CLASSIFICATION DE D'ALBUMS MUSICAUX A PARTIR DE TRACE DE DIFFUSION SPATIO-TEMPORELLE

Louise Marchal et Alexia Bourmaud

Encadré par Sylvain Lamprier

PLDAC

1. Introduction

Depuis plusieurs années, les plateformes de streaming en lignes sont de plus en plus utilisées. Ces plateformes, souvent spécialisées dans un type de streaming (films, séries, musiques…), proposent de plus en plus de produits pour satisfaire leur clientèle.

La plateforme de streaming musical, Deezer, propose par exemple plus de 53 millions de titres[1]. En plus de posséder un large choix de produits, ces plateformes tentent de fidéliser leurs clients en leurs proposant des sélections de produits personnalisées.

Pour pouvoir proposer une sélection de produits personnalisée à chaque client il faut, pour ces plateformes, connaître d'une part les goûts du client et d'autre part posséder une description de leurs produits afin de trouver ceux qui correspondent au mieux aux goûts du client. Par exemple sur une plateforme de streaming musical, si un client écoute de nombreuses musiques décrites comme appartenant au genre rock, alors la plateforme aura tendance à lui proposer d'autres musiques également décrites comme étant du rock.

Cependant il est difficile et coûteux, pour ces plateformes, d'attribuer une description manuellement à tous les produits qu'elles diffusent, elles cherchent donc souvent des moyens automatiques de le faire. Cela leur permettrait d'enrichir leur base de données et ainsi d'améliorer les performances des recommandations qu'elles apportent à leurs clients.

Il existe plusieurs méthodes pour décrire de façon automatique un produit.

Les algorithmes de classification non supervisée permettent de regrouper des produits similaires ensemble, une description est alors attribuée à tout le groupe.

Les algorithmes de classification supervisée permettent à partir de produits déjà décrits de trouver la description d'autres produits.

Dans ce projet, nous avons travaillé en collaboration avec la plateforme de streaming, Deezer, qui nous a fourni une partie de ses données afin que nous trouvions un moyen d'attribuer un genre musical aux albums qu'elle possède.

En général la classification musicale s'appuie sur des attributs audios tels que le taux d'harmonique, le taux de silence, le flux spectral... Dans la majorité des cas, la classification supervisée est utilisée. Les modèles couramment employés sont SVM (*Support Vector Machine*), la régression logistique ou les réseaux de neurones[2], [3].

Néanmoins, les données fournies par Deezer s'appuient sur l'évolution du nombre d'écoutes de chaque album. Ainsi pour chaque album à classifier nous avons travaillé sur les traces de diffusion spatio-temporelle du nombre d'écoutes de cet album. La base de données utilisée est composée pour chaque album de son nombre d'écoutes dans plusieurs villes à plusieurs dates. Il est donc possible d'observer l'évolution du nombre d'écoutes d'une musique pour chaque ville en fonction du temps. En partant de l'hypothèse qu'il y a un lien entre la diffusion spatio-temporelle des écoutes d'un album et le genre de cet album, nous avons testé plusieurs algorithmes de classifications.

Dans un premier temps nous allons vous présenter différentes recherches effectuées sur la classification à partir de données spatio-temporelles. Nous allons aborder deux types de classifieurs, la machine à vecteurs de support (SVM) et le réseau de neurones récurrents (RNN), ainsi que les résultats obtenus avec chacun d'eux.

2. Etat de l'art

Les séries spatio-temporelles sont fréquemment utilisées pour faire de la classification, dans divers domaines. En géoscience par exemple, les images enregistrées par les satellites au cours du temps peuvent être utilisées pour attribuer à des régions du monde un niveau de stabilité. Pour cela il est possible d'utiliser un modèle d'apprentissage basé sur une machine à vecteurs de support (SVM), en modifiant la fonction noyau. L'utilisation d'une fonction radiale basique, par exemple, améliore les résultats[4].

Dans le domaine de la sécurité et notamment dans la classification de la densité de la foule dans des lieux publics, les séries spatio-temporelles sont très utiles. Ainsi à partir de données spatio-temporelles extraites de vidéo-surveillances, il est possible de détecter le niveau de densité d'une foule afin de prévenir les risques liés à une trop forte concentration de personnes dans un même lieu. Traditionnellement, cette étude se faisait grâce à la détection et au suivi de personnes dans la foule, cependant ces deux techniques sont difficilement réalisables. Pour pallier ce problème il est possible d'employer une détection de points d'intérêt combinée avec une machine à vecteurs de support afin de repérer le niveau de densité d'une foule[5].

Toujours dans le domaine de la sécurité mais ici en informatique, il est possible d'observer et de détecter les services réseau utilisés pas des flux de communication à l'aide de réseaux de neurones profonds. Cela a pour but d'améliorer la gestion et la surveillance des réseaux de l'internet. Les flux de données étant souvent caractérisés par des données spatiales et temporelles, les réseaux de neurones récurrents et convolutifs présentent de très bons résultats, pour ce qui est de détecter les flux de communication[6].

En épidémiologie, de nombreuses études cherchent à améliorer la détection de menaces dues à des agents pathogènes notamment dans les aliments ingérés. L'utilisation de biocapteurs se révèle être un très bon outil pour détecter les agents pathogènes. Les biocapteurs sont des instruments combinant un réactif biologique sensible aux agents pathogènes et un détecteur qui traduit la réponse biologique en signal électrique. Cependant les biocapteurs ne sont pas très efficaces pour détecter les faibles concentrations d'agents pathogènes, ce qui pose problème lorsque les agents en question sont très nocifs même ingérés en très faible quantité. Certains chercheurs ont donc combiné l'action des biocapteurs avec une machine à vecteurs de support afin d'améliorer la détection d'agents pathogènes[7].

D'autres outils tels que les réseaux de neurones profonds peuvent être utilisés en épidémiologie, comme dans d'autres domaines pour réaliser des prédictions sur des séries spatio-temporelles. En particulier, les réseaux de neurones récurrents et ses dérivées, les réseaux de neurones spatio-temporels, sont appréciés car ils modélisent au mieux les évolutions spatiales et temporelles[8].

Aucune étude n'a encore été faite dans le but de classifier des musiques à partir de leur diffusion spatio-temporelle. Cependant dans divers domaines, des données

spatio-temporelles sont utilisées pour réaliser des classifications, grâce notamment à des machines à vecteurs de support ou à des réseaux de neurones profonds.

3. Formalisation du problème

A. Présentation du problème

Soit X l'ensemble des données de travail, comportant pour chaque album, pour chaque date et pour chaque ville le nombre d'écoutes de cet album. X contient n exemples. Alors on a $\forall i \in \mathbb{N}\ et\ 1 \leq i \leq n,\ x_i \in \mathbb{N}^{|T| \times |V|}$ où T est l'ensemble des dates et V est l'ensemble des villes. La première valeur de T représente le premier jour où a été écouté l'album.

Soit Y l'ensemble des labels tel que $\forall i \in \mathbb{N}\ et\ 1 \leq i \leq n,\ y_i \in \mathbb{N}$.

