CLASSIFICATION DE MORCEAUX DE MUSIQUE EN GENRE MUSICAUX

L'objectif est de réaliser un classifieur de musiques selon les genres musicaux. Selon ces genres les fréquences dominantes varient, on pourrait alors à la lecture des fréquences deviner le genre musical : rap, classique, jazz, rock, etc. Il existe des algorithmes (e.g. K-NN, SVM) permettant de séparer des vecteurs en différentes classes. En effectuant un encodage du signal par une Short-Time-Fourier-Transform (STFC) ou par les Mel-Frequency Cepstral Coefficients (MFCC), vous classifierez ce signal selon le genre musical dont il est issu. Vous présenterez une analyse qualitative et quantitative des résultats (matrice de confusion, performances en temps d'exécution). Le binaire permettant d'effectuer des inférences devra être un programme C compilé. Vous avez la liberté d'utiliser des bibliothèques extérieures.

Toutefois une idée centrale de ce projet est de poser la problématique de l'utilisation de l'intelligence artificielle issue de l'apprentissage d'une machine ou d'algorithmes sur une cible quelconque (ordinateur, smartphone, carte avec ou sans OS, FPGA). Il s'agit de proposer une solution la moins coûteuse possible tout en préservant la performance. La question de la minimalité des ressources utilisées pour exécuter un calcul rejoint la problématique de la consommation d'énergie. D'une manière générale, moins on fait appel aux appels systèmes d'un OS, moins on fait appel à des bibliothèques, plus le système est minimal. Encore faut-il pouvoir garantir son efficacité. Les étapes principales de la chaîne de traitement sont présentées dans les paragraphes suivants.

1 - Entrées et représentation compressée de l'entrée.

La base de données la plus simple à utiliser, même si celle-ci n'est pas parfaite, est la collection GTZAN disponible sous différents formats à l'adresse suivante : http://marsyas.info/downloads/datasets.html. L'ensemble de données se compose de 1000 pistes audios de 30 secondes, de 10 styles différents, chacun représenté par 100 pistes. Les pistes sont toutes des fichiers audios monocanal, échantillonnés sur 16 bits (2 octets) à la fréquence de 22050 Hz. En tenant compte de ces données, chaque morceau est constitué d'environ 650 000 échantillons. Cela est difficilement interprétable par un algorithme. Les Short-Time-Fourier-Transforms permettent définir une représentation compressée des données en préservant le sens de celles-ci. Sur la figure 1, on peut voir un spectrogramme issu du calcul de STFT, le temps est segmenté, la gamme de fréquence l'est également, l'intensité de ces fréquences est représentée par un gradient de couleur. Afin d'obtenir un vecteur 1D réduit du spectrogramme, on peut représenter chaque colonne par sa moyenne et son écart-type selon soit l'axe du temps, soit l'axe des fréquences. La dimension du vecteur issu de cet encodage est : 2 x nbSegTemporels ou 2 x nbSegFrequentiels. Un exemple de code pour extraire la STFT d'un signal échantillonné est proposé dans les codes pour démarrer.

2 - Entraînement d'un classifieur

Les pistes audios encodées définissent un nuage de points dans un espace à n - dimensions. Il faut

réussir à trouver les hyperplans séparant ce nuage de points en dix classes. Il est possible d'effectuer avec cela avec un algorithme d'apprentissage supervisé : la régression SVM.
La figure 2 présente un classifieur SVM linéaire (voire également [1] pour son fonctionnement)
Un exemple de classifieur est proposé dans le dossier contenant les codes pour démarrer.

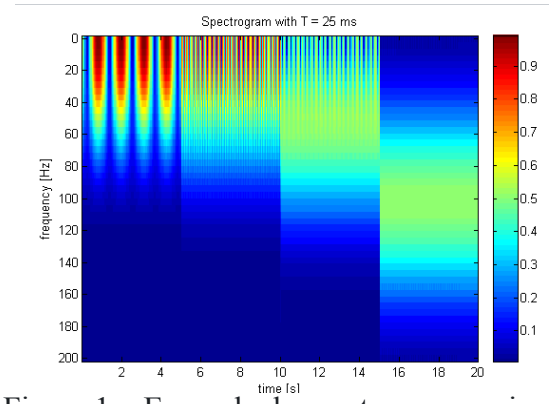

Figure 1 : Exemple de spectrogramme issu de l'encodage d'un signal par STFT (Wikipedia : Short Time Fourier Transform)

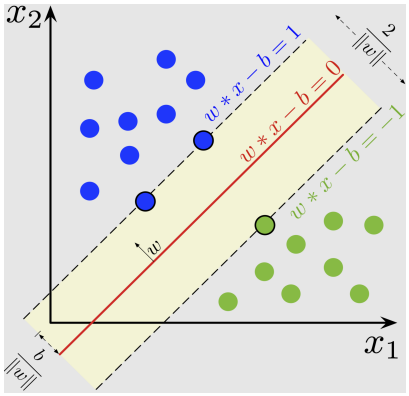

Figure 2 : Classifieur SVM linéaire (Wikipedia : Support Vector Machine)

