Projet 3A - Détection du langage d'un locuteur sur enregistrement audio

Rémi Delbouys - Rémi Le Scaon

Février 2015

Contents

1 Remerciements **3**

2 Introduction **4**
2.1 Description du projet 4
2.2 Motivation à long terme du projet : Axa et la fraude 4
2.3 Utilisation de données labélisées par la langue 4
2.4 Contenu du rapport 4

3 Signal audio **5**
3.1 Bases de traitement du signal 5
3.2 Variable en entrée : un fichier audio wav 6
3.3 Variables candidates à la détection du langage 6

4 Deep learning **7**
4.1 Multi-Layer perceptron 8
4.2 Convolutional Neural Network 9
4.3 Recurrent Neural Network 10

5 Mise en oeuvre **11**
5.1 Langage et bibliothèques utilisées 11
5.2 Architecture de l'implémentation 11

6 Résultats **13**
6.1 Première approche : jeux de données classiques 13
6.2 Deuxième approche : cartographie du son 15

7 Avancement du projet **17**

8 Pistes de développement **17**

9 Annexe : Courbes ROC pour la première approche **19**

1 Remerciements

Nous tenons à remercier le département d'informatique et AXA de nous avoir donné l'opportunité de travailler sur ce sujet à la fois intéressant et formateur.

Nous remercions particulièrement Kevin Vu pour son aide fréquente et avisée ainsi qu'Aimé Lachapelle pour son implication.

2 Introduction

2.1 Description du projet

Dans le cadre de la chaire X-AXA consacrée aux *Data science for insurance sector*, nous avons choisi de nous positionner sur l'un des sujets de projet 3A proposés par l'entreprise. Il s'agit de rechercher le type de *features* calculés sur des données audio et le type de méthode d'apprentissage permettant d'identifier la langue utilisée par la personne qui parle. Il ne s'agira donc pas de s'appuyer sur le texte, l'idée étant d'évaluer la qualité de l'information contenue dans la voix.

2.2 Motivation à long terme du projet : Axa et la fraude

En réalité, ce projet s'inscrit dans une ambition plus grande que la détection du langage d'un locuteur. En effet, il s'inscrit dans l'initiative d'AXA de mettre les sciences des données au service de la détection de fraude.

Dans le secteur de l'assurance, la fraude est un enjeu majeur de la performance d'une assurance. Les pertes dues à celle-ci sont estimées à plusieurs milliards d'euros par an, si bien que certaines start-up en font leur activité principale. C'est en partie pour cela qu'AXA a créé dans ses locaux, situé à Suresnes, le *Data Innovation Lab*. Cette entité est spécialisée dans l'analyse des données d'AXA pour en exploiter de l'information. Plusieurs de leurs projets concernent la détection de fraude. Cela va de l'analyse de texte à la détection de réseau dans des graphes.

Dans le cas présent, il y a à l'origine de notre travail un projet de détection de fraude sur la base de données vocales. Il s'agit en effet de déterminer si une personne qui appelle l'assurance pour déclarer un sinistre est en train de frauder.

S'il s'avère que les résultats d'un classifieur permettant de détecter la fraude sont convaincants, l'entreprise pourrait l'appliquer, notamment dans le cadre de l'aide au discours de ses opérateurs, ou encore dans la classification des fraudes.

2.3 Utilisation de données labélisées par la langue

Initialement, il était prévu que nous puissions travailler directement sur des données vocales labélisées par le caractère frauduleux de l'appel. Cependant, des complications d'ordre juridique nous ont contraints à réorienter le projet vers des données plus facilement accessibles.

2.4 Contenu du rapport

Dans ce rapport, nous commençons par présenter des éléments de traitement du signal permettant de construire les différents ensembles de *features*. Nous présenterons d'abord ceux qui sont classiquement utilisés, puis nous étudierons une deuxième approche se basant sur des cartographies construites sur les données sonores.

Dans un deuxième temps, nous présentons des techniques d'apprentissage profond, dans le but de les appliquer aux jeux de données calculés précédemment.

Ensuite, nous détaillerons la mise en oeuvre algorithmique de ces techniques d'apprentissage.