Soient les couples $(x_i, y_i)_{1 \leq i \leq n} \in X \times Y$ où x_i désigne le i-ième albums y_i désigne le label associé à x_i. La base d'apprentissage S se constitue de la façon suivante :

$$S = \{(x_1, y_1), (x_2, y_2) \dots (x_n, y_n)\} \in (X \times Y)^n$$

Donc $x_{i,t,v}$ correspond au nombre d'écoutes du i-ième album au t-ième temps et dans la v-ième ville.

On cherche une fonction f qui prédit au mieux le label y_i du i-ième album.

$f(x_i) = \hat{y}$ avec $\hat{y}$ la prédiction de f concernant le label de x_i.

On cherche f qui minimise : $\sum_i L(f(x_i), y_i)$ avec $L(\hat{y}, y)$ une fonction de coût.

Durant notre projet nous avons utilisé deux bases de données différentes. La première est une base que nous avons créée nous-mêmes, dans le but de nous familiariser avec les données et les algorithmes de classification que nous avons utilisés. La deuxième est la base de données fournie par Deezer.

B. Données artificielles

Dans le cadre de notre projet, nous avons créé une base de données dont la structure se rapproche de la base de données fournie par Deezer. Le principal problème consistait à créer une base de données pas trop prévisible mais pour laquelle il serait tout de même possible de classer les exemples par genre.

Nous avons créé 700 exemples ($n = 700$), 5 villes différentes et 100 pas de temps. Dans cette partie $Y = \{1,2,3\}$.

Soit $\mu \in \mathbb{R}$ et $\sigma \in \mathbb{R}$ les paramètres de la loi normale. Voici comment on créé le temps 0 des exemples : $\forall j \in V\ x_{i,0,j} \sim \mathcal{N}(\mu, \sigma^2)$ où σ varie pour nos tests et μ est choisi en fonction du genre que l'on attribue à cet exemple et à une matrice de transformation.

Puis $\forall t \in T, \forall j \in V\ x_{i,t,j} \sim \mathcal{N}(\mu, \sigma^2)$ de même μ est choisi grâce au genre de l'album et à une matrice de transformation se basant sur la valeur au temps $t-1$ et σ varie pour nos tests.

La matrice de transformation est construite suivant une loi uniforme différente pour chaque genre.

C. Données Deezer

Deezer nous a fourni une base de données contenant 1543 albums, ainsi que leur trace de diffusion spatio-temporelle et leur genre.

Dans cette base un même album peut posséder plusieurs genres. Cependant, parmi les 51 genres présents dans la base, certains ne sont que très peu représentés, par exemple les genres folk, pop indé/folk, rock & roll/rockabilly, musique arabe, dub, musique religieuse, gospel, rock français, blues et dancehall/reggae ne sont associés qu'à un seul album.

Etant donné que l'apprentissage est difficile avec des classes possédant si peu d'exemples, nous avons décidé de regrouper manuellement les genres en onze classes.

-La classe 0 regroupe les genres films/jeux vidéo, musiques de films, sports, comédies musicales, classique, opéra, musique religieuse, jeunesse, gospel et possède 80 exemples.

-La classe 1 contient le genre rap français et possède 158 exemples.

-La classe 2 regroupe les genres rap/hip hop et east coast et possède 628 exemples.

-La classe 3 regroupe les genres pop, pop internationale, afro pop, pop latine, pop indé, pop indé/folk et possède 399 exemples.

-La classe 4 regroupe les genres dance, disco, dancehall/reggae, singer & songwriter et possède 196 exemples.

-La classe 5 contient les genres jazz, blues, country, soul, reggae, soul & funk et possède 50 exemples.

-La classe 6 contient le genre R&B et possède 74 exemples

-La classe 7 regroupe les genres electro, techno/house, pop/electro rock, trance et possède 165 exemples.

-La classe 8 contient les genres música colombiana, chanson française, world, latino, musique africaine, variété internationale, musique arabe, musique brésilienne et possède 87 exemples.

-La classe 9 regroupe les genres alternative, chill out/trip-hop/lounge, dub et possède 130 exemples.

-La classe 10 regroupe les genres rock, rock indé, hard rock, rock & roll/rockabilly, rock français, métal, folk et possède 87 exemples.

Dans la base de données certains albums sont associés à plusieurs genres, cela a pour conséquence que certains d'entre eux sont associés à plusieurs classes.

D'autre part, la base de données fournie par Deezer contenait le nombre d'écoutes de chaque album et le nombre de personnes différentes ayant écouté la musique, pour chaque ville (2683) pour chaque date (358), pour chaque tranche d'âge ("15<",

“15-18”, “18-24”, “25-34”, “35-44”, “45-54”,” >55”) et chaque plateforme d’écoute (“ios”, “android” et “web”).

Pour notre projet nous nous sommes intéressées au nombre d’écoutes de chaque album, toute plateforme d’écoute et tranche d’âge confondues.

Dans la suite du présent rapport nous allons vous présenter deux modèles permettant de réaliser une classification sur les données que nous possédons. Dans un premier temps nous aborderons la machine à vecteurs de support, puis dans un second temps les réseaux de neurones récurrents.

4. La machine à vecteurs de support

A. Introduction au SVM

La machine à vecteurs de support (en anglais *support vector machine*, SVM) est une technique d'apprentissage supervisé destinée à résoudre des problèmes de classification sur des données pas nécessairement linéairement séparables.

SVM va permettre de trouver une frontière linéaire, définit comme suit : $wx + b$ à laquelle vont s'ajouter parallèlement deux marges de chaque côté de la frontière.

Le but est donc de trouver les valeurs optimales de w et de b en minimisant l'erreur.

$$w^*, b^* = \min_{w,b} \frac{1}{2}\|w\|^2 + K\sum_i \xi_i \qquad tel\ que\ y_i\left(wx^i + b\right) \geq 1 - \xi_i\ et\ \xi_i \geq 0$$

On tolère que la frontière fasse des erreurs de prédiction. Le but reste néanmoins d'en faire le moins possible. Les variables « ressorts » ξ_i sont les raisons pour lesquelles il n'est pas nécessaire que les exemples soient linéairement séparables contrairement au perceptron par exemple car cette technique accepte que la frontière ne prédise pas correctement tous les exemples.

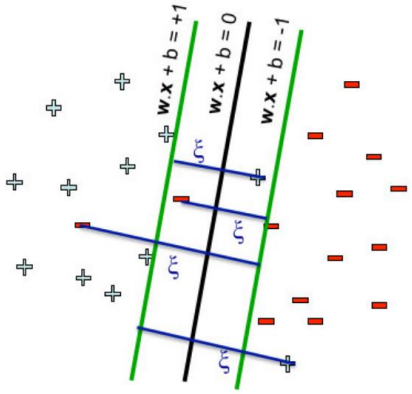


Figure 1 : Schéma explicatif du SVM à marge "molle"

De plus, grâce au SVM, il est possible de séparer des données non linéairement séparables, en les représentants dans un espace de plus grande dimension dans lequel les données sont linéairement séparables. Cette astuce se nomme le "kernel-trick", il suffit pour l'utiliser de remplacer le produit scalaire w.x dans la formule de la frontière par un noyau de la forme $K(w,x) = \varphi(w).\varphi(x)$ avec φ une fonction de transformation qui permet de changer d'espace.

B. Méthodologie

Nous avons utilisé SVM Light pour prédire les genres associés à chaque album. Puisque SVM Light ne permet pas de faire directement de la prédiction multi classe, il nous a fallu utiliser la méthode du one versus all pour réaliser nos prédictions. La méthode du one versus all consiste en l'apprentissage binaire de chaque classe contre toutes les autres.

La prédiction se fait ensuite en appliquant sur l'exemple les modèles appris sur chaque classe puis en prédisant la classe qui retourne le meilleur score.

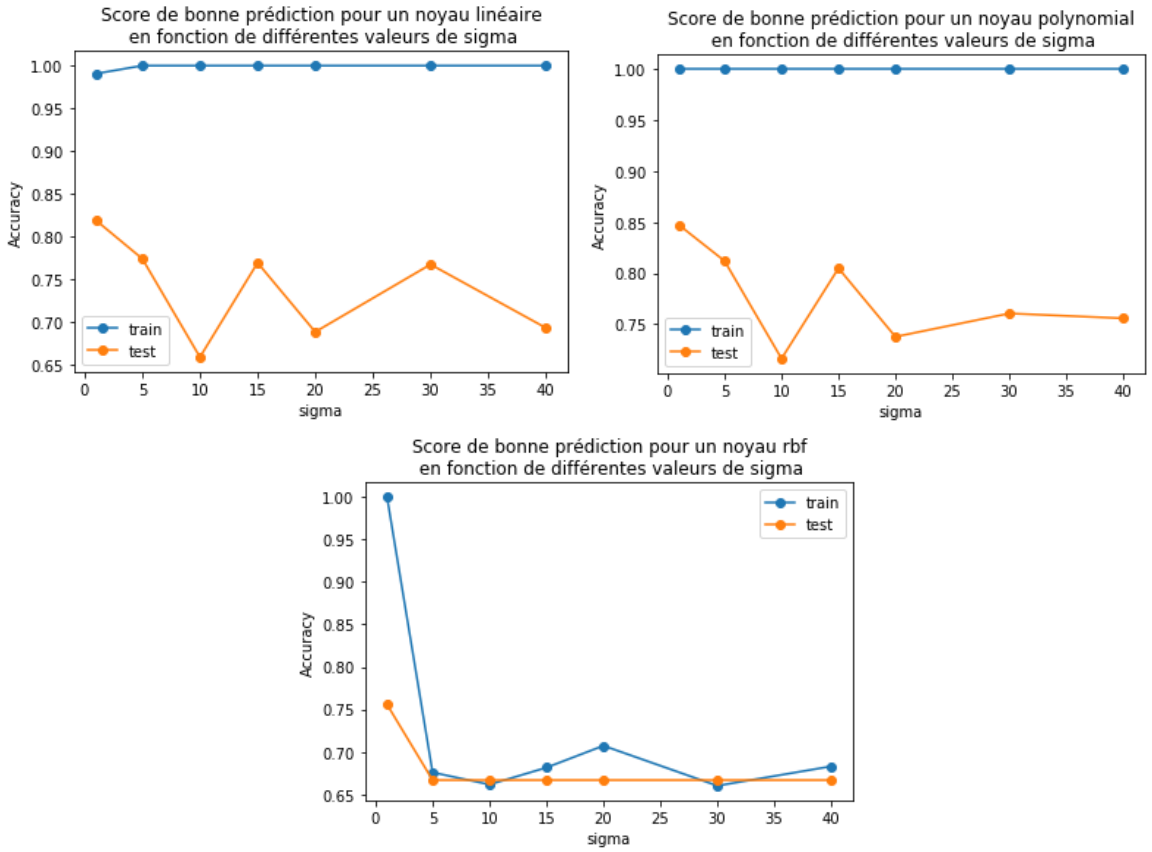
Afin d'obtenir des résultats plus fiables nous avons utilisé la technique de la validation croisée. Lors d'une validation croisée, la base de données est divisée en N blocs. Pour chaque bloc, les données de ce bloc sont utilisées comme ensemble de test et les données des N-1 autres blocs sont employées comme base d'apprentissage.

Enfin, nous avons testé différents noyaux dans le but de trouver l'espace de représentation des données qui permet de séparer au mieux les données.

C. Résultats avec les données artificielles

Les premiers tests ont été effectués sur les données artificielles. Ces essais avaient pour but de nous familiariser avec l'algorithme SVM light, et de vérifier s'il est capable de classifier des données représentant des traces de diffusion spatio-temporelles. L'objectif premier est de biaiser suffisamment nos données pour que l'apprentissage ne soit pas trop facile. Puis de déterminer quels noyaux prédisent le mieux et lesquels sont donc le mieux adaptés pour des données spatio-temporelles.

La valeur de sigma permet d'ajouter ou d'enlever du bruit dans la base de données.



On peut voir que plus les données sont biaisées plus le score de bonnes prédictions en phase de test diminue. Néanmoins les résultats sont prometteurs pour la suite car il

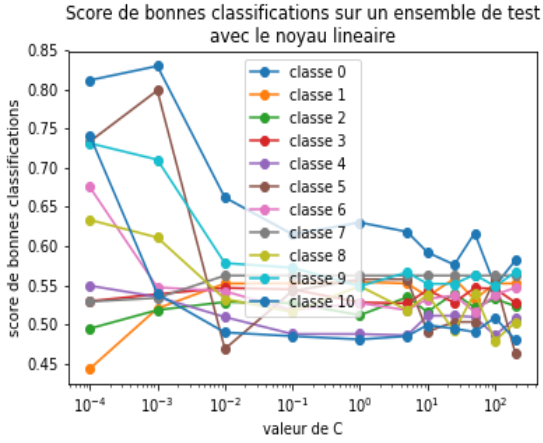
semble possible d'associer un genre musical à des albums à partir de leurs traces de diffusion spatio-temporelle.