3 - Prédiction et tests

Dans cette partie, nous allons effectuer une prédiction, nous allons créer le modèle en important les paramètres trouvés par le classifieur. Nous encodons le morceau avec les STFT et nous effectuons la prédiction. L'ensemble du processus de prédiction est un binaire compilé issu d'un programme C (cf.. figure 4). Afin de valider le modèle, il est intéressant d'utiliser des morceaux de musique différents de la base de données d'entraînement. Par exemple, il est possible de séparer la base de données en 75 % de données d'entraînement et en 25 % de données de test.
Une représentation visuelle et quantitative de la performance du modèle est la matrice de confusion. Il s'agit d'un tableau à double entrées de dimension égale au nombre de classes avec en colonne les classes prédites et en ligne les classes données en entrée. Par exemple, le classifieur d'images en figure 3 montre une bonne performance du modèle mais des difficultés à différencier certaines classes comme par exemple une voiture et un camion ou le chat et le chien …

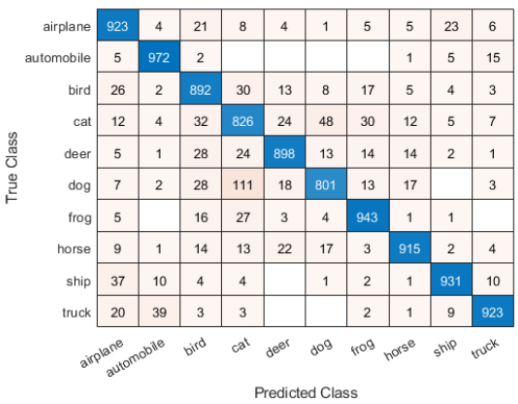

Figure 3 : Matrice de confusion (mathworks.com)

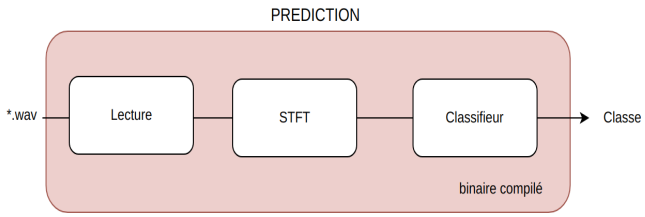

Figure 4 : Etapes de l'inférence

—

[1] scikit-learn.org/stable/modules/svm.html#svm-classification

4 - Gestion de projet

Les fonctions utiles seront dans un dossier include contenant les fichiers *.c et *.h.
Un Makefile compilera l'ensemble du projet.

4.1 Git

A la fin des séances vous exporterez votre travail sur GitHub :

- git init
- git add <files> (e.g. git add . ajoute l'ensemble des fichiers au futur commit)
- git commit -m "<message descriptif>"
- git remote add origin git@github.com:<user>/<reponame>.git (ajout du répo distant sous le nom origin)
- git push origin master.

Utiliser Git garantit une synchronisation du code entre les membres de l'équipe, un historique des versions et l'impossibilité de perdre une ligne de code ! Une bonne pratique est de créer au moins deux branches avec en permanence sur la branche principale appelée - master - un code fonctionnel (qui compile et s'exécute) et sur la branche - develop - les nouvelles fonctionnalités en cours de développement.

- git branch -b develop (création d'une branche develop)
- git checkout master (visualisation de la branche master)
- git merge develop (import des modifications de la branche develop sur la branche master)
- git push origin develop (envoi des modifications effectuées sur la branche develop vers le repo distant (e.g github ici représenté par le mot-clé origin)

Il est intéressant de créer un fichier .gitignore contenant l'ensemble des fichiers et dossiers que l'on ne veut pas intégrer au git (e.g. data/ build/) En effet, il n'est pas utile de pousser 1 Go de données sur GitHub ou de pousser ses fichiers compilés contenus dans build/.

4.2 Makefile

Dans des projets de taille moyenne contenant plusieurs modules *.h et *.c, le fichier Makefile bien configurer contient les instructions de compilation pour obtenir l'exécutable final. De plus, au cours du développement du projet, les fichiers *.h et *.c qui ne sont pas modifiés entre deux compilations ne sont pas re-compilés : cela permet un gain de temps lors de la compilation.
Un Makefile générique est proposé dans le repo GitHub. Il suffit de taper la commande `make` pour compiler le projet.

5 - Pistes d’amélioration

Il est possible d’améliorer la chaîne de traitement en travaillant au niveau de l’encodeur comme par exemple avec le Mel-frequency cepstrum MFC. L'échelle des mels est une échelle psycho-acoustique de la hauteur des sons : fondée sur la perception humaine des sons qui est plus sensible aux variations dans le grave que dans les aigus, elle permet de mieux représenter notre perception globale d'un spectre audio.

Il est aussi possible d’utiliser un autre type de classifieur comme par exemple le réseau de neurones. Pour chaque autre méthode intégrée à la chaîne de traitement : vous donnerez une comparaison quantitative des résultats avec votre modèle précédent.

Enfin vous pouvez également, utiliser d’autres bases de données voire des signaux non musicaux (e.g. voix humaines) Les applications sont directes dans le domaine de la robotique sociale (e.g. robots compagnons, …) ou de l’internet des objets (e.g. enceintes connectées, …).

—

[1] scikit-learn.org/stable/modules/svm.html#svm-classification