DE L'INTERET DES FEATURES ANALYTIQUES : UNE ETUDE POUR LA CLASSIFICATION DE SONS DE PANDEIRO

Pierre Roy
Sony CSL
6, rue Amyot
75 005 Paris
roy@csl.sony.fr

François Pachet
Sony CSL Paris
6, rue Amyot
75 005 Paris
pachet@csl.sony.fr

Sergio Krakowski
IMPA
Rio de Janeiro
Brésil
skrako@gmail.com

RÉSUMÉ

L'analyse automatique de sons numérisés intéresse aujourd'hui de nombreuses applications de l'informatique musicale, de la classification de musique pour la gestion de catalogues à la musique interactive. Dans le cadre de la classification supervisée, nous présentons une méthode originale permettant d'améliorer les performances des systèmes standards. Cette méthode consiste à utiliser, en entrée des classifieurs, des *features* particulièrement bien adaptées au problème à résoudre, au contraire des systèmes traditionnels, qui utilisent soit des features « standards », telles que celles du *feature set* MPEG7-audio soit des features ad hoc trouvées manuellement. Pour identifier ces features, notre méthode explore un très grand espace de fonctions, appelées fonctions analytiques, obtenues par composition d'opérateurs mathématiques et du signal audio. Notre méthode permet de produire ces compositions fonctionnelles, de les évaluer, et de déterminer des feature sets adaptés au problème à résoudre. Nous présentons ici une étude sur l'analyse automatique de sons de pandeiro (tambourin brésilien). Deux problèmes d'identification de sons de pandeiro sont abordés : sons entiers et attaques. Nous évaluons le gain en précision de notre méthode sur ces deux problèmes, par rapport à l'approche standard.

1. FEATURES ACOUSTIQUES

La classification audio utilise le plus souvent l'approche dite *bag-of-frames* (voir (Sheirer and Slaney, 97) pour un exemple standard). Cette approche consiste à calculer des features générales et bien connues en traitement du signal audio, sur des *« frames »* successives d'un signal de départ, puis à les sélectionner (Peeters and Rodet, 02) et les combiner de manière aveugle (d'où le *« bag »*), pour créer un vecteur de données censé représenter les caractéristiques importantes du signal pour le problème de classification en cours. Ces données sont ensuite utilisées pour entraîner un classifieur, à partir d'une base de signaux préalablement étiquetés (base d'apprentissage). Enfin, on évalue la performance du classifieur ainsi obtenu en utilisant une seconde base (base de test) contenant des signaux étiquetés différents, pour éviter les problèmes d'*overfit*.

L'utilisation de ces features vise essentiellement à réduire la dimension du problème. En effet, le signal lui-même pourrait, en théorie, être utilisé directement en entrée d'un classifieur, mais il est en général de dimension trop élevée pour être utilisé tel quel, et par ailleurs sa représentation (temps/amplitude) est mal adaptée aux problèmes de classification perceptive. Une source de référence pour des features acoustiques est par exemple MPEG7-audio (Kim et al., 05) ou plus spécifiquement (Peeters, 04). Ces features sont de dimension moindre que le signal, et contiennent des informations statistiques, à la fois temporelles (e.g. Zero-crossing rate), spectrales (e.g. SpectralCentroid), ou plus « perceptives » (sharpness, relative loudness, etc.).

Si cette approche fonctionne de façon satisfaisante pour certains problèmes relativement simples (classification speech/musique (Sheirer and Slaney 97), classification en genre (Pampalk et al. 05 ; Tzanetakis and Cook, 02), classifications de sons d'orchestre (Peeters, 03) ou de sons percussifs (Herrera et al., 02 ; van Steelant et al.. 04), elle s'avère insuffisante dans les cas plus complexes, nécessitant alors une approche manuelle *ad hoc*, comme par exemple l'identification de sons de Tabla par (Gillet and Richard, 03). La raison que nous avançons dans ce travail est que les features générales utilisées, telles que celles de MPEG7, ne capturent pas nécessairement les caractéristiques perceptives pertinentes des signaux à identifier. Certains algorithmes d'apprentissage, comme les méthodes à noyaux (*kernel*) (Schölkopf and Smola, 02) ou les Support Vector Machines (Boser et al. 92 ; Shawe-Taylor and Cristianini, 2000), opèrent des transformations de l'espace des features avec le but d'améliorer la séparation inter classe. Néanmoins, la sophistication croissante des algorithmes de sélection de feature ou d'apprentissage ne peut pas compenser le manque d'information contenue dans le feature set initial.

Pour trouver de meilleures features, on peut s'inspirer de la manière dont un expert en traitement du signal identifie (ou plutôt, invente) des features ad hoc. On peut, par exemple, normaliser un signal, ajouter un filtre en amont d'une feature de l'énergie d'un signal pour améliorer la performance d'un classifieur, ou bien effectuer des pré ou post processing pour isoler les aspects pertinent du signal (Sheirer and Slaney 97), ou

encore définir des features arbitraires en utilisant des connaisances musicales (Peeters et al., 2000).

Nous proposons ici d'automatiser une partie de ce processus de création de feature ad hoc, à l'aide d'un algorithme d'exploration d'un très grand espace de fonctions du signal. Ces fonctions sont construites par composition – au sens de la composition mathématique de fonctions – d'opérateurs élémentaires. Elles sont dites *analytiques* car décrites par cette composition explicite (et non pas en extension, ou par un programme).

Cet article est structuré comme suit : dans la section 2 nous présentons le système (EDS) qui permet la création automatique et l'exploration d'un grand nombre de features analytiques. La section 3 est consacrée à un compte rendu d'expérience nous permettant de comparer notre approche à l'approche standard utilisant des features générales, sur deux problèmes d'identification de sons de pandeiro.

2. CREATION DE FEATURES ANALYTIQUES : LE SYSTEME EDS

EDS – Extractor Discovery System – est un projet - et un système - développé au laboratoire Sony CSL à Paris (Pachet et Zils, 04 ; Zils, 04) dont le but est d'étudier de manière expérimentale la notion de feature analytique pour les applications en traitement du signal audio.

Le système EDS permet d'explorer efficacement l'espace des features anaytiques sur tout problème de classification audio supervisée. Un problème de classification est déterminé par la donnée d'une ou plusieurs bases d'échantillons audio étiquetés (en général manuellement) par leur classe.

L'exploration de l'espace des features analytiques repose sur diverses méthodes de création de fonctions à partir d'opérateurs de base, considérés comme élémentaires. Ces deux aspects sont décrits dans les sections suivantes.

2.1. La bibliothèque d'opérateurs élémentaires

Le choix des opérateurs de base est évidemment arbitraire. Ceux-ci ont été sélectionnés de manière à permettre d'engendrer des fonctions signal ayant un niveau d'abstraction « raisonnable ». Ces opérateurs sont soit des fonctions mathématiques de base (e.g. multiplication d'un signal par un scalaire, valeur absolue, max), soit des opérateurs de traitement du signal de relativement bas niveau, tels que Fft, dB, Root-Mean-Square, SpectralSkewness, SpectralCentroid, etc. Cette bibliothèque comprend également des opérateurs spécialisés dans le traitement de la parole (comme *Pitch* ou *Ltas*, implémentés à l'aide de la bibliothèque Praat (http://www. Praat.org). Cette liste peut être bien sûr modifiée par l'utilisateur selon les types d'applications. Dans ce papier nous décrivons les résultats obtenus avec les 78 opérateurs de l'Annexe 1.

L'espace des fonctions obtenues par composition fonctionnelle arbitraire des opérateurs de base est a priori infini. On peut cependant évaluer le nombre de fonctions que l'on peut construire en limitant le nombre d'opérateurs de base qu'elles utilisent. Pour un nombre d'opérateur maximum de 5 on peut construire $2{,}5.10^9$ fonctions. En pratique, nous explorons l'espace des fonctions de taille inférieure à 10 opérateurs, dont la taille est estimée à 5.10^{20}. L'estimation de la taille de ces espaces est obtenue par une énumération exhaustive des constructions syntaxiquement correctes, et en discrétisant les paramètres scalaires des fonctions (par exemple on ne considère que 10 valeurs pour les fréquences de coupure des filtres, ce qui est sous-évalué). On observe que le nombre de fonctions croit exponentiellement avec la « longueur » maximum de ces fonctions. Voici quelques exemples typiques de fonctions engendrées:

```
Mean(Mfcc(Differentiation(x),5))
Median(Rms(Split(Normalize(x),32)))
```

La première fonction calcule la moyenne des 5 coefficients cepstraux de la dérivée du signal (représenté par *x*). La seconde calcule la valeur moyenne de l'énergie (*Rms*) des *frames* successives de 32 échantillons du signal (*Split*) préalablement normalisé.

La création des features est contrôlée par deux mécanismes :

1 – Le typage. Chaque opérateur élémentaire est typé par les dimensions physiques de ses arguments d'entrée et de sortie. Ces types sont utilisés pour éviter la construction de features n'ayant pas de sens syntaxique. Par exemple, l'opérateur *fft* prend en entrée des amplitudes en fonction du temps et retourne en sortie des amplitudes en fonction des fréquences. On peut donc produire *fft(HpFilter(x))*, mais pas, par exemple, *fft(max(x))*.

2 – Les heuristiques. Celles-ci apportent un surcroît de contrôle, en pénalisant certaines combinaisons, jugées peu prometteuses. Par exemple, le système évitera de construire des features comportant plusieurs fois de suite le même (genre) d'opérateurs, comme *fft(fft(fft(fft(x))))*.

En pratique, étendre cette bibliothèque avec un nouvel opérateur élémentaire implique essentiellement de définir pour chaque nouvel opérateur 1) les règles de typage correspondantes, et 2) définir éventuellement des heuristiques concernant ce nouvel opérateur.

2.2. La création de features analytiques

La création de features analytiques par composition fonctionnelle des opérateurs élémentaires est fondée sur

un algorithme de recherche génétique (Koza, 1992). La **Figure 5** montre le système en fonctionnement. Les étapes principales de l'algorithme sont les suivantes :

1 – Construction d'une famille (ou population) de features initiale, par composition aléatoire d'opérateurs élémentaires.

2 – Evaluation des features. Cette évaluation consiste à calculer la fonction sur tous les signaux de la base d'apprentissage, et à comparer ces résultats avec les étiquettes. La corrélation entre ces deux séries de valeurs constitue la *fitness* de la fonction. Le calcul de cette corrélation est décrit en section 2.3.

3 – Itérer en construisant la population suivante. Celle-ci est composée des meilleures features trouvées dans la poulation courante, ainsi que de nouvelles features obtenues par des transformations génétiques des features courantes.

Voici deux exemples de features (décrites précédemment) construites à l'étape 1:

```
(A) Mean(Mfcc(Differentiation(x),5))
(B) Median(Rms(Split(Normalize(x),32)))
```

Les transformations génétiques effectuées à l'étape 3 sont les suivantes :

Substitution
La substitution consiste à remplacer un des opérateurs apparaissant dans la feature par un autre, de type équivalent. Par exemple :

(A') `Max(Mfcc(Differentiation(x),5))`

est une substitution (`Max` remplace `Mean`) de (A)

Clonage
Le clonage est un cas particulier de substitution qui consiste à copier une feature en changeant la valeur d'un de ses paramètres numériques, e.g. :

(B') `Median(Rms(Split(Normalize(x),64)))`

est un clone de (B). Les valeurs de ces paramètres sont choisies dans un intervalle de valeurs « raisonnables », lui-même dépendant de l'opérateur et de la fréquence d'échantillonage.

Mutation
La mutation est une extension de la substitution à des ensembles d'opérateurs apparaissant dans la définition d'une feature, tout en respectant les règles de typage :

(A") `Mean(Chroma(Normalize(x)))`

est une mutation de (A) dans laquelle l'expression `Chroma (Normalize(x))` remplace `Mfcc (Differentiation (x),5).`

Croisement
Le croisement consiste à combiner deux features pour en fabriquer une troisième, en respectant les règles de typage. Par exemple :

(C) `Mean(Rms(Split(Normalize(x),32)))`
(C') `Median(Rms(Split(Differentiation(x)))`

sont des croisements entre (A) et (B).

Addition
L'addition est l'ajout d'un opérateur supplémentaire à une feature:

(B") `Abs(Median(Rms(Split(Normalize(x),32))))`

est une addition de (B).

2.3. Evaluation des features

Nous avons besoin d'un critère calculable qui mesure la « qualité » d'une feature, i.e. sa capacité à séparer entre eux les éléments de classes différentes. Il existe plusieurs critères envisageables pour évaluer cette propriété.

Le discriminant de Fisher (Fisher, 36) est souvent utilisé car il est rapide à calculer et il est fiable pour ce qui concerne les problèmes à deux classes. En revanche, il est notoirement mal adapté aux problèmes multi-classe, en particulier pour les distributions non connexes.

Pour améliorer l'évaluation d'une feature, nous choisissons plutôt d'évaluer la performance d'un classifieur construit à la volée avec cette feature unique. Chaque feature est ainsi utilisée pour entrainer et évaluer un classifieur sur les échantillons d'apprentissage. La performance de ce classifieur (plus précisément sa F-mesure (van Rijsbergen, 79) est, *in fine*, le critère d'évaluation utilisé dans EDS pour la feature. Cette mesure donne de bien meilleurs résultats que le discriminant de Fisher.

3. CLASSIFICATION DE SONS DE PANDEIRO

Le pandeiro est un type de tambourin utilisé notamment dans la musique traditionnelle brésilienne (samba, côco, capoeira, chôro). Comme c'est le cas pour de nombreux instruments de musique populaire, il n'existe pas de méthode officielle pour jouer le pandeiro. Le troisième auteur a proposé une méthode de notation systématique, reposant sur 6 classes de sons (illustrés Figure 1):

tung : le son le plus grave, dit aussi son ouvert.

ting : un son grave mais plus haut que le tung ; également dit ouvert.

pa : un son medium produit en frappant la membrane au millieu du pandeiro ; le **pa** est proche du slap de conga.

PA : le son medium plus fort, comme un *slap* de conga.

tr : un son aigu métallique, comme un trémolo.

tchi : une seule attaque, aiguë et métallique.

L'intérêt de disposer d'une analyse automatique des sons de pandeiro est de permettre une interaction en temps réel entre un pandeiriste et un système de synthèse musicale, sans délai perceptible (cet aspect fait l'objet de travaux en cours). Pour ce faire, nous nous sommes intéressés à deux problèmes d'analyse.

Figure 1**.** Les différents gestes produisant les six sons de base du Pandeiro.

Le premier consiste à identifier les six types de sons en entrainant le système avec des échantillons de sons entiers (chaque son dure environ 150 ms). Le second problème, beaucoup plus difficile mais plus « utile » dans notre contexte, est d'identifier les sons de pandeiro à partir des seules attaques (de l'ordre de 3ms, soit 128 samples à 44.100 Hz), de manière à permettre une interaction temps-réel avec un système informatique. A cet effet nous devons non seulement disposer d'un classifieur précis et robuste, mais aussi très rapide : idéalement, l'analyse de l'attaque devrait être réalisée en moins de 15 millisecondes.

3.1. Bases de données et échantillons disponibles

Sons entiers

Nous avons construit pour cette étude une base de 2388 échantillons de sons complets de pandeiro. Ces sons ont été enregistrés par Sergio Krakowski. Ils proviennent du même instrument, et ont été saisis à l'aide d'un micro Shure Beta 98 relié à une carte audio MOTU Traveller. Ces 2388 échantillons se répartissent de façon équilibrée : 398 échantillons par classe.

Attaques

Nous disposons aussi d'une base de 2184 échantillons d'attaques de pandeiro, répartis aussi de manière équilibrée dans les six catégories à reconnaître (364 par classe).

3.2. Protocole expérimental : les bases

Afin de comparer l'efficacité des features analytiques, nous définissons un ensemble de *features de référence* dont la liste complète est en Annexe 3. Ces features sont générales, utilisées de manière récurrente dans la littérature sur l'analyse audio, parfaitement bien définies mathématiquement, et bien connues. Elles incluent notamment les features proposées dans la norme MPEG7-audio, ainsi que plusieurs autres telles que le *Chroma*, souvent utilisé pour l'analyse de musique tonale (Gomez, 2006).

Nous évaluons ensuite les performances de deux classifieurs, l'un avec cet ensemble de features générales, l'autre avec les meilleures features analytiques trouvées par EDS. A chaque fois, nous conduisons deux expériences.

Dans la première les classifieurs sont entrainés sur la base d'apprentissage, et testés sur la base de test. Chacune de nos bases de données (sons entiers et attaques) est scindée en deux afin d'obtenir une base d'apprentissage et une base de test distinctes. Dans les deux cas nous utilisons les deux tiers des échantillons pour l'apprentissage et le tiers restant pour l'évaluation. En outre, les échantillons sont mélangés avant la séparation en deux bases, afin de minimiser les artéfacts dus par exemple à l'évolution de la tension de la membrane en cours d'enregistrement, ou à de légères variations dans le jeu de l'instrumentiste.

Dans la seconde, les classifieurs sont entraînés et testés sur la base de test uniquement, en utilisant la cross-validation « 10 fois » (*10 fold cross-validation*) : chaque classifieur est entrainé sur $^{9}/_{10}$ èmes des échantillons de la base de test, et sa précision est testée sur le 10ème restant. Cette opération est répétée 10 fois, et à chaque fois, les échantillons d'apprentissage et de tests sont tirés au sort. Cette technique est réputée être fiable et éliminer un maximum de biais.

Cette double expérience a pour but de montrer que l'avantage obtenu avec les features analytiques est robuste, et indépendant en particulier des conditions de l'expérience. Par ailleurs la cross-validation dans la deuxième expérience se justifie par le fait qu'EDS utilise déjà la base d'apprentissage pour construire ses features analytiques, et il n'est donc pas souhaitable de la réutiliser pour entraîner les classifieurs.

Enfin, dans le cas des attaques, une dernière expérience consiste à entrainer des classifieurs en utilisant le signal lui-même, considéré comme une feature (de 128 valeurs). En effet, ces signaux étant très courts, on peut se permettre de les utiliser directement comme feature. Les résultats obtenus montrent que le signal n'est pas une bonne feature (cf. Section suivante).

3.3. Choix des classifieurs

Il existe une riche littérature concernant les algorithmes d'apprentissage supervisé (Witten et Eibe, 05) ainsi que de nombreuses implémentations des principaux algorithmes. Nous avons mené nos expérimentations en utilisant plusieurs algorithmes considérés comme les plus puissants (SVM, *k*NN, J48, réseaux de neurones). Dans un souci de clarté et de concision, nous ne donnons ici que les résultats obtenus avec SVM (Shawe-Taylor and Cristianini, 2000), le plus stable et le plus performant des algorithmes que nous avons utilisés. Nous utilisons l'algorithme SVM dans l'implémentation fournie par la bibliothèque Weka (http://www.cs.waikato.ac.nz/ml/weka) et en utilisant les paramètres par défaut (notamment le noyau polynômial).

Nous avons utilisé EDS de façon totalement automatique pour la création des features analytiques. Pour chaque problème, nous avons fait tourner la recherche génétique jusqu'à ce qu'EDS n'améliore plus substantiellement les fitness des features. Concernant les sons entiers, EDS a évalué environ 40.000 features. Concernant les attaques, EDS a évalué environ 200,000 features.

Les features de référence (cf. Annexe 2) ne sont pas toutes calculables sur tous les échantillons, notamment pour les très petits échantillons. Dans le cas des attaques, seules 17 features sont calculables, pour une dimension totale du feature set de 90. Pour les sons entiers, toutes les features ont pu être calculées, pour une dimension totale de 100.

La section suivante décrit le processus (délicat) de sélection des features.

3.4. Sélection des features

Pour comparer les deux approches (features générales vs features analytiques), il est important d'entraîner les classifieurs sur des espaces de dimensions identiques. Nous avons sélectionné 90 features analytiques (scalaires) parmi les 200.000 calculées pour les attaques, et 100 features (scalaires) parmi les 40.000 calculées pour les sons entiers.

Pour illustrer l'apport des features analytiques, nous présentons les résultats obtenus avec deux méthodes de feature selection. Cette dernière doit en effet prendre en compte les corrélations entre features, afin d'éviter de sélectionner plusieurs features, éventuellement de bonne qualité, mais similaires, et donc redondantes. Il existe de nombreuses méthodes de feature selection (Guyon and Elisseeff, 03) et le choix de méthode de sélection peut s'avérer déterminant dans la qualité d'un classifieur.

Pour s'assurer que nos résultats ne dépendent pas de l'algorithme de feature selection, nous utilisons deux méthodes différentes. D'une part, nous utilisons l'algorithme IGR (Information Gain Ratio) (Quinlan, 93). Techniquement celui-ci correspond à l'algorithme *AttributeSelection* de Weka avec les paramètres suivants : l'*evaluator* est un *InfoGainAttributeEval* et le *search* est un *Ranker*, qui permet de fixer a priori la dimension du feature set que l'on veut obtenir.

D'autre part nous avons développé un algorithme de sélection de features adapté à EDS. L'idée est de construire un feature set qui « couvre » optimalement les classes à apprendre, du point de vue de la F-mesure (Cf. section 2.3). L'algorithme itère sur toutes les classes et sélectionne itérativement les features ayant la meilleure F-mesure pour cette classe.

Enfin, nous donnons les résultats pour des feature sets de différentes tailles (de 1 à 100). Ce dernier aspect est important car comme nous le verrons les features d'EDS permettent de construire des features set de taille plus petite que les feature sets standards à performance égale ou supérieure.

3.5. Résultats et commentaires

Les tableaux Figure 2 et Figure 3 montre les résultats obtenus pour les différentes expériences.

Experiment Description			Feature Set Dimension										
			100	90	75	50	25	15	10	5	3	2	1
Reference	*IGR*	*Train/Test*	97,9	97,6	97,5	96,9	94,3	89	85,3	61,4	51,6	39,4	37
Reference	*IGR*	*10-fold XV*	97,1	96,7	96,9	95,2	94,5	89,1	83,6	47,5	43,3	26,8	19,2
EDS	*IGR*	*Train/Test*	98,6	98,4	98,4	97,9	96,6	95,2	86	71,7	66	43,5	47,4
EDS	*IGR*	*10-fold XV*	97,4	97,5	96,9	96,9	95,1	91,2	90,7	86,6	59,5	43,3	43,6
Reference	*EDS FS*	*Train/Test*	98	97,9	97,7	97,2	94,8	93,5	91,8	88,2	63,9	48,5	43,3
Reference	*EDS FS*	*10-fold XV*	97,3	97,2	96,8	96,5	95	93,1	91,8	87,3	60,5	50,4	41,8
EDS	*EDS FS*	*Train/Test*	98,6	98,5	98,1	97,6	96,7	96	95,7	91,2	76,4	76,1	56,2
EDS	*EDS FS*	*10-fold XV*	97,5	97,5	97	97,1	96	95,2	94,4	91,7	75,4	74,8	51,2

Figure 2**.** Tableau récapitulatif des résultats obtenus sur les **sons entiers** de pandeiro. **IGR** désigne la méthode de sélection Information Gain Ratio. **EDS FS** désigne notre algorithme de feature selection basé sur la F-mesure. **Train/Test** désigne l'expérience où le classifieur est entraîné sur la base d'apprentissage et évalué sur la base de test. **10-fold XV** désigne la méthode de cross-validation 10 fois.

Experiment Description			Feature Set Dimension									
			90	75	50	25	15	10	5	3	2	1
Reference	*IGR*	*Train/Test*	54,6	53,8	44,8	44,1	41,8	42	40,6	35,9	36,1	32,7
Reference	*IGR*	*10-fold XV*	46	45,3	44,4	40,1	37,7	34	34,1	31,8	31	32
EDS	*IGR*	*Train/Test*	65,2	63,4	56,5	50,1	48,5	42,3	33,4	34,8	34	32,6
EDS	*IGR*	*10-fold XV*	58,3	56,8	53,8	51,9	50,3	47,5	40,7	35	26,9	27,6
Reference	*EDS FS*	*Train/Test*	54,7	54,3	52,5	46,3	44,8	43,7	40,7	36,2	33,9	32,4
Reference	*EDS FS*	*10-fold XV*	47,1	47,3	44,8	42,5	40,2	38,7	38,1	34,4	32,3	30,7
EDS	*EDS FS*	*Train/Test*	65,9	65,1	62,8	60,8	59,1	57,6	55,2	43,7	42,6	40,5
EDS	*EDS FS*	*10-fold XV*	58,3	58,7	57,9	56,3	55,9	56,7	53,7	39,7	39,6	35,5
Signal			31	31	31	31	31	30	30	29	29	28

Figure 3**.** Tableau récapitulatif des résultats obtenus sur les **attaques** de pandeiro.

Pour les deux problèmes considérés dans cet article, les features analytiques créées à l'aide d'EDS permettent d'améliorer la performance des classifieurs.

Pour les sons entiers, le problème de classification est relativement facile. L'utilisation du jeu complet des features de référence (de dimension 100) apporte une précision d'environ 97%. Les features analytiques apportent un gain en performance, évidemment limité (de l'ordre de 1%). Le gain est plus intéressant si l'on réduit la dimension du feature set: 5 features analytiques apportent une précision supérieure à ce qu'apporte 10 features de référence (en utilisant la cross-validation).

Pour le problème des attaques, les résultats sont plus intéressants. On peut noter qu'à l'inverse du problème des sons entiers, le gain diminue avec la dimension du feature set. D'autre part, l'évolution du gain EDS dépend de l'algorithme de feature selection utilisé. L'algorithme IGR (standard) ne sélectionne pas les meilleures features EDS pour des feature sets de petite taille. En revanche notre algorithme de sélection, permet d'obtenir de meilleurs résultats pour toutes les tailles de feature set. Ceci est illustré par la Figure 4. Ce résultat montre la difficulté d'interpréter directement les résultats des classifieurs.

Ce gain en performance apporté par les features analytiques pour les petits feature sets a de nombreux avantages. Il permet en particulier une diminution substantielle du temps de classification pour les applications temps-réel. Pour le problème des attaques, 5 features analytiques suffisent à atteindre une précision supérieure à celle obtenue avec toutes les features de référence. Ces features sont :

- Variance (Mfcc (Power (Abs (x), 3), 11))
- Square (Log10 (Rms (Mfcc (Differentiate (Square (x)), 2))))
- Abs (Percentile (Mfcc (BpFilter (Integration (x), 310, 985), 10), 50))
- Abs (SpectralDecrease (Triangle (Square (Normalize (x)))))
- Abs (Iqr (Integration (Mfcc0 (Hann (HpFilter (Differentiate (x), 100)), 10))))

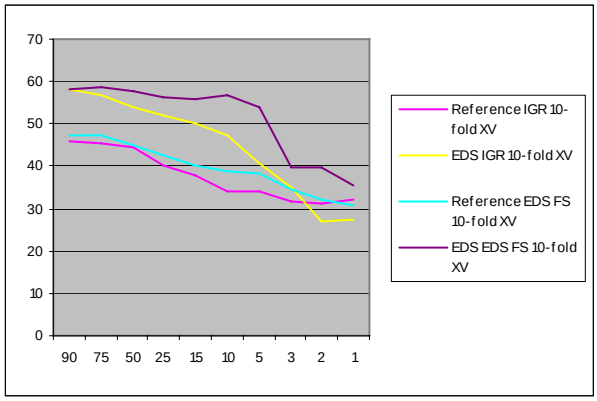

Figure 4. Performances comparées des features analytiques (EDS) avec les features de référence, en utilisant deux méthodes de feature selection, pour l'identifications des attaques de sons de pandeiro.

Dans le contexte d'exécution en temps réel qui nous intéresse, c'est un point essentiel en faveur des features analytiques. Sur un PC du moment (3Ghz), le calcul des cinq features analytiques pour un signal de 2,8 ms prends 3,15 ms, alors que le calcul des 25 features élémentaires requiert 7,6 ms, soit 2,5 fois plus de temps.

Qualité des enregistrements

Il est à noter enfin que nous n'avons pas mené d'expériences avec des échantillons enregistrés dans des conditions différentes de celles décrites dans cet article. Nous ne savons pas ce qu'un changement de microphone, d'instrument, d'instrumentiste, ou même simplement de lieu d'enregistrement, pourrait avoir comme conséquences sur l'efficacité de nos features. Nous n'avons pas mesuré leur robustesse face à ces changement de paramètres. Ces features étant par construction spécifiques au problème à résoudre, il est possible qu'elles « généralisent mal » en cas de changement de conditions d'applications. Mais ce problème est général à la classification supervisée et est

indépendant de la notion de feature analytique (Cf. par exemple le fameux « album effect » (Youngmoo et al., 06)). Des expériences en cours sont destinées à mesurer ces effets, en évaluons les résultats obtenus ici sur de novelles bases de sons construites dans des conditions légèrement différentes, ainsi qu'avec des sons obtenus en temps-réel, à la manière de l'étude menée pour la classification de sons de beat-boxing (Hazan and Ramirez, 05).

Notons par ailleurs que nous avons obtenus des résultats sensiblement différents de ceux présentés ici en utilisant une base d'échantillons plus petite (800 sons pour les attaques au lieu de 2184 de cette étude). Ceci montre l'importance de disposer de bases de taille suffisante pour interpréter les résultats de manière convaincante.

Néanmoins, d'autres travaux que nous avons menés sur des sons de nature différente, notamment des aboiements de chiens (plusieurs chiens enregistrés en réaction à des situations variées), nous ont montré que les features EDS généralisent aussi bien que des features standard à de nouveaux chiens et à de nouvelles situations (Molnár et al. 07).

4. CONCLUSION

Nous avons présenté une méthode de création de features acoustiques dites analytiques, par composition d'opérateurs élémentaires. Nous avons illustré cette approche sur deux exemples musicaux d'identification de sons de pandeiro. Pour ces deux problèmes, les features analytiques améliorent les performance des classifieurs par rapport aux features générales de type MPEG7, dans une approche *bag-of-frame*. Le gain est notable tant en ce qui concerne la précision des classifieurs que pour le temps de calcul des features, les features analytiques permettant une réduction notable de la dimension du feature set.

REMERCIEMENTS

Nous remercions Sylvain Marchand pour ses critiques constructives sur la version préliminaire de ce papier.

RÉFÉRENCES

Eric D. Scheirer, and Malcolm Slaney (1997) Construction and evaluation of a robust multifeature speech/music discriminator. Proc. ICASSP '97.

P. Herrera, A. Yeterian, F. Gouyon (2002) Automatic classification of drum sounds: a comparison of feature selection methods and classification techniques. Proceedings of 2nd International Conference on Music and Artificial Intelligence, Edinburgh, Scotland.

G. Peeters, X. Rodet (2002) Automatically selecting signal descriptors for sound classification. Proceedings of the 2002 ICMC, Goteborg (Sweden).

H.G. Kim, N. Moreau, T. Sikora (2005) Mpeg7 Audio and Beyond: Audio Content Indexing and Retrieval. Wiley & Sons.

Pachet, F. and Zils, A. (2004) Automatic Extraction of Music Descriptors from Acoustic Signals. Proceedings of ISMIR 2004.

A. Zils, (2004) Extraction de descripteurs musicaux: une approche évolutionniste, Thèse de doctorat, Université Paris 6.

J. R. Koza, (1992) "Genetic Programming: on the programming of computers by means of natural selection", Cambridge, MA: The MIT Press.

I.H. Witten, F. Eibe, (2005) Data Mining: Practical Machine Learning Tools and Techniques. M. Kaufmann Publisher, 2nd Edition.

B. E. Boser, I. M. Guyon, and V. N. Vapnik (1992) *A training algorithm for optimal margin classifiers*. In D. Haussler, editor, 5th Annual ACM Workshop on COLT, pages 144-152, Pittsburgh, PA. ACM Press.

John Shawe-Taylor & Nello Cristianini (2000), Support Vector Machines and other kernel-based learning methods - Cambridge University Press.

Bernhard Schölkopf and Alex Smola (2002) Learning with Kernels, MIT Press, Cambridge, MA.

Youngmoo E. Kim, Donald S. Williamson, Sridhar Pilli (2006) Towards Quantifying the "Album Effect" in Artist Identification. ISMIR 2006, 7th International Conference on Music Information Retrieval, Victoria, pp. 393-394.

Geoffroy Peeters, Stephen McAdams, Perfecto Herrera (2000), Instrument Sound Description in the Context of MPEG7, Proceeding of ICMC 2000, Berlin, Germany.

J.R. Quinlan, (1993) C4.5: Programs for machine learning. Morgan Kaufmann.

C. J. van Rijsbergen (1979) Information Retrieval. Butterworths, London.

Elias Pampalk, Arthur Flexer & Gerhard Widmer (2005) Improvements of Audio-Based Music Similarity and Genre Classification (pp. 628-633), ISMIR 2005, ISMIR 2005, 6th International Conference on Music Information Retrieval London, UK.

G. Tzanetakis, P. Cook (2002) Musical Genre Classification, IEEE Transactions on Speech and Audio Processing, Vol. 10, No. 5, July, pp. 293-301.

Emilia Gomez Gutierrez (2006) Tonal Description of Music Audio Signals, PhD Thesis, Universitat Pompeu Fabra, Barcelone.

Geoffroy Peeters (2003) Automatic Classification of Large Musical Instrument Databases Using Hierachical Classifiers with Inertia Ratio Maximization, 115th AES Convention New-York, NY, USA

Geoffroy Peeters (2004) A large set of audio features for sound description in the Cuidado project. http://recherche.ircam.fr/equipes/analyse-synthese/peeters/ARTICLES/Peeters_2003_cuidadoaudiofeatures.pdf

D. van Steelant, K. Tanghe, S. Degroeve, B. De Baets, M. Leman, and J.-P. Martens, (2004) "Classification of percussive sounds using Support Vector Machines", Proceedings of the annual machine learning conference of Belgium and The Netherlands, Brussels, Belgium.

Guyon, A. Elisseeff (2003), An Introduction to Variable and Feature Selection, Journal of Machine-Learning Research, 3(2003) 1157-1182.

Gillet, O. and Richard, G. (2003). Automatic Labelling of Tabla Signals. In Proceedings of the 4th International Symposium on Music Information Retrieval (ISMIR2003), Baltimore, Maryland.

Hazan and R. Ramirez (2005), Towards automatic transcription of expressive oral percussive performances, Proceedings of International Conference on Intelligent User Interfaces, San Diego, USA.

R. A. Fisher (1936) "The use of Multiple Measurements in Taxonometric Problems" Ann. Eugenics, vol. 7, pp. 179-186.

Csaba Molnár, Frédéric Kaplan, Pierre Roy, François Pachet, Péter Pongrácz, Antal Dóka, and Ádám Miklósi (2007) A machine learning approach to the classification of dog (Canis familiaris) barks, à paraître dans Animal Cognition.

ANNEXES

Nous mettons à disposition des lecteurs intéressés l'ensemble des sons utilisés dans cette étude ainsi que les valeurs des features utilisées (fichiers au format Weka *arff*), à l'adresse suivante :

http://music.csl.sony.fr/pandeiro

4.1. Annexe 1 – Les opérateurs élémentaires d'EDS

La liste des opérateurs élémentaires utilisés par EDS dans cette étude est la suivante :

Abs	HFC	Pitch
Arcsin	HMean	PitchBands
AttackTime	HMedian	Power
Autocorrelation	HMax	Range
Bandwidth	HMin	RemoveSilentFrames
BarkBands	HpFilter	RHF
Bartlett	Integration	Rms
Blackman	Inverse	SpectralCentroid
BpFilter	Iqr	SpectralDecrease
Centroid	Length	SpectralFlatness
Chroma	Log10	SpectralKurtosis
Correlation	LpFilter	SpectralRolloff
dB	Max	SpectralSkewness
Differentiation	MaxPos	SpectralSpread
Division	Mean	Split
Envelope	Median	SplitOverlap
Fft	MelBands	Sqrt
FilterBank	Min	Square
Flatness	Mfcc0	Sum
Hamming	Mfcc	TwelveTones
Hann	Multiplication	Triangle
Hanning	Normalize	Variance
HarmSpectralCentroid	Nth	Zcr
HarmSpectralDeviation	NthColumns	Harmonicity(Praat)
HarmSpectralSpread	PeakPos	Ltas(Praat)
HarmSpectralVariation	Percentile	

Une description précise de chaque opérateur peut être trouvée dans (Zils, 04).

4.2. Annexe 2 – Les features de référence

La liste des features standard utilisées comme base de comparaison avec les features analytiques est la suivante (les features précédés par '*' n'ont pas pu être évaluées sur les attaques, trop courtes) :

```
* HarmonicSpectralCentroid(Hanning(x))
* HarmonicSpectralDeviation(Hanning(x))
* HarmonicSpectralSpread(Hanning(x))
Log10(AttackTime(x))
*Pitch(Hanning(x))
SpectralCentroid(Hanning(x))
* SpectralFlatness(Hanning(x))
SpectralSpread(Hanning(x))
Centroid(x)
PitchBands(Hanning(x),12.0)
Mfcc0(Hanning(x),20.0)
* HarmonicSpectralVariation(SplitOverlap(Hanning(x),2048,0.5))
Rms(x)
RHF(Hanning(x))
HFC(Hanning(x))
SpectralKurtosis(Hanning(x))
SpectralSkewness(Hanning(x))
SpectralRolloff(Hanning(x))
Iqr(x)
Chroma(Hanning(x))
MelBands(Hanning(x),10.0)
BarkBands(Hanning(x),24.0)
Zcr(x)
```

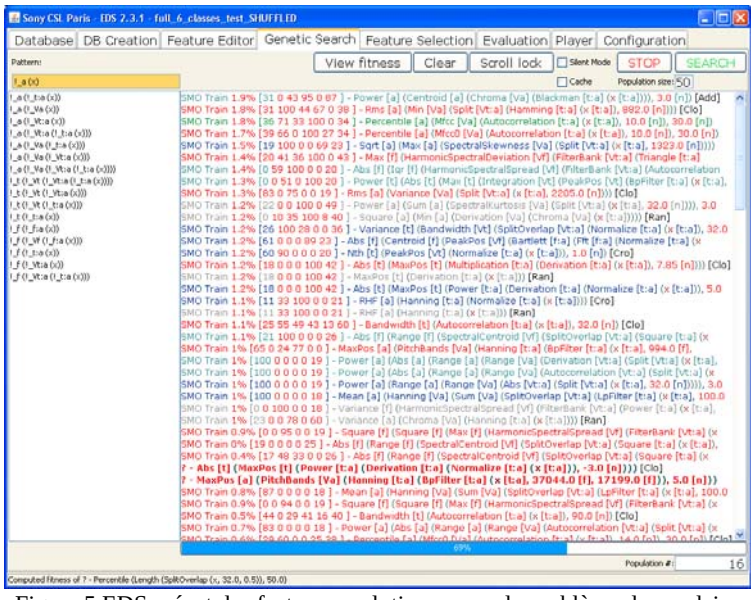

Figure 5 EDS créant des features analytiques pour le problème du pandeiro.