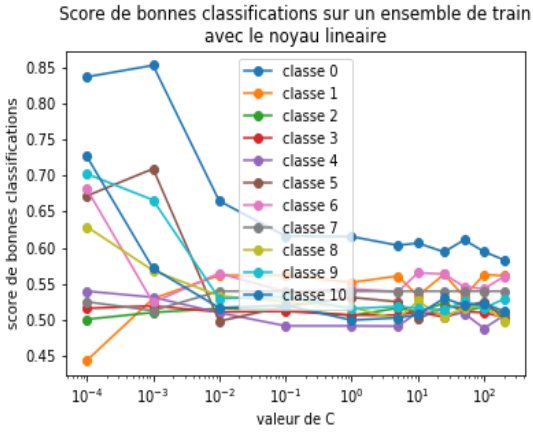
D. Résultats sur les données Deezer

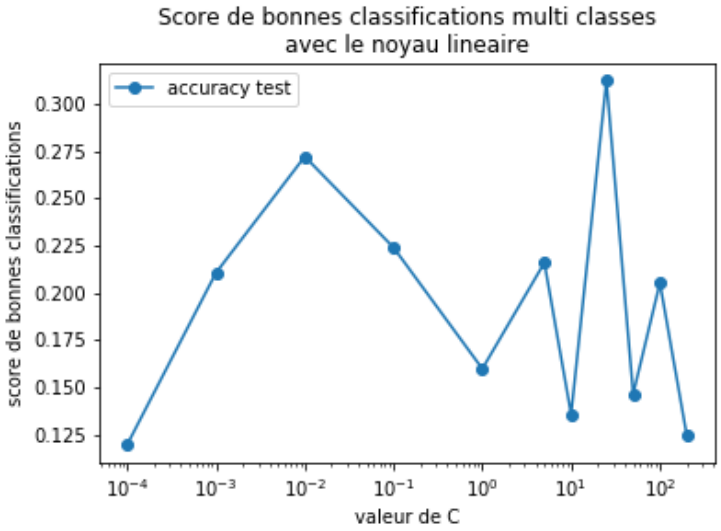
Il est maintenant nécessaire de vérifier les performances de SVM sur les données réelles fournies par Deezer.

Dans le but de trouver le modèle optimal, nous avons fait varier la taille de la marge de la frontière de nos modèles grâce au paramètre C de SVM light. Nous avons appris les modèles avec les valeurs de C suivantes: 0.0001, 0.001, 0.01, 0.1, 1, 5, 10, 25, 50 et 100.

Les graphiques suivants présentent la variation du score de bonnes prédictions pour chaque classe sur les ensembles d'apprentissage et de test, en fonction de la valeur du paramètre C, pour un noyau linéaire.



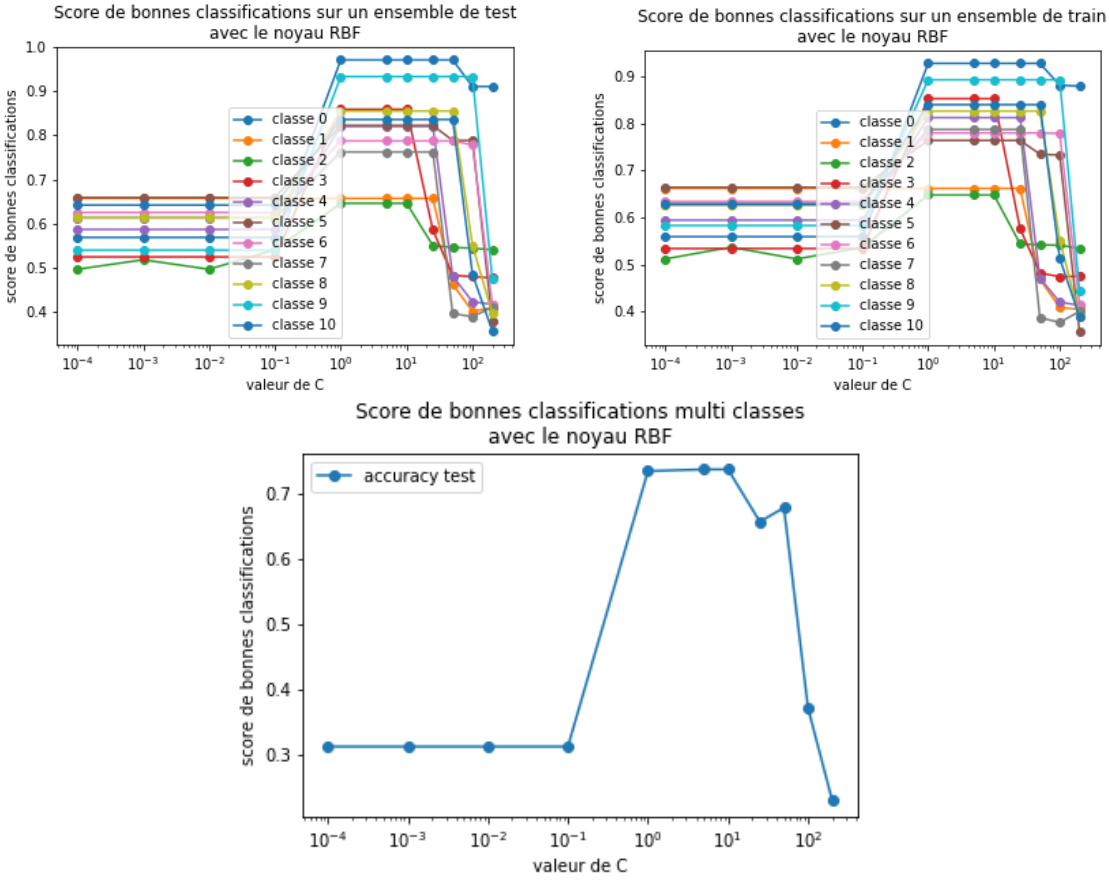




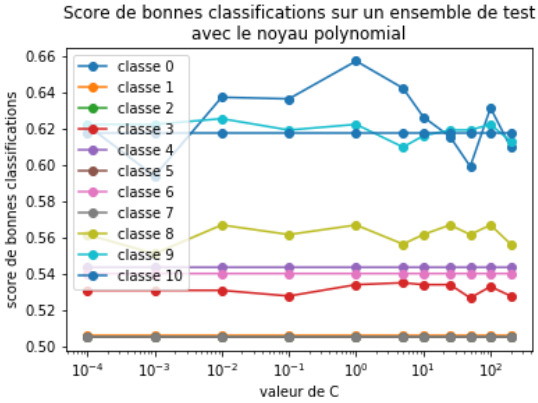
Nous observons qu'avec un noyau linéaire et une petite valeur de C (entre 0.0001 et 0.01), les scores de bonnes classifications de chaque classe sont disparates. Ils sont très bon pour certains (0.82 pour la classe 0, "film") et moins bon pour d'autre (0.45, pour

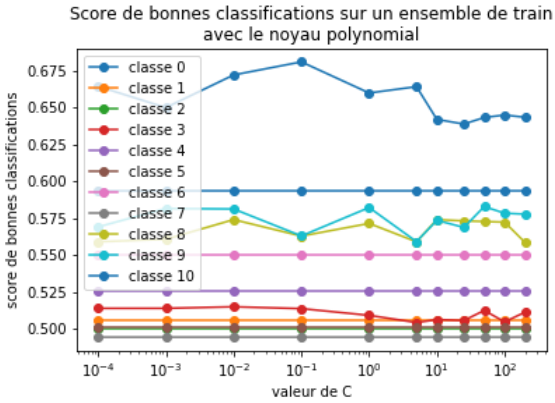
la classe1, “rap français”). Cependant pour des valeurs de C plus grandes les scores sont plus homogènes, ils sont compris entre 0.65 et 0.48. Nous n’observons pas de grandes différences entre les scores de bonnes prédictions obtenus sur les ensembles d’apprentissage, et ceux obtenu sur les ensembles de test. D'autre part, nous remarquons que les meilleurs scores en classification multi classe sont obtenus pour des valeurs de C valant 0.01 , on obtient alors 27% de réussite, et 25 qui permet d’obtenir un score de 31% de bonnes classifications.

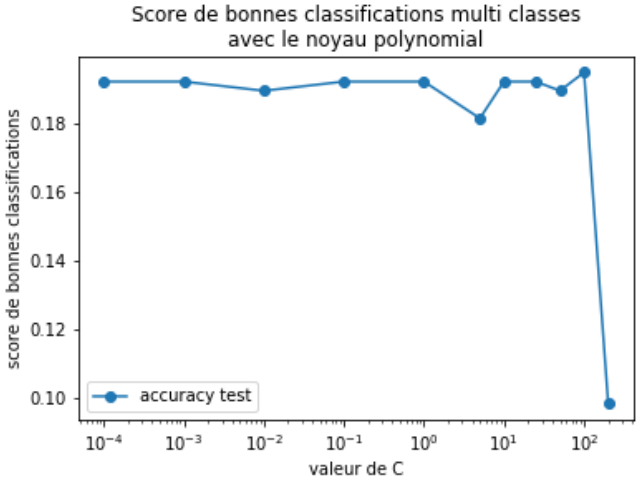
Le noyau linéaire semble avoir quelques difficultés à réaliser une classification multi classe sur les données fournies, cette observation n’est pas très étonnante puisqu’il est peu probable que des traces de diffusion spatio-temporelle soient linéairement séparables. Pour pouvoir séparer des données non linéairement séparables il est possible de changer le noyau de SVM, il s’agit de l’astuce du “kernel-trick” expliquée précédemment. Les résultats suivants présentent les variations du score de bonnes prédictions pour différentes valeurs du paramètre C, d’abord classe par classe, puis en multi classe, sur des noyaux RBF (*radial basic function*), polynomial et sigmoïde.

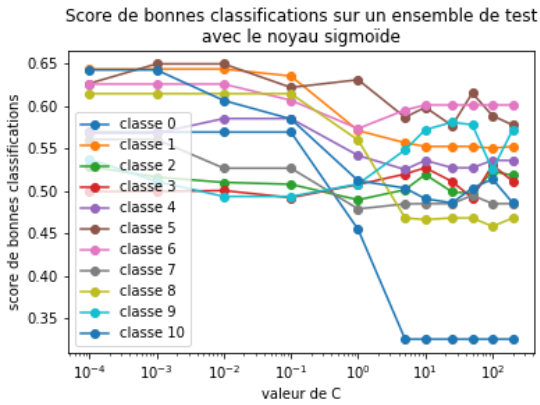


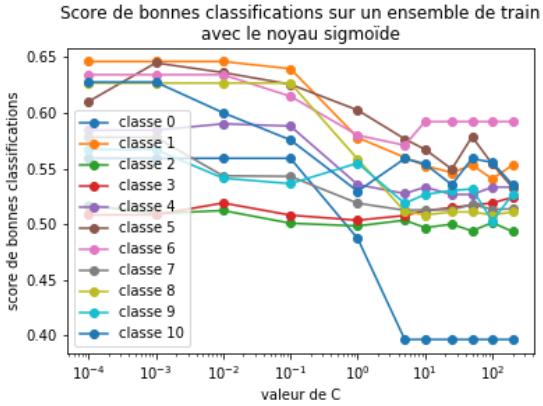
Le noyau RBF fourni des résultats très satisfaisant. Nous observons des scores compris entre 0.65 et 0.99, pour les classes séparées, que ce soit sur les ensembles d’apprentissage ou les ensembles de test. De plus pour des valeurs de C valant 1, 5 ou 10 les scores de bonnes classifications en multi classe sont très bons, ils avoisinent les 74% de réussite.

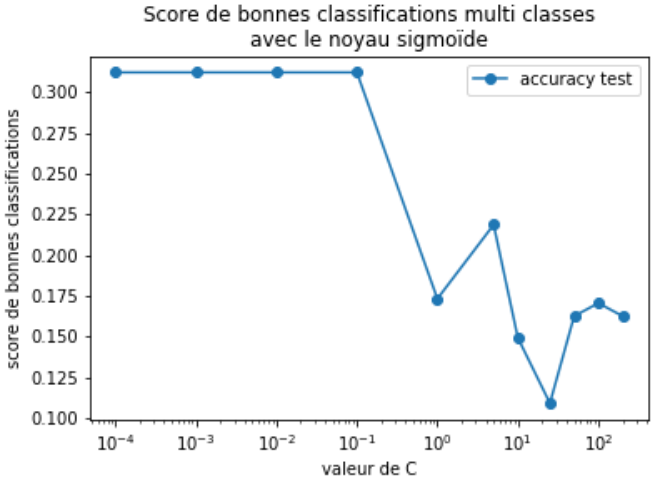

Score de bonnes classifications sur un ensemble de test
avec le noyau polynomial
score de bonnes classifications
valeur de C
classe 0
classe 1
classe 2
classe 3
classe 4
classe 5
classe 6
classe 7
classe 8
classe 9
classe 10


Score de bonnes classifications sur un ensemble de train
avec le noyau polynomial
score de bonnes classifications
valeur de C
classe 0
classe 1
classe 2
classe 3
classe 4
classe 5
classe 6
classe 7
classe 8
classe 9
classe 10


Score de bonnes classifications multi classes
avec le noyau polynomial
score de bonnes classifications
valeur de C
accuracy test


Score de bonnes classifications sur un ensemble de test
avec le noyau sigmoïde
score de bonnes classifications
valeur de C
classe 0
classe 1
classe 2
classe 3
classe 4
classe 5
classe 6
classe 7
classe 8
classe 9
classe 10


Score de bonnes classifications sur un ensemble de train
avec le noyau sigmoïde
score de bonnes classifications
valeur de C
classe 0
classe 1
classe 2
classe 3
classe 4
classe 5
classe 6
classe 7
classe 8
classe 9
classe 10


Score de bonnes classifications multi classes
avec le noyau sigmoïde
score de bonnes classifications
valeur de C
accuracy test

Les scores obtenus avec les noyaux polynomial et sigmoïde sont beaucoup plus constant que ceux obtenus avec les noyaux linéaires et RBF. En effet, en classification classe par classe les scores de bonnes prédictions sont compris entre 0.67 et 0.5. Ces scores ne varient pas beaucoup en fonction de la valeur de C. Cependant on observe qu'avec le noyau sigmoïde et des valeurs de C supérieurs à 5, le score de la classe 0, "film", est plus faible que les autres (0.35 sur les ensembles d'apprentissage et 04 sur les ensembles de test).

De plus, en utilisant un noyau polynomial, le score de bonnes prédictions multi classe ne varie presque pas en fonction de la valeur de C, il est globalement de 0.19, sauf pour C valant 200, où il est très faible (0.09).

Enfin, en utilisant un noyau sigmoïdale, les meilleurs scores sont obtenus pour des valeurs de C inférieures ou égales à 0.1. Le score de bonnes classifications est alors de 0,31.