Enfin, nous présenterons les résultats obtenus.

3 Signal audio

3.1 Bases de traitement du signal

Source principale : [7]

3.1.1 Transformée de Fourier

La transformée de Fourier $\mathcal{F}$ est définie tout d'abord pour toute fonction intégrable sur $\mathbb{R}$ par

$$\forall \xi \in \mathbb{R}, \mathcal{F}(f)(\xi) = \hat{f}(\xi) = \int_{\mathbb{R}} \mathrm{e}^{-i2\pi\xi x} f(x) \mathrm{d}x$$

L'opérateur $\mathcal{F}$ est inversible dès lors que la transformée de Fourier du signal est intégrable, si bien qu'on retrouve le signal initial en appliquant l'opérateur $\overline{\mathcal{F}}$ défini par

$$\forall x \in \mathbb{R}, \overline{\mathcal{F}}(\hat{f})(x) = f(x) = \int_{\mathbb{R}} \mathrm{e}^{i2\pi\xi x} \hat{f}(\xi) \mathrm{d}\xi$$

Elle s'étend par un argument de densité aux fonctions de carré intégrable.

Nous nous intéressons particulièrement à la transformée de Fourier en temps discret. On définit cette fois-ci les opérateurs $\mathcal{F}$ et $\overline{\mathcal{F}}$ sur l'ensemble des suites sommables puis on l'étend aux suites de carré intégrable :

$$\forall u \in l^1(\mathbb{Z}), \forall \nu \in [-\frac{1}{2}, \frac{1}{2}[, \mathcal{F}(\nu) = \hat{u}(\nu) = \sum_{n=-\infty}^{+\infty} u_n \mathrm{e}^{-i2\pi n\nu}$$

et si $\hat{u}$ est intégrable :

$$\forall n \in \mathbb{Z}, \overline{\mathcal{F}}(\hat{u})_n = u_n = \int_{-\frac{1}{2}}^{\frac{1}{2}} \hat{u}(\nu) \mathrm{e}^{i2\pi n\nu} \mathrm{d}\nu$$

On peut ensuite restreindre cette description aux signaux représentés par une suite finie $u = [u_0, \ldots, u_{N-1}]$, qui nous intéressent dans le cas de l'enregistrement d'un signal sonore. On associe à cette suite sa transformée de Fourier discrète $U = [U_0, \ldots, U_{N-1}]$ définie pour tout $k \in \{0, \ldots, N-1\}$

$$U_k = \sum_{n=0}^{N-1} u_n \mathrm{e}^{-2i\pi \frac{k}{N} n}$$

La formule d'inversion s'écrit pour tout $n \in \{0, \ldots, N-1\}$

$$u_n = \frac{1}{N} \sum_{k=0}^{N-1} U_k \mathrm{e}^{2i\pi \frac{k}{N} n}$$

3.1.2 Théorème de Shannon

Le théorème de Shannon impose que pour échantillonner un signal de fréquence f_0, il faut une fréquence d'échantillonnage f_e au moins deux fois supérieure à f_0. Plus généralement, si le spectre fréquentiel d'un signal est compris entre f_0 et f_1, il est nécessaire d'avoir $f_e > 2|f_1 - f_0|$. Pour

s'en convaincre, il suffit de considérer une fonction f de carré intégrable dont le spectre est nul en dehors de l'intervalle $[-B, B]$. Prenons $f_e < 2B$ et considérons la fonction obtenue en périodisant $\mathcal{F}(f)$:

$$F(\lambda) = \sum_{n \in \mathbb{Z}} \mathcal{F}(f)(\lambda - n f_e)$$

Cette fonction est périodique de période $1/f_e$. Elle admet donc une décomposition en série de Fourier. En appliquant la formule d'inversion de Fourier à cette série, on obtient que f s'écrit :

$$f(t) = \sum_{n \in \mathbb{Z}} f(nT) \frac{\sin(\frac{\pi}{T}(t - nT))}{\frac{\pi}{T}(t - nT)}$$

On retrouve donc le signal initial à partir des valeurs de f aux points nT.

3.2 Variable en entrée : un fichier audio wav

3.2.1 Le format wav

Le format wav est constitué d'un en-tête de 44 octets, suivis des données audio. L'en-tête contient notamment :

- la fréquence d'échantillonnage
- le nombre de canaux (de 1 à 6)

Le fréquence d'échantillonnage la plus fréquente est 44100Hz, ce qui d'après le théorème de Shannon permet d'échantillonner tout signal de fréquence inférieure à 22050Hz, ce qui correspond la borne haute de la fourchette constituée par le domaine audible.

Notons qu'à cette fréquence, 10 secondes d'enregistrement correspondent à 441 000 points par canaux. Il n'est donc pas envisageable de sélectionner les différentes valeurs du signal d'entrée comme *features* pour un algorithme de classification.

3.3 Variables candidates à la détection du langage

3.3.1 Variables utilisées dans la détection de genre et d'émotion

Deux articles scientifiques, [12] et [11], consacrés respectivement à la détection du genre et de l'émotion d'un locuteur, nous permettent de cibler un ensemble de variables.

La plus simple d'entre elles est la hauteur (*pitch frequency*), définie comme la fréquence d'amplitude maximale de la fenêtre de parole considérée. On peut ajouter à cela les deux formants, définis comme étant les fréquences associées aux deuxième et troisième pics. Ces variables suffisent souvent à déterminer le genre de la personne qui parle. On peut également citer les variables issues des MFCC (mel-frequency cepstrum coefficients). Ceux-ci sont obtenus en changeant l'échelle du spectre des fréquences de façon à favoriser les fréquences audibles par l'être humain, puis prendre le logarithme de ce nouveau spectre puis calculer la transformée en cosinus discrète. Associés à une méthode des moments généralisées, ces coefficients permettent de discriminer efficacement le genre [12] et l'émotion [11].

Notons l'existence des LPC (*linear prediction coefficients*) et LPCC (*linear prediction cepstral coefficients*) parfois utilisés dans le traitement de la parole.

Concernant la détection de langage, les MFCC pourront également s'avérer efficaces.

3.3.2 Autres variables possibles

D'après [13], les pauses lors d'un discours sont corrélées au débit de la parole. On peut donc être intéressé par l'étude des pauses lors d'un discours : fréquence des pauses, longueur des pauses, espacement entre les pauses sont autant de variables qui peuvent entrer en jeu lors de la détection du langage. Remarquons que ces variables peuvent aussi entrer en compte dans la détection de la fraude, puisqu'on peut espérer que les silences traduisent éventuellement des hésitations.

De même, dans l'optique d'une détection de schéma de fraude, pour prendre en compte le signal dans sa chronologie est d'étudier la variation d'amplitude au cours du discours. Enfin, on peut penser à des variables plus globales, comme la longueur des réponses de l'individu, ou la longueur des phrases.

3.3.3 Une autre approche : des cartographies du signal

Dans la partie suivante, nous présenterons les *convolutional neural networks*. Notamment, ceux-ci s'appliquent aux images. Par conséquent, il peut être intéressant de convertir le signal sonore en image pour appliquer cette technique d'apprentissage.

- **Cartographie d'amplitude** : il s'agit simplement de la représentation de l'amplitude en fonction du temps (cf. Fig 1).
- **Cartographie en fréquence** : en abscisse, la fenêtre de temps considérée, en ordonnée, la fréquence. La couleur représente la valeur de l'amplitude associée à cette fréquence (cf. Fig 2).
- **Cartographie représentant les MFCC** : en abscisse, la fenêtre de temps considérée, en ordonnée, les coefficients. La couleur représente la valeur du coefficient. (cf. Fig 3).

Figure 1: cartographie de l'amplitude

4 Deep learning

Nous avons choisi de nous intéresser aux réseaux de neurones pour tirer l'information significative de nos jeux de données. Avant de nous lancer dans nos implémentations, nous avons cherché à nous renseigner sur ce qui avait été fait dans ce domaine pour l'analyse de signaux audio. Dans

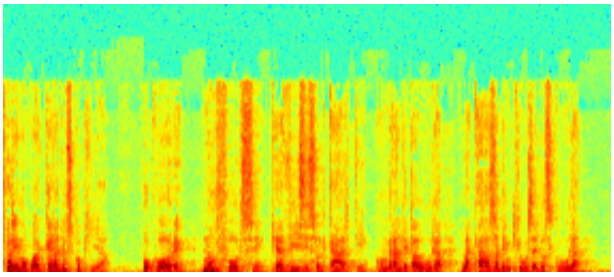

Figure 2: cartographie de la fréquence

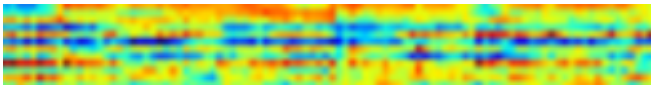

Figure 3: cartographie des MFCC

un premier temps nous allons détailler une structure basique de réseau de neurones ainsi que la technique de *back-propagation* servant à optimiser les coefficients du réseau, avant de nous intéresser à deux structures plus complexes qui ont été conçues dans ce domaine.

4.1 Multi-Layer perceptron

4.1.1 Structure

Il s'agit du type de réseau de neurones le plus simple. Un exemple de réseau de ce type est présenté sur la figure 4. Il comporte trois types de couches :

- Input layer (1,2) : constituent l'entrée du réseau, les coefficients calculés précédemment sont branchés sur les noeuds de cette couche.
- Output layer (5) : constitue la sortie du réseau, le nombre de noeuds correspond au nombre de labels que l'on souhaite obtenir, un seul dans notre cas.
- Hidden layers (3,4) : constituent l'intérieur du réseau, le nombre de ces couches, ainsi que le nombre de noeuds par couche doit être ajusté pour obtenir la meilleure prédiction possible.

Au sein d'une même couche, les neurones ne sont pas reliés entre eux. De plus, les couches successives sont connectées dans un sens uniquement. La quantité arrivant au neurone j est égale à celle de la sortie du neurone i multipliée par un coefficient $W_{i,j}$. Enfin, une fonction d'activation f est appliquée à l'entrée de chaque noeud pour obtenir sa sortie. Voir [1].

4.1.2 L'algorithme de Back-propagation

Étant donné un échantillon de couples entrées / sorties attendues (X_i, A_j), l'objectif est d'adapter au mieux les coefficients $W_{i,j}$ afin que le réseau de neurones produise pour une entrée donnée une sortie aussi proche que possible de celle attendue. L'algorithme de *Back-propagation* permet après plusieurs itérations d'approcher cet objectif [10].

Algorithme de *Back-propagation* :

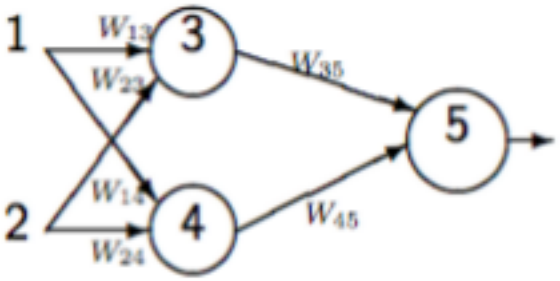

Figure 4: Multi-layer perceptron

1. On initialise les poids $W_{i,j}$ à des valeurs arbitraires
2. Calcul de la sortie du réseau pour les poids actuels.
3. Calcul de l'erreur quadratique totale du réseau E.
4. Calcul des dérivées de E par rapport à chaque poids.
5. Mise à jour des poids par méthode du gradient.
6. Ré-itérer à partir de 2.

4.2 Convolutional Neural Network

4.2.1 Structure

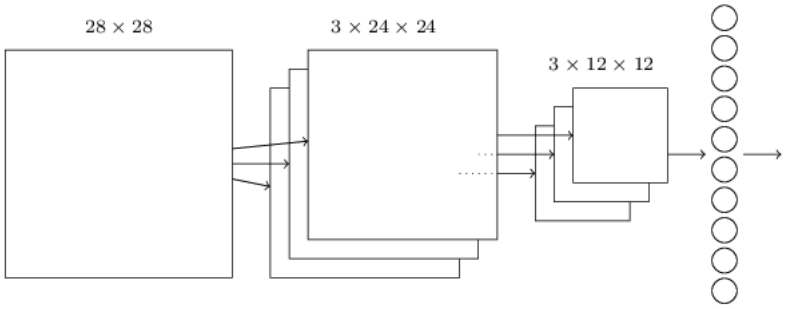

Figure 5: Structure d'un Convolutional Neural Network

À la manière des multi-layer perceptron, les quantités dans un *Convolutional Neural Networks* (CNN) transitent dans un seul sens. Ils sont constitués de plusieurs types de couches appliquées successivement aux données d'entrée [8] :

- Convolutional Layer : Constitue le noyau de la structure. Le principe est de faire glisser un filtre (matrice de taille $n * m$) sur l'ensemble des données. À chaque application du filtre correspond un neurone dans la couche suivante qui recevra le résultat du calcul.
- ReLU Layer : Conserve le même nombre de neurones que la couche qui le précède, applique simplement une fonction de transfert ($seuil$, $tanh\ldots$).
- Pooling Layer : Réduit le nombre de neurones en appliquant des fonctions d'agrégation à des groupes de neurones (max, $mean$, min, $sum\ldots$).
- Fully Connected Layer : Chaque neurone de cette couche est connecté à l'intégralité des neurones de la couche précédente via une liaison pondérée.

Classiquement, un CNN est constitué de plusieurs itérations de cette succession de couches. Un des avantages des CNN est qu'ils sont entraînés rapidement.

4.2.2 Applications

Ils sont souvent utilisés en vision par ordinateur car les étapes de convolution tirent partie de la structure 2D des données et des relations de voisinage. Des chercheurs ont par exemple utilisé un CNN pour classer les images du jeu de données ImageNet [5]. Leur jeu de données était constitué de 1,2 millions d'images d'entraînement, classées dans 1000 catégories. En construisant un réseau avec 5 étapes de convolution suivies de ReLU, ainsi que trois étapes de fully-connected, ils ont réussi à trouver le label des images avec un taux d'erreur de 35%, ce qui est bien meilleur que les méthodes d'apprentissage classiques.

Cependant cette méthode a également été appliquée à de nombreux problèmes de *Natural Language Processing*. Par exemple, des chercheurs ont choisi d'utiliser un CNN pour identifier des locuteurs [6]. En se servant des MFCC comme variables d'entrée, ils ont réussi à obtenir de très bon résultats de classification.

4.3 Recurrent Neural Network

4.3.1 Structure

La caractéristique principale des *Recurrent Neural Networks* est que les neurones internes à une couche peuvent être reliés entre eux. Ils peuvent également être reliés à eux même, et à des neurones de couches précédentes en sens inverse [2]. Cette caractéristique correspond à la prise en compte du temps dans le réseau de neurone : les données introduites à un instant t dans le réseau ré-intégrés dans le calcul des nouvelles données. La forme de RNN la plus utilisée est celle des LSTM [9] qui permettent de ne conserver que les informations les plus pertinentes du passé. Les RNN nécessitent de grande quantité de données pour être entraînés, et peuvent donc s'avérer inefficaces lorsqu'on dispose de trop peu de données.

4.3.2 Applications

Les RNN sont souvent utilisés en *Natural Language Processing* et les applications en reconnaissance vocale sont très prometteuses. Par exemple, des chercheurs ont réussi à obtenir des taux d'erreur très bas sur la reconnaissance de mots à partir de bases de données de plus de 81h d'enregistrements de voix [4]. Nous avons cependant préféré concentrer nos efforts sur l'implémentation d'un CNN

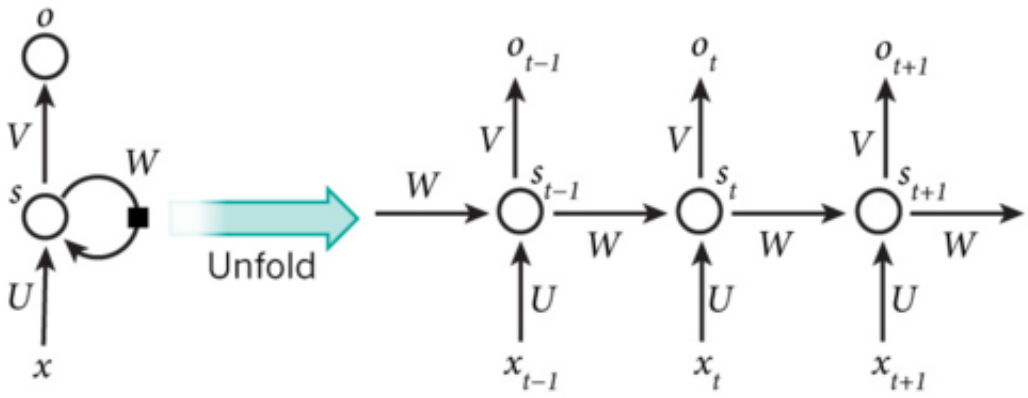

Figure 6: Structure schématique d'un Recurrent Neural Network

durant ce projet, celui-ci pouvant être appliqué facilement aux données 1D mais également aux données 2D comme les cartographies du signal.

5 Mise en oeuvre

5.1 Langage et bibliothèques utilisées

Pour la mise en oeuvre des techniques de traitement du signal et d'apprentissage vues dans les deux paragraphes précédents, nous avons choisi d'utiliser le langage python.

Pour l'apprentissage, nous utilisons :

- La bibliothèque Scikit Learn[1] pour l'implémentation des techniques d'apprentissage classiques. Complète et bien documentée, elle réponds aux besoins classiques d'une mise en oeuvre d'algorithme d'apprentissage
- La bibliothèque Theano[2] pour l'implémentation des algorithmes d'apprentissage profond. Cette bibliothèque présente l'avantage d'être suffisamment bas niveau pour permettre une modularité intéressante.

5.2 Architecture de l'implémentation

Nous présentons principalement deux codes, qui ont été écrit avec l'aide de Kevin Vu, notre tuteur chez AXA.

- Tout d'abord, un code appliquant différentes méthodes d'apprentissage aux ensembles de données classiques (les MFCC, la hauteur,...)

[1] http://scikit-learn.org/stable/
[2] http://deeplearning.net/software/theano/

- Ensuite, nous présenterons le code correspondant à l'approche cartographique des données sonores. Le seul algorithme d'apprentissage utilisé dans ce cas est le *Convolutional Neural Network*, bien adapté à l'analyse d'image.

5.2.1 Première approche : jeux de données classiques

La structure du code présente trois phases principales :

1. ***Preprocessing*** : Il s'agit de la phase de lecture et de calibrage des fichiers audio. Premièrement, un fichier est lu, puis le canal principal est extrait. Il est ensuite sectionné en plusieurs sous-échantillons de taille donnée en argument (permet de gonfler artificiellement le nombre d'échantillons).

   Les fonctions correspondantes sont implémentées dans le fichier *preprocessing.py*.

2. **Calcul des jeux de données** : Nous avons ensuite créé une classe *Dataset*, dont la fonction est de calculer un ensemble de *features* à partir :

   - d'un échantillon (à savoir un *array* de valeurs, correspondant à l'échantillon sonore)
   - et de la liste des noms des *features* à extraire de l'échantillon

   A ce stade, les *features* qui peuvent être calculés sont :

   - La hauteur
   - Les premiers formants
   - Les moyennes de chaque type de MFCC
   - Les écarts-types de chaque type de MFCC
   - L'amplitude moyenne
   - L'écart-type de l'amplitude
   - Les variables relatives au silence
   - Un ensemble de coefficients aléatoires (témoin)

   Cette classe est implémentée dans le fichier *dataset.py*.

3. **Apprentissage et test** : La troisième phase consiste en l'application d'une liste d'algorithmes d'apprentissage à un ensemble de données labélisées et la vérification de la qualité de la prédiction sur un ensemble de test extrait avant la phase d'entraînement. L'ensemble des algorithmes d'apprentissages appliqués est :

   - *Gradient boosting*
   - *Random forest*
   - *Multilayer perceptron*
   - *Convolutional neural network*

   Cette phase est implémentée dans la fonction *apply_to_algo_list_with_a_test_set*, du fichier *apply.py*.

5.2.2 Deuxième approche : cartographie du son

Pour la deuