Q spectrogramme et 3) MFCC.

comparaison entre leur algorithme et le nôtre sur CAL500. Malheureusement, la conclusion principale en est que CAL500 est une trop petite base de données pour comparer nos méthodes.

Ensuite, M. Mandel travaille sur les données de son jeu en ligne [ME07], et à mis certains résultats préliminaires sur son blog [Man]. Un article sur sa méthode devrait paraître dans la même édition spéciale du *Journal of New Music Research*[3] que le nôtre [BMEML08].

Edith L.M. Law[4] de CMU travaille également sur les données de son jeu [LvADC07]. Aux dernières nouvelles, elle devrait faire une comparaison entre son modèle, celui de UCSD, et le nôtre.

E. Ravelli[5] a manifesté son intérêt pour le sujet suite à notre article à ISMIR [EBML07], mais nous ne savons pas s'il continue activement cette recherche.

Enfin, quelques articles sont sortis dans les semaines précédent la remise de ce mémoire [KSE08, Law08, ME08, TTKV08], mais dans tous les cas les travaux qui y sont rapportés sont fait sur de petites bases de données qui ne sont en rien comparables à celles que nous utilisons et qui proviennent de Last.fm

[3] http://www.tandf.co.uk/journals/titles/09298215.asp
[4] http://www.cs.cmu.edu/~elaw/
[5] http://www.emmanuel-ravelli.com/

CHAPITRE 5

SIMILARITÉ

Dans ce chapitre nous appliquons notre modèle d'étiquetage automatique à la similarité musicale. Nous justifions tout d'abord cette application, en montrant qu'elle nous permet d'évaluer notre modèle tout en étant une bonne approximation d'un sytème de recommendation musicale. Ensuite, à la section 5.2, nous expliquons comment nous avons construit notre similarité de référence. Une fois fait, nous créons notre propre mesure de similarité à partir de nos étiquettes (ou *autotags*). Enfin, nous la comparons avec d'autres mesures et discutons des résultats.

5.1 Justification

Nous justifions ici l'application de notre modèle d'étiquetage automatique à la similarité musicale. Tout d'abord, cela permet de mesurer la qualité de notre approche. Ensuite, l'un de nos buts initiaux est la recommendation de musique, et cela implique une bonne mesure de similarité.

5.1.1 Evaluation du modèle

Nous avons vu nos algorithmes d'étiquetage automatique au chapitre 4. Le problème qui se pose maintenant est l'évaluation de notre système. L'erreur de classification ne représente pas une mesure fiable. Certains artistes non *rock* peuvent avoir certaines chansons qui sonnent *rock* ou encore l'artiste peut avoir été mal étiqueté. De la même façon nous jugeons peu fiables dans notre cas les mesures typiques en recherche d'information, par exemple *precision* et *recall* telles qu'utilisées par Turnbull et al. [TBTL08] ou Mandel [Man] sur des données plus propres, à savoir CAL500 (section 3.6) et un jeu en ligne (section 3.7) respectivement.

En appliquant les autotags à une tâche telle que la similarité et en les comparant à la performance des tags de Last.fm obtenus par AudioScrobbler [Aud], nous pouvons comparer la qualité relative des deux. Cette façon de procéder ne nous fait pas dépendre de la qualité parfois douteuse des tags de Last.fm puisqu'ils ne seront pas la vérité de

référence.

5.1.2 Recommandation

L'un de nos buts est de faire de la recommandation de musique. Le principe le plus répendu, par exemple à Last.fm ou Pandora[1], est de recommander des artistes à partir d'un artiste de départ donné par l'utilisateur. La majorité des recommandeurs sur l'Internet offrent cette option[2]. Ces sytèmes commerciaux doivent prendre en considération de nombreux critères afin de satisfaire leurs utilisateurs. En voici quelques-uns tirés du travail de Herlocker et al. [HKTR04] sur le sujet :

- justesse (de l'anglais *accuracy*) de la recommandation : aspect fondamental, la recommandation doît être réllement similaire à l'artiste de départ, du moins du point de vue de l'utilisateur ;
- couverture du catalogue : un bon système de recommandation est capable de recommander toutes les chansons qu'il possède. Cela implique qu'il n'y a pas (ou peu) de *cold-start problem* (section 1.2) ;
- confiance de l'utilisateur dans le système : il faut parfois recommander des artistes que l'utilisateur connaît pour qu'il fasse ensuite confiance au système ;
- nouveauté des recommandations : toujours recommander The Beatles n'est pas une très bonne idée, l'utilisateur aurait pu y penser par lui-même ;
- surprise de l'utilisateur : cela est lié à la nouveauté, mais avec un aspect supplémentaire. Par exemple, recommander des chansons d'un artiste que l'utilisateur aime déjà peut lui faire découvrir quelque chose de nouveau mais lui recommander une chanson d'un artiste dont il n'a jamais entendu parler rajoute un effet de surprise, le système n'a pas pu utiliser d'algorithme trivial pour sa recommandation ;
- explication de la recommandation : les utilisateurs préfèrent ne pas voir le recommandeur comme une boîte noire (Herlocker et al. [HKR00]). Ils veulent par exemple une phrase leur disant : *nous vous recommandons cette chanson car l'on y retrouve de la guitare et une voix féminine avec un rythme blues.*

[1] `www.pandora.com`

[2] blogue de P. Lamere (entrée sur le sujet) : `http://blogs.sun.com/plamere/entry/what_s_the_point_of`

La plupart de ces concepts ne sont mesurables que sur un système commercial déployé et via l'analyse du comportement des utilisateurs. Le plus important demeure tout de même la notion de similarité, ce que l'on a appelé la justesse de la recommandation, et celle-ci peut se mesurer à partir d'une similarité de référence. Nous sommes conscients de ne toucher dans ce travail qu'à l'un des aspects d'un système de recommendation, celui de similarité, mais c'est le plus important, et les résultats sont mesurables dans un contexte de recherche académique (sans accès à un système commercial tel Last.fm) à partir de n'importe quelle similarité de référence que l'on possède.

5.2 Similarités de référence

Nous présentons la construction des deux similarités de référence qui nous servent à mesurer les similarités prédites par notre modèle. L'un de leur avantage est d'être construites à partir d'informations disponibles gratuitement sur l'Internet et elles peuvent donc être reproduite pour fins de comparaison. Il faut noter que le travail présenté ici est principalement l'oeuvre de Paul Lamere et Stephen Green de Sun Microsystems[3]. Le programme utilisé pour construire la première similarité est en train de devenir *open source*[4].

La première similarité est basée sur les habitudes d'écoute des utilisateurs de Last.fm. Si un nombre significatif de gens écoutent à la fois les artistes A et B, nous considérons ces deux artistes similaires. Évidemment, certains utilisateurs écoutent plus de musique que d'autres, et certains artistes sont plus populaires que d'autres. Les moteurs de recherche pour le texte (MRT) font face à un problème similaire. Ils veulent trouver les documents qui ont des mots similaires, mais sans que les mots fréquemment utilisés (tel *chat*) prennent toute l'attentation aux dépends des mots rares (tel *prestidigitation*). De plus, on ne veut pas que les documents plus longs soit systématiquement considérés plus similaires. Les MRT assignent un poids à chaque mot dans le document et ce poids se veut une représentation de l'importance d'un mot pour ce document. De nombreuses techniques de pondération ont été développées (voir Zobel et Moffat [ZM98] pour différents exemples), mais le plus populaire est le *term frequency-inverse document frequency*, ou TF×IDF.

[3] `http://research.sun.com/`

[4] `http://blogs.sun.com/searchguy/entry/minion_an_open_source_search1`

TF×IDF attribue un poids élevé aux mots qui apparaissent fréquemment dans un document et rarement dans les autres. L'idée fondamentale est que ces mots aux poids élevés sont les plus représentatifs de ce document et les plus discriminants pour ce documant par rapport à l'ensemble des documents. D'habitude, les poids associés à un document sont traités comme un vecteur dont la norme est normalisée à 1. Avec Last.fm, on peut considérer un artiste comme étant un "document" dont les "mots" sont les utilisateurs qui ont écoutés cet artiste. Le poids par TF×IDF pour un utilisateur pour un artiste prend en considération la popularité globale de l'artiste et s'assure qu'un utilisateur qui a écouté plus d'artistes ne domine pas nécessairement les autres. La similarité résultante nous semble raisonnable, et ne semble pas trop biaisée envers les artistes populaires.

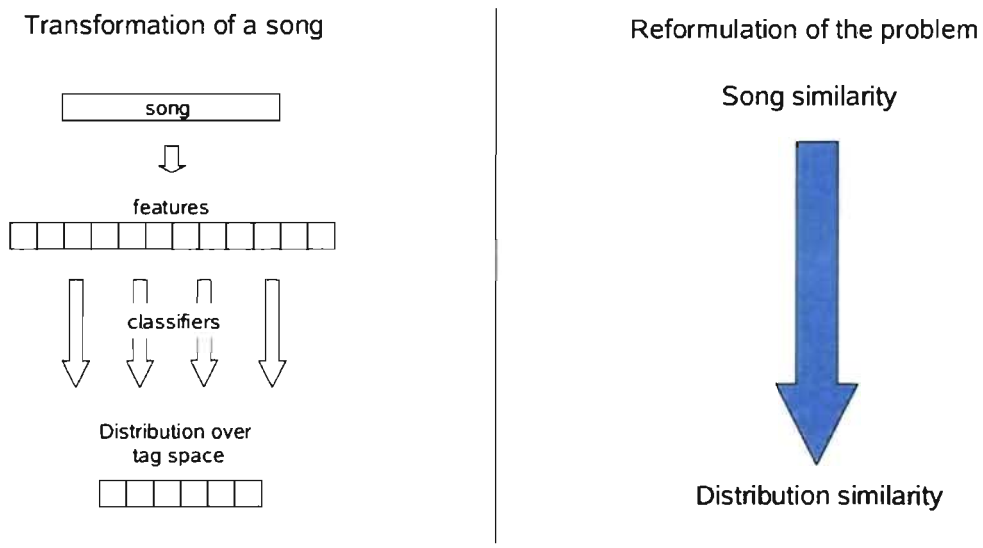
Nous utilisons également une deuxième similarité de référence pour être sûrs de ne pas sur-apprendre (de l'anglais *overfit*) les données de Last.fm. AMG (décrit à la section 3.3) contient notemment des listes d'artistes similaires. Nous utilisons une idée d'Ellis et al. [EWBL02] basée sur la distance d'Erdös. Si un artiste $A1$ est similaire à un artiste $A2$, ils ont une distance de 1. Si $A3$ est similaire à $A2$, $A1$ et $A3$ ont une distance de 2 et ainsi de suite. Vue d'une autre façon, la distance est le nombre de pas nécessaires pour aller d'un artiste à un autre en se déplaçant de listes en listes. La similarité s'obtient en prenant l'inverse de la distance. Nous avons chercher les artistes similaires de 4672 artistes sur AMG pour créer notre similarité de référence.

5.3 Expériences

Nous utilisons les autotags pour évaluer la similarité entre artistes. L'idée est résumée par la figure 5.1 : nous transformons le problème en une comparaison de vecteurs, ce qui est mathématiquement aisé. Nous présentons ici la méthode qui nous crée une similarité dérivée des autotags et nous comparons cette similarité avec une similarité dérivée des tags réels sur Last.fm. Pour ces expériences, nous utilisons les 360 boosters appris par FilterBoost.

Nous créons les autotags pour un artiste en appliquant nos boosters à tous les segments de 5s de toutes les chansons de cet artiste. On obtient une seule valeure en prenant la médiane. La moyenne semble donner des résultats similaires, mais nous nous méfions du phénomène suivant : si un groupe métal met une intro classique dans ses chansons,

FIG. 5.1 – Similarité



Reformulation de la similarité.

nous ne voulons pas qu'une valeur très classique pour quelques segments pousse tout l'artiste à être considéré classique. La médiane évite facilement ce problème.

Nous utilisons la similarité par cosinus (de l'anglais *cosine similarity*) pour évaluer la ressemblance entre les vecteurs. Celle-ci est définie par $s_{cos}(A_1, A_2) = A_1 * A_2 / (||A_1|| \, ||A_2||)$. Une autre possibilité est la distance euclidienne, dans nos expériences préliminaires, la différence n'est pas très significative. Nous discutons d'autres possiblités à la section 6.2.4.

Nous obtenons une matrice de similarité carrée $N \times N$, avec N le nombre d'artistes. Pour nos expériences, $N = 1000$, nous nous sommes restreints aux artistes ayant le plus de chansons dans notre base de données et dont nous avions un minimum de tags sur Last.fm. Pour comparer ces matrices, nous itérons sur chaque artiste en les considérant à tour de rôle artiste cible. À chaque itération, nous générons une liste ordonnée d'artistes, l'ordre dépendant de la distance avec l'artiste cible (le plus proche au sommet, puis par ordre décroissant de similarité). Nous pouvons évidemment faire cela avec notre similarité générée par les autotags, celle par les vrais tags de Last.fm, et la similarité de référence utilisée. Nous utilisons ensuite les mesures de comparaison de listes ordonnées vues à la section 2.5.2. Nous obtenons une valeur en prenant la moyenne sur les N artistes. Les

résultats sont donnés dans le tableau 5.1.

TAB. 5.1 – Résultats en similarité

Groundtruth	Model	TopN 10	Kendall 50	TopBucket 20
Last.fm	social tags	0.437	−0.057	42.98%
	autotags	0.140	−0.381	18.5%
	random	0.006	−0.626	2.0%
AMG	social tags	0.234	−0.287	26.8%
	autotags	0.104	−0.445	14.2%
	random	0.006	−0.626	2.0%

Performance comparée aux similarités de références Last.fm (haut) and AMG (bas).

Nous concluons, au regard de ces résultats, que bien qu'inférieur en qualité aux tags réels de Last.fm, les autotags ont tout de même capturé une partie de leur information. Le fait que la similarité induite par les autotags soit nettement supérieure à une similarité aléatoire nous confirme qu'il y a quelque chose à apprendre, et que notre algorithme y parvient partiellement.

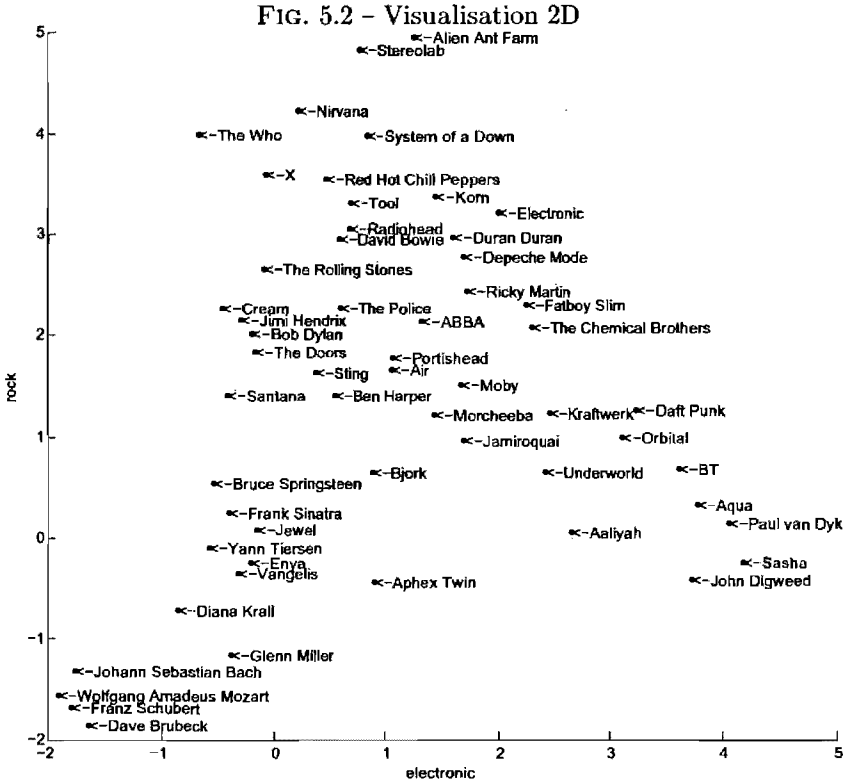
Nous mentionnons une seconde expérience, effectuée dans [ELBMG08] avec les 60 tags appris par AdaBoost. Le fait de mélanger la similarité induite des autotags avec celle induite des tags de Last.fm améliore les deux mesures de similarité. Nous pouvons donc conclure que les autotags sont même capable d'aider les tags de Last.fm, sans doute en les complétant pour les artistes peu étiquetés.

5.4 Visualisation

Créer une mesure de similarité nous permet de développer différents outils de visualisation de la musique. Par exemple, à la figure 5.2, nous montrons la position de certains artistes placés selon leurs composantes *rock* et *electronic* telles que prédites par nos autotags. La valeur pour un artiste est la médiane des valeurs pour chaque segment des chansons de cet artiste. Une évaluation subjective du résultat est encourageante. En particulier, les artistes classiques et jazz se retrouvent en bas à gauche du graphe correspondant à une faible composante *rock* et *electronic*.

Une autre idée, utilisée depuis longtemps par Paul Lamere dans ses démonstrations, est de "marcher" d'un artiste à un autre dans l'espace de similarité. La figure 5.3 en donne

FIG. 5.2 – Visualisation 2D

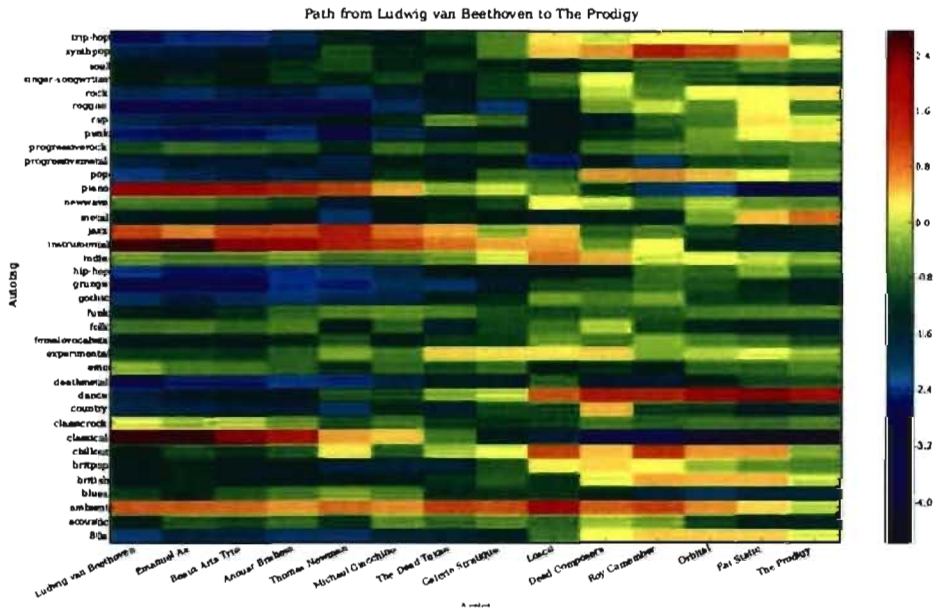


Représentation 2D d'artistes selon leur composante rock et électronique.

un exemple avec un trajet allant de Beethoven à The Prodigy. L'espace de similarité est projeté en deux dimensions par l'algorithme ISOMAP (décrit notamment dans Bishop [Bis06]) puis un algorithme de plus court chemin est utilisé pour trouver le trajet. A noter que l'on pourrait directement utiliser l'algorithme de plus court chemin dans l'espace à haute dimension, mais projeter les données en deux dimensions est plus intuitif.

Dans la figure 5.3, nous montrons les valeurs d'autotags pour les différents artistes du trajet ce qui permet de voir certaines transitions. Par exemple, la valeur de *classical* diminue de façon constante et la valeur de *dance* augmente.

FIG. 5.3 – Visualisation de chemins musicaux



Chemin de Beethoven à The Prodigy dans l'espace de similarité et évolution des autotags d'un artiste à l'autre.

CHAPITRE 6

CONCLUSION

Dans cette section nous revenons tout d'abord sur les résultats de nos expériences afin de se questionner sur nos méthodes d'évaluation, en particulier de l'utilité d'appliquer notre modèle à la similarité entre artistes. Ensuite, nous discuterons de différents aspect de notre modèle qui peuvent être améliorés, et quels sont les pistes à suivre les plus intéressantes selon nous.

6.1 Discussion des résultats

L'un des défis dans ce travail était de travailler avec une grande base de données, mais bruitée. Les tags de Last.fm sont mal répartis sur l'ensemble des artistes, inconsistants et parfois volontairement mal utilisés. Nous pensons que leur nombre compense pour le bruit dans l'apprentissage mais que l'on ne peut pas s'en servir pour valider l'apprentissage. Nous ne portons pas grand intérêt aux ensembles de tests (Table 4.2) qui sont pourtant l'élément de base pour mesurer la performance en apprentissage statistique. Nous avons senti le besoin d'utiliser ces autotags dans une tâche annexe, la similarité entre artistes, afin de vérifier leur qualité.

Nous l'avons vu, nous montrons ainsi le mérite de nos *autotags*. Mais cette méthode pose problème lorsque l'on veut se comparer à d'autre méthodes. Il semble acceptable de penser que de bons autotags impliqueront une bonne similarité, mais est-ce que les meilleurs autotags impliqueront la meilleure similarité ? Il n'est pas évident, si nous reproduisons les *hierarchical gaussian mixtures* de UCSD [TBTL08], que nous puissions décider de nos qualités relatives en passant par la similarité.

Nous voulons remettre en perspective l'évaluation de notre modèle. L'un de nos buts était la recommandation et créer une similarité entre artistes est un passage obligé pour y parvenir. De plus, au début de ces travaux, il n'y avait pratiquement aucun travail de référence sur l'étiquetage automatique en musique. La comparaison d'algorithmes n'était donc pas une priorité car il fallait d'abord montrer la faisabilité d'une telle tâche, mais à mesure que de nouveaux modèles vont apparaître (section 4.7), il sera important pour

la communauté de définir des expériences de comparaison. Par exemple, la publication de la base de donnée CAL500 (section 3.6) est un premier pas dans cette direction. Malheureusement, telle qu'étudiée dans [BMEML08], elle contient trop peu d'exemples pour être véritablement valide, mais l'idée de tester nos modèles sur des datasets non-bruités et faits par des experts semble la première chose à faire. Il faut savoir que certaines compagnies ont construit ce genre de données, la plus connue étant Pandora[1]. Il faut donc espérer que l'une de ces compagnies décide d'encourager la recherche en publiant sa propre base de données d'étiquettes.

6.2 Travaux futurs

Nous présentons quelques avenues de recherche pertinentes à ce travail, en mettant l'accent sur celles que nous avons commencé à étudier.

6.2.1 Attributs audio

L'une des directions de recherche active en apprentissage statistique est les algorithme *parcimonieux* (*sparse* en anglais), comme dans *sparse features* ou *sparse representation.* L'intuition en est que si la même information est contenue dans deux représentations, les algorithmes d'apprentissage statistique fonctionnent normalement mieux avec celle la plus compacte. L'idée des *sparse features* est donc de comprimer l'information dans quelques valeurs actives à la fois. Par exemple, une idée naïve pour rendre un spectrogramme plus parcimonieux serait de mettre à 0 les valeurs inférieures à un certain seuil. L'information principale est sans doute contenue dans les plus grandes valeurs et on n'aurait donc compacté l'information.

Beaucoup de travail a été fait sur les représentations parcimonieuses pour les images, voir par exemple Lee et al. [LEN08]. De même, Smith et Lewicki [SL05] proposent une méthode pour produire des *sparse features* pour les images et la musique. Nous étudions ces représentations dans Manzagol et al. [MBME08]. En particulier, ces représentations peuvent être apprises sur un corpus de son particulier afin de mieux le représenter. On peut donc espérer développer des *features* spécifiques à certains genres musicaux, voir à

[1] www.pandora.com

certains tags. Cela ouvre la voie à de nombreuses améliorations de notre modèle en terme de performance simplement en ayant de meilleures données d'entrée.

De façon plus générale, il faudrait faire une étude extensive des meilleurs *features* pour l'étiquetage automatique, voir les meilleurs *features* pour chacun des tags. Ceux que nous utilisons dans ce travail sont classiques et ont fait leur preuve avec le temps mais il faudrait valider ce choix de façon plus rigoureuse.

6.2.2 Échantillonage pour FilterBoost

La sélection des exemples dans FilterBoost est simpliste, l'oracle allant chercher les exemples de façon aléatoire. Pourtant, nous savons que ce qui est utile au boosting sont les exemples difficiles et nous avons une notion de similarité dans notre base de données. En effet, la musique est organisée par artiste et album, et ceux-ci ont normalement un son similaire. En poursuivant ce raisonnement, si l'on tombe sur un exemple difficile, il est probable que l'on en trouvera d'autres du même artiste, et c'est une piste à suivre pour l'oracle.

Dans Kégl et al. [KBME08], nous adaptons ce principe pour en faire un algorithme de type Metropolis-Hastings (une description complète de cet algorithme est offerte par Bishop [Bis06]). C'est une simple formalisation du concept d'échantillonage intelligent où le prochain point sélectionné dépend du point courant. Nous remplaçons l'oracle et le filtre de FilterBoost par cet "oracle intelligent" et cela semble aider l'apprentissage.

FilterBoost est un algorithme récent, nous nous attendons à voir de nombreuses versions apparaître dans un avenir proche, et il faudra surveiller celles qui ont le potentiel d'améliorer nos performances.

6.2.3 Sélection des étiquettes

Sur l'ensemble des étiquettes que l'on essaie d'apprendre, certains ont peu de sens (par exemple *favorites*) ou du moins ne peuvent êtres apprises à partir de features audio. Il serait donc intéressant de filtrer ces tags et ne garder que ceux qui peuvent être utilisés par notre algorithme. Torres et al. [TTBL07] proposent une technique pour déterminer quels mots représentent réellement un concept musical et donc devrait pouvoir être appris. De notre côté, nous pouvons nous demander si, simplement à partir de la performance

de notre booster, nous pouvons déterminer la qualité d'un tag. L'idée peut se résumer ainsi, "si l'on apprend mal un tag, c'est que c'est un mauvais tag". Ce raisonnement est circulaire, mais fait du sens lorsque l'on voit que notre algorithme fonctionne sans difficulté sur des tags comme *jazz* mais ne fait pas beaucoup mieux que l'aléatoire sur d'autres.

De façon plus objective, nous pourrions utiliser le même genre de structure qu'à la section 6.2.2, à savoir que la musique est composé d'artistes, d'albums et de chansons. On peut s'attendre à ce que ceux-ci soit un minimum similaire à l'interne, d'une chanson à l'autre d'un même album par exemple. En conséquence, un booster qui étiquetterait de manière totalement différent les segments d'une même chanson pourrait nous indiquer que ce tag (du moins ce booster) est peu fiable. On pourrait ainsi réduire et améliorer le vocabulaire (ensemble des tags) utilisé.

6.2.4 Distances et comparaison de distributions

Pour comparer les distributions de mots, nous avons utilisé la similarité du cosinus et essayé la distance euclidienne. De nombreuses autres distances auraient le potentiel de mieux comparer ces distributions. On peut penser aux $p - normes$, à savoir $||A - B||^p$, dont la distance euclidienne est le cas particulier $p = 2$. Une autre mesure populaire est la divergence de Kullback-Leibler (voir par exemple Bishop [Bis06]). Celle-ci a d'ailleurs été utilisé pour ce problème par Barrington et al. [BTTL]. Il faudrait vérifier empiriquement laquelle s'applique le mieux au problème présent.

Une idée plus intéressante serait d'apprendre la distance. Par exemple, un réseau de neurones pourrait prendre deux distributions en entrée et sortir une valeur reliée à leur similarité. Les exemples d'apprentissage pourraient être tirés d'un sous-ensemble d'une similarité de référence (voir section 5.2). Ainsi, ce réseau de neurones pourrait apprendre les relations entre les tags, bien que ceux-ci soient appris indépendamment. La valeur des autotags *hip hop* et *hip-hop* pourraient se renforcer mutuellement car le réseau apprendrait que ce sont des synonymes.

Dans des expériences préliminaires, l'apprentissage d'une distance (ou d'une similarité) semblait plus difficile que prévu et ne donnait pas de meilleurs résultats que la similarité du cosinus. Nous pensons cependant que le meilleur système prendra avantage de l'interdépendance des tags et ce peut importe qu'ils soient appris indépendamment ou

non au départ. Une distance apprise est donc très prometteuse.

BIBLIOGRAPHIE

[Aka74] H. Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, AC-19(6) :716–723, 1974.

[AM05] P. Ahrendt and A. Meng. Music genre classification using the multivariate ar feature integration model. Extended Abstract, 2005. MIREX genre classification contest (`www.music-ir.org/evaluation/mirex-results`).

[AP03] J.J. Aucouturier and F. Pachet. Representing musical genre : A state of the art. *Journal of New Music Research*, 32(1) :83–93, 2003.

[Aud] Audioscrobbler. Web Services described at `http://www.audioscrobbler.net/data/webservices/`.

[Bas] C. Bastuck. An extensible and multiperspective approach for music similarity. *Music Information Retrieval Evaluation Exchange (MIREX)*, Vienna, 2007, available at `http://www.music-ir.org/mirex/2007/index.php/Audio_Music_Similarity_and_Retrieval_Results`.

[BCE05] J. Bergstra, N. Casagrande, and D. Eck. Genre classification : Timbre- and rhythm-based multiresolution audio classification. MIREX genre classification contest, 2005.

[BCE+06] J. Bergstra, N. Casagrande, D. Erhan, D. Eck, and B. Kégl. Aggregate features and AdaBoost for music classification. *Machine Learning*, 65(2-3) :473–484, 2006.

[Ber06] J. Bergstra. Algorithms for classifying recorded music by genre. Master's thesis, Université de Montréal, 2006.

[Bis06] C. Bishop. *Pattern Recognition and Machine Learning.* Springer Verlag, 2006.

[BLE06] J. Bergstra, A. Lacoste, and D. Eck. Predicting genre labels for artists using freedb. In *Proceedings of the 7th International Conference on Music Information Retrieval (ISMIR 2006)*, 2006.

[BLEW03] A. Berenzweig, B. Logan, D. Ellis, and B. Whitman. A large-scale evaluation of acoustic and subjective music similarity measures. In *Proceedings of*

the 4th International Conference on Music Information Retrieval (ISMIR 2003), 2003.

[BMEML08] T. Bertin-Mahieux, D. Eck, F. Maillet, and P. Lamere. Autotagger : a model for predicting social tags from acoustic features on large music databases. *Journal of New Music Research*, 2008. (to appear).

[BS08] J. K. Bradley and R. Schapire. Filterboost : Regression and classification on large datasets. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*. MIT Press, Cambridge, MA, 2008.

[BTTL] L. Barrington, D. Turnbull, D. Torres, and G. Lanckriet. Semantic similarity for music retrieval. *Music Information Retrieval Evaluation Exchange (MIREX)*, Vienna, 2007, available at `http://www.music-ir.org/mirex/2007/index.php/Audio_Music_Similarity_and_Retrieval_Results`.

[Cas05] N. Casagrande. Automatic music classification using boosting algorithms and auditory features. Master's thesis, Université de Montréal, 2005.

[CCH06] Ó. Celma, P. Cano, and P. Herrera. Search sounds : An audio crawler focused on weblogs. In *ISMIR*, pages 365–366, 2006.

[DHS00] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley-Interscience Publication, 2000.

[EBML07] D. Eck, T. Bertin-Mahieux, and P. Lamere. Autotagging music using supervised machine learning. In *Proceedings of the 8th International Conference on Music Information Retrieval (ISMIR 2007)*, 2007.

[Eck07] D. Eck. Beat tracking using an autocorrelation phase matrix. In *Proceedings of the 2007 International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1313–1316. IEEE Signal Processing Society, 2007.

[ELBMG08] D. Eck, P. Lamere, T. Bertin-Mahieux, and S. Green. Automatic generation of social tags for music recommendation. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 385–392. MIT Press, Cambridge, MA, 2008.

[EWBL02] D. Ellis, B. Whitman, A. Berenzweig, and S. Lawrence. The quest for ground truth in musical artist similarity. In *Proceedings of the 3th International Conference on Music Information Retrieval (ISMIR 2002)*, 2002.

[FHT98] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression : a statistical view of boosting. Technical report, Stanford University, 1998.

[FS96] Y. Freund and R.E. Shapire. Experiments with a new boosting algorithm. In *Machine Learning : Proceedings of the Thirteenth International Conference*, pages 148–156, 1996.

[GM99] B. Gold and N. Morgan. *Speech and Audio Signal Processing : Processing and Perception of Speech and Music.* John Wiley & Sons, Inc., New York, NY, USA, 1999.

[GT] M. Guy and E. Tonkin. Tidying up tags. *D-Lib Magazine.* online article : www.dlib.org/dlib/january06/guy/01guy.html.

[HKR00] J. L. Herlocker, J. A. Konstan, and J. T. Riedl. Explaining collaborative filtering recommendations. In *Computer Supported Cooperative Work*, pages 241–250, 2000.

[HKTR04] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.*, 22(1) :5–53, 2004.

[KBME08] B. Kégl, T. Bertin-Mahieux, and D. Eck. Metropolis-hastings sampling in a filterboost music classifier. In *International Workshop on Machine Learning and Music (MML 2008).* 2008. (submitted).

[KPW04] P. Knees, E. Pampalk, and G. Widmer. Artist classification with web-based data. In *Proceedings of the 5th International Conference on Music Information Retrieval (ISMIR 2004)*, 2004.

[KSE08] Y. Kim, E. Schmidt, and L. Emelle. Moodswings : a collaborative game for music mood label collection. In *Proceedings of the 9th International Conference on Music Information Retrieval (ISMIR 2008)*, 2008.

[Lam08] P. Lamere. Semantic tagging and music information retrieval. *Journal of New Music Research*, 2008. (to appear).

[Law08] E. Law. The problem of accuracy as an evaluation criterion. In *ICML Workshop on Evaluation Methods in Machine Learning*, 2008.

[LEN08] H. Lee, C. Ekanadham, and A. Ng. Sparse deep belief net model for visual area v2. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*. MIT Press, Cambridge, MA, 2008.

[LKS+98] T. Lambrou, P. Kudumakis, R. Speller, M. Sandler, and A. Linney. Classification of audio signals using statistical features on time and wavelet tranform domains. In *Proc. Int. Conf. Acoustic, Speech, and Signal Processing (ICASSP-98)*, volume 6, pages 3621–3624, 1998.

[LOL03] Tao Li, Mitsunori Ogihara, and Qi Li. A comparative study on content-based music genre classification. In *SIGIR '03 : Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 282–289, New York, NY, USA, 2003. ACM Press.

[LvADC07] E. Law, A. v. Ahn, R. Dannenberg, and M. Crawford. Tagatune : a game for music and sound annotation. In *Proceedings of the 8th International Conference on Music Information Retrieval (ISMIR 2007)*, 2007.

[Man] M. Mandel. blog at http ://blog.mr-pc.org/2008/03/04/autotagging/.

[MBME08] P.-A. Manzagol, T. Bertin-Mahieux, and D. Eck. On the use of sparse time relative auditory codes for music. In *9th International Conference on Music Information Retrieval (ISMIR 2008)*. 2008. (submitted).

[ME05] M. Mandel and D. Ellis. Song-level features and support vector machines for music classification. In T. Crawford and M. Sandler, editors, *Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR 2005)*, 2005.

[ME07] M. I. Mandel and D. P. W. Ellis. A web-based game for collecting music metadata. In *Proceedings of the 8th International Conference on Music Information Retrieval (ISMIR 2007)*, 2007.

[ME08] M. Mandel and D. Ellis. Multiple-instance learning for music information retrieval. In *Proceedings of the 9th International Conference on Music Information Retrieval (ISMIR 2008)*, 2008.

[MF05] C. McKay and H. Fujinaga. Automatic music classification and the importance of instrument identification. In *Proceedings of the Conference on Interdisciplinary Musicology (CIM05)*, Montreal, Canada, 2005.

[MF06] C. McKay and I. Fujinaga. Musical genre classification : is it worth pursuing and how can it be. In *Proceedings of the 7th International Conference on Music Information Retrieval (ISMIR 2006)*, 2006.

[Mir] MIREX : Music Information Retrieval Evaluation eXchange, information at http ://www.music-ir.org/mirex2007/index.php/.

[MNBD06] C. Marlow, M. Naaman, D. Boyd, and M. Davis. Position paper, tagging, taxonomy, flickr, article, toread. In *Collaborative Web Tagging Workshop at WWW2006, Edinburgh, Scotland*, May 2006.

[Mul] Multiboost, a pure C++ implementation of AdaBoost.MH by N. Casagrande available at `http://www.iro.umontreal.ca/~casagran/`.

[PC] F. Pachet and D. Cazaly. A taxonomy of musical genres. In *RIAO*.

[PG99] D. Perrott and R. Gjerdingen. Scanning the dial : An exploration of factors in the identification of usical style. In *Society for Music Perception and Cognition Conference (SMPC)*, Evanston, IL, 1999.

[PS] T. Pohle and D. Schnitzer. Striving for an improved audio similarity measure. *Music Information Retrieval Evaluation Exchange (MIREX)*, Vienna, 2007, available at `http://www.music-ir.org/mirex/2007/index.php/Audio_Music_Similarity_and_Retrieval_Results`.

[SL05] E. Smith and M. S. Lewicki. Efficient coding of time-relative structure using spikes. *Neural Computation*, 17(1) :19–45, 2005.

[SS99] R. E. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3) :297–336, 1999.

[TBTL07] D. Turnbull, L. Barrington, D. Torres, and G. Lanckriet. Towards musical query-by-semantic-description using the cal500 data set. In *SIGIR '07 :*

Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval, pages 439–446, New York, NY, USA, 2007. ACM.

[TBTL08] D. Turnbull, L. Barrington, D. Torres, and G. Lanckriet. Semantic annotation and retrieval of music and sound effects. *IEEE Transactions on Audio, Speech & Language Processing*, 16(2), 2008.

[TC02] G. Tzanetakis and P. Cook. Musical genre classification of audio signals. *IEEE Transactions on Speech and Audio Processing*, 10(5) :293–302, Jul 2002.

[TLBL07] D. Turnbull, R. Liu, L. Barrington, and G. Lanckriet. A game-based approach for collecting semantic annotations of music. In *Proceedings of the 8th International Conference on Music Information Retrieval (ISMIR 2007)*, 2007.

[TTBL07] D. Torres, D. Turnbull, L. Barrington, and G. Lanckriet. Identifying words that are musically meaningful. In *Proceedings of the 8th International Conference on Music Information Retrieval (ISMIR 2007)*, 2007.

[TTKV08] K. Trohidis, G. Tsoumakas, G. Kalliris, and I. Vlahavas. Multi-label classification of music into emotions. In *Proceedings of the 9th International Conference on Music Information Retrieval (ISMIR 2008)*, 2008.

[Tza] G. Tzanetakis. Marsyas submissions to mirex 2007. *Music Information Retrieval Evaluation Exchange (MIREX)*, Vienna, 2007, available at `http://www.music-ir.org/mirex/2007/index.php/Audio_Music_Similarity_and_Retrieval_Results`.

[vAD04] L. von Ahn and L. Dabbish. 2004, labeling images with a computer game. In *CHI '04 : Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 319–326, New York, NY, USA, 2004. ACM Press.

[WC04] K. West and S. Cox. Features and classifiers for the automatic classification of musical audio signals. In *Proceedings of the 5th International Conference on Music Information Retrieval (ISMIR 2004)*, 2004.

[ZM98] J. Zobel and A. Moffat. Exploring the similarity space. *SIGIR Forum*, 32(1) :18–34, 1998.