En conclusion, il est préférable d'utiliser un noyau RBF pour réaliser des classifications sur les données fournies par Deezer, car le score obtenu est bien meilleur (0,73) qu'avec les autres noyaux.

5. Réseau de neurones récurrent

Puisque nos données peuvent être interprétées comme un nombre d'écoutes par villes d'un album variant au cours du temps, nous avons décidé d'utiliser un réseau de neurones récurrent pour essayer de prédire le genre de nos musiques.

A. Introduction aux réseaux de neurones récurrent

Un réseau de neurones est constitué d'un ensemble de modules, appelés neurones, reliés entre eux. Les neurones sont répartis en couches, plus il y a de couches plus le réseau est dit profond. Chaque couche de neurones prend en entrée la sortie de la couche précédente. La première couche prend en entrée les exemples, et la dernière couche retourne les labels prédits par le réseau de neurones.

L'entraînement d'un réseau de neurones se fait par rétro propagation du gradient de l'erreur. En effet, une fois que les données d'apprentissage ont parcouru tout le réseau, les labels prédits sont comparés aux labels réels. S'ils ne sont pas égaux, le réseau de neurones est parcouru dans le sens inverse afin que certains paramètres du réseau soient modifiés et qu'à l'itération suivante les prédictions soient meilleures.

Il existe plusieurs types de réseaux de neurones, nous avons utilisé un réseau de neurones récurrents (RNN) car il permet de prendre en compte l'évolution temporelle des données. Dans un RNN, chaque neurone possède deux entrées et deux sorties. L'une des sorties prédit le label des données passées en entrée, et l'autre correspond à un ensemble de données. Cet ensemble est nommé "hidden". L'une des entrées correspond à la sortie "hidden" du neurone précédent, l'autre entrée contient des données qui n'ont pas encore été observées.

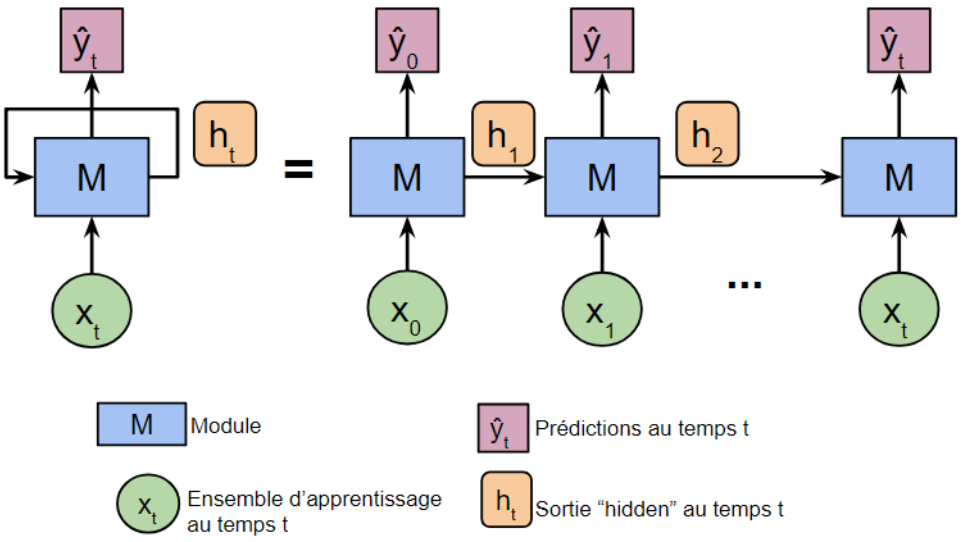


Figure 2 : Fonctionnement d'un réseau de neurones récurrents

Dans notre cas, les données qui n'ont pas encore été observées, passées dans chaque couche de neurones correspondent au nombre d'écoutes de chaque

musique, dans chaque ville, à un temps donné. Nous avons donc autant de modules que de dates différentes dans notre base de données (358). La première couche prend en entrée le nombre d'écoutes de chaque album dans chaque ville, le jour de sa sortie sur Deezer. La deuxième couche prend en entrée les données sorties par la première couche ainsi que le nombre d'écoutes de chaque album dans chaque ville, le lendemain de sa sortie, et ainsi de suite, jusqu'à la dernière couche qui prédit en sortie le label de chaque exemple.

B. Résultats sur un RNN simple

i. Parcours dans l'ordre chronologique

Lors de l'apprentissage du réseau de neurones récurrents nous avons utilisé la fonction d'optimisation Adam, ainsi que la fonction de coût cross entropie, puisque notre ensemble d'apprentissage est très déséquilibré.

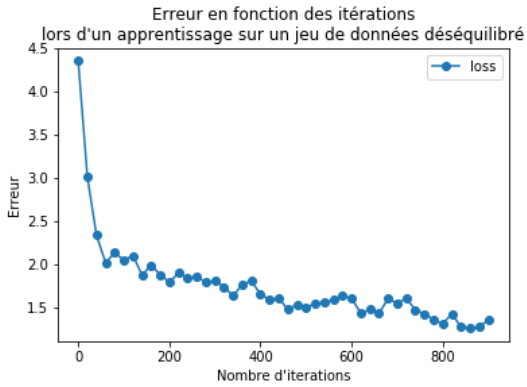
On cherche à maximiser la vraisemblance $max \prod_i P(y_i|x_i)$

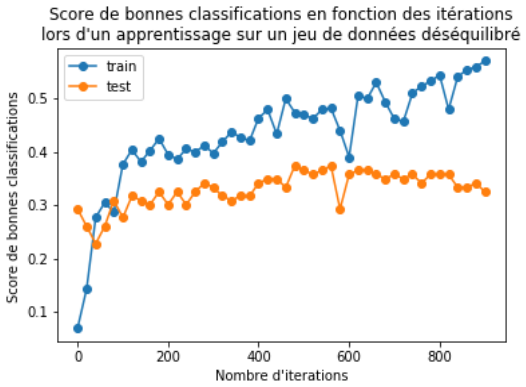
Ce qui revient à chercher $\min -\sum_i \log(P(y_i|x_i))$

Où $P(y_i|x_i) = \frac{\exp(RNN_i(x_i)}{\sum_{y'} \exp(RNN_{y'}(x_i))}$

Une seule itération d'un réseau de neurones récurrents (forward et backward) peut se révéler très longue lorsqu'il y a 358 modules (car 358 temps). Nous avons donc choisi de regrouper nos dates par groupes de dix afin d'avoir moins de modules dans le réseau. Nous sommes ainsi passées de 358 modules à 36. De plus, nous avons utilisé la méthode du mini batch. En effet, nous avons appris à chaque itération sur un sous-ensemble d'environ 200 albums et non sur l'ensemble de nos données. Ces deux techniques nous ont permis de réduire considérablement le temps d'apprentissage.

Les graphiques suivants présentent l'erreur et les scores de bonnes prédictions du RNN appris en fonction du nombre d'itérations.



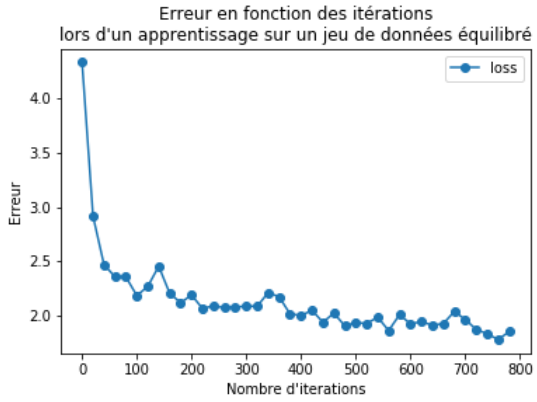


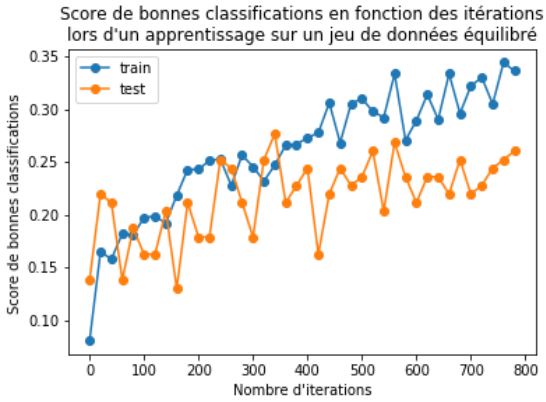
Nous observons que le coût décroît de façon logarithmique pour atteindre une erreur d'environ 1,5. Le score de bonnes classifications, quant à lui, augmente de façon logarithmique en fonction du nombre d'itérations. Cependant après 600 itérations le score de l'ensemble de test commence à décroitre légèrement et se stabilise autour

de 0.33, alors que le score de l'ensemble d'entraînement ne cesse de croître jusqu'à 0.57. Cela traduit un début de sur-apprentissage. Ce n'est pas surprenant car plus un réseau de neurones apprend plus il a tendance à sur-apprendre.

Nous pouvons en conclure que le nombre d'itérations optimal pour entraîner le réseau de neurones sur nos données se situe aux alentours des 550 itérations, le score obtenue sur les données de test est alors le meilleur (0.37).

Malgré l'utilisation de la fonction de coût cross entropie, il est envisageable que l'apprentissage du réseau de neurones soit faussé par des données très déséquilibrées (628 albums de la classe 2, le rap, contre 50 albums de la classe 5, le jazz). Les résultats suivants ont été obtenus en réalisant l'apprentissage du réseau de neurones sur un jeu de données équilibrés.





Comme précédemment l'erreur décroît de façon logarithmique, le score de bonnes prédictions croît de façon logarithmique, cependant le score obtenu sur l'ensemble de test croît moins rapidement et se stabilise autour de 0.23 à partir de 400 itérations.

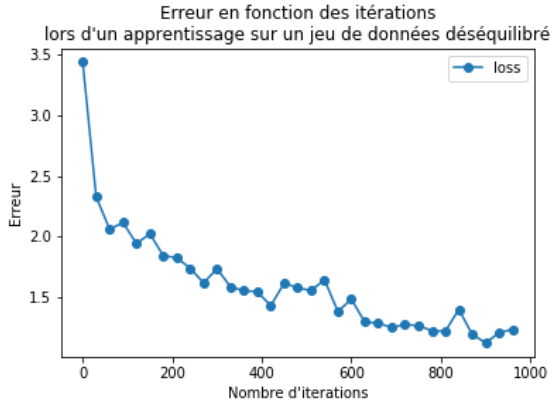
Nous observons que globalement les scores de bonnes prédictions sont moins bons lorsque le RNN apprend sur un ensemble de données équilibré que lorsqu'il apprend sur un ensemble déséquilibré.

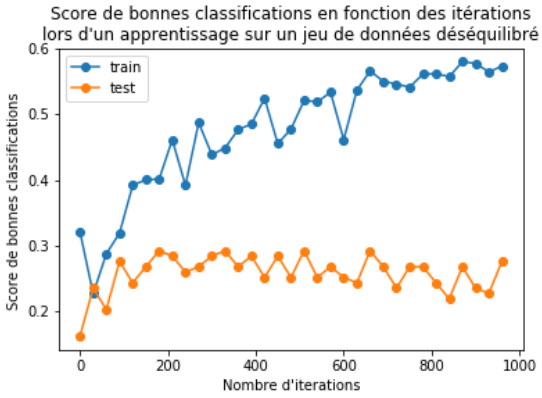
ii. Parcours dans l'ordre antéchronologique

L'un des problèmes souvent rencontré avec des réseaux de neurones récurrents est le problème du vanishing-gradient. En effet, plus le nombre de modules du réseau est important (ici 36), plus le gradient rétro-propagé est faible sur les modules de départ. Si le gradient est très faible, la mise à jour des poids est infime et le réseau de neurones ne se corrige presque pas.

Une solution à ce problème est de parcourir les modules dans l'autre sens c'est-à-dire en commençant le forward par le dernier temps puis en remontant jusqu'au premier.

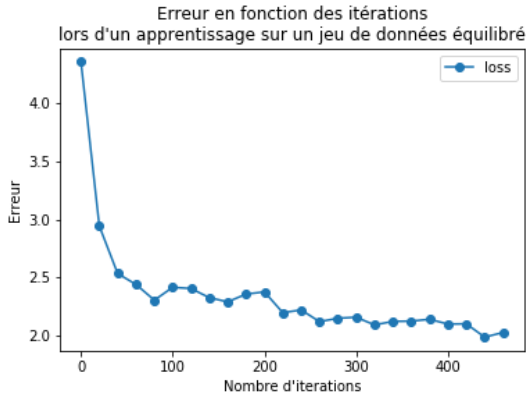
Les résultats suivants présentent le score de bonnes prédictions en fonction des itérations, avec un réseau de neurones récurrents ayant appris sur un jeu de données déséquilibrés.

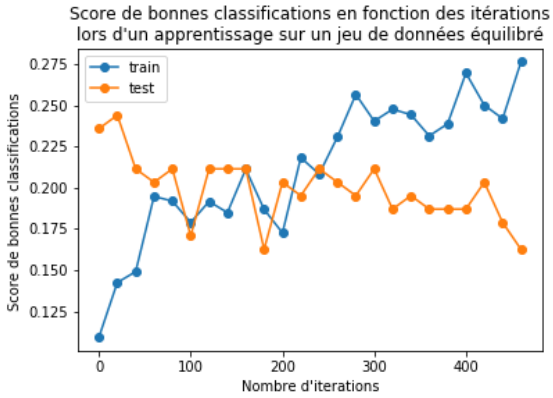




Nous observons, comme précédemment, une décroissance logarithmique de l'erreur jusqu'à atteindre 1,2. En ce qui concerne le score de bonnes classifications, on observe une croissance logarithmique pour l'ensemble d'apprentissage jusqu'aux environs de 0,57. Le score de bonnes classifications pour l'ensemble de test augmente légèrement jusqu'à 0.3, puis oscille autour de ce score. Cette différence de score à partir de 400 itérations traduit un sur-apprentissage.

Comme lors de l'apprentissage du RNN avec des temps dans l'ordre chronologique, il est envisageable que les résultats précédents soient faussés par le jeu de données déséquilibré. Les résultats suivants présentent le score de bonnes classifications au cours des itérations, pour un réseau de neurones récurrents parcourant les dates dans l'ordre antéchronologique et ayant appris sur un jeu de données équilibré.





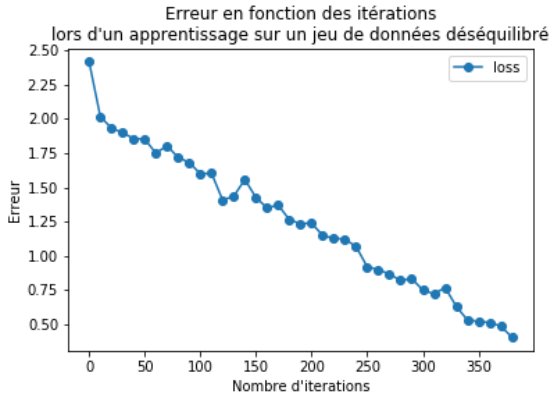
Nous observons que le score de bonnes classification sur l'ensemble de test décroit globalement jusqu'à 0.16, alors que le score sur l'ensemble d'apprentissage augmente jusqu'à 0.27. On observe clairement du sur-apprentissage.

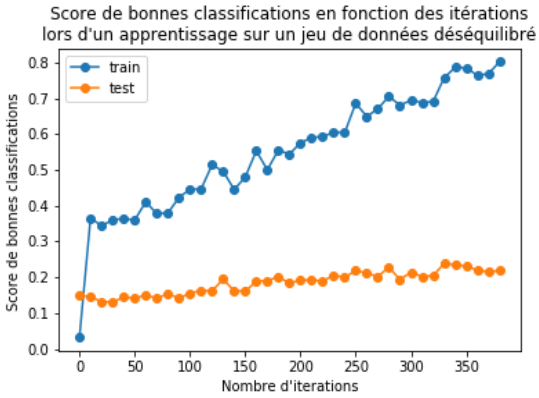
Au vu des résultats obtenus, on peut en déduire que parcourir les dates dans l'ordre antéchronologique ne permet d'améliorer les performances du réseau de neurones récurrent. De plus il semblerait que les réseaux de neurones ayant appris sur un jeu de données déséquilibré soient plus performant que ceux ayant appris sur un jeu de données équilibré.

C. GRU

Toujours dans le but de contrer le problème du vanishing-gradient, il est possible de modifier le mécanisme de déclenchement du réseau de neurones récurrents. L'un des mécanismes les plus performants et les plus simples est le mécanisme GRU (*Gated Recurrent Unit*).

Les graphiques suivants présentent les scores de bonnes prédictions au cours des itérations, sur un ensemble de données déséquilibré.





Nous observons une décroissance linéaire du coût, ainsi qu'une très forte croissance du score de bonnes prédictions pour l'ensemble d'apprentissage (0.8 au bout de 350 itérations). Le score de bonnes classifications sur l'ensemble de test croît quant à lui beaucoup plus lentement (0.23 au bout de 350 itérations). Il est fort possible que ce score augmente encore.

6. Conclusion

Afin d'attribuer automatiquement un genre musical à un album, à partir de ses traces de diffusion spatio-temporelle, différents modèles de classifieurs ont été testés durant ce projet.

Le modèle SVM permet d'obtenir un score de bonnes classifications en test de 73%, en utilisant un noyau RBF. Le modèle RNN, en parcourant les dates dans l'ordre chronologique, présente un score de bonnes prédictions en test maximal de 37% au bout de 550 itérations. Ce score est légèrement moins bon lorsque les dates sont parcourues dans l'ordre antéchronologique, le score obtenu est de 30% de bonnes prédictions. Dans tous les cas il est préférable d'utiliser un jeu de données déséquilibré pour apprendre le RNN, car les scores sont meilleurs qu'avec un jeu de données équilibré.

Enfin, malgré l'utilisation d'un réseau de neurones récurrents ayant le mécanisme GRU comme mécanisme de déclenchement, les scores obtenus reste faibles, 23% de bonnes classifications.

Nous en concluons qu'il est préférable d'utiliser un SVM avec un noyau RBF qu'un RNN car en plus d'obtenir un meilleur score il est également plus rapide.

Pour approfondir nos recherches, il aurait été intéressant d'entraîner le réseau de neurones, utilisant le mécanisme GRU, plus longtemps. Les résultats obtenus auraient probablement été meilleurs.

D'autre part, il aurait été envisageable d'apprendre un réseau de neurones convolutif car ce dernier est réputé performant sur des données spatio-temporelles.

Enfin, il aurait également été intéressant d'avoir plus d'exemples dans chaque classe pour ne pas être contraintes de faire des regroupements de genres lors de l'apprentissage. Ou bien d'essayer de réaliser des groupements de genres automatiques en réalisant des algorithmes de classification non supervisé, sur nos données comme l'algorithme des k-moyennes.

7. Références

[1] « À propos », *Deezer*. [En ligne]. Disponible sur: https://www.deezer.com/fr/company. [Consulté le: 10-juin-2018].

[2] K. Aryafar, S. Jafarpour, et A. Shokoufandeh, « Automatic musical genre classification using sparsity-eager support vector machines », in *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*, Tsukuba, Japan, 2012, p. 1526-1529.

[3] K. Aryafar et A. Shokoufandeh, « Multimodal Sparsity-Eager Support Vector Machines for Music Classification », in *2014 13th International Conference on Machine Learning and Applications*, Detroit, MI, USA, 2014, p. 405-408.

[4] S. Réjichi et F. Chaabane, « SVM spatio-temporal classification of HR satellite image time series using graph based kernel », in *2014 1st International Conference on Advanced Technologies for Signal and Image Processing (ATSIP)*, Sousse, Tunisia, 2014, p. 390-395.

[5] S. Lamba et N. Nain, « A Large Scale Crowd Density Classification Using Spatio-Temporal Local Binary Pattern », in *2017 13th International Conference on Signal-Image Technology Internet-Based Systems (SITIS)*, Jaipur, India, 2017, p. 296-302.

[6] M. Lopez-Martin, B. Carro, A. Sanchez-Esguevillas, et J. Lloret, « Network Traffic Classifier With Convolutional and Recurrent Neural Networks for Internet of Things », *IEEE Access*, vol. 5, nº 17246460, p. 18042-18050, sept. 2017.

[7] Y. Zuo, S. Chakrabartty, Z. Muhammad-Tahir, S. Pal, et E. C. Alocilja, « Spatio-Temporal Processing for Multichannel Biosensors Using Support Vector Machines », *IEEE Sens. J.*, vol. 6, nº 9191641, p. 1644-1651, déc. 2006.

[8] A. Ziat, E. Delasalles, L. Denoyer, et P. Gallinari, « Spatio-Temporal Neural Networks for Space-Time Series Forecasting and Relations Discovery », in *2017 IEEE International Conference on Data Mining (ICDM)*, New Orleans, LA, USA, 2017, p. 705-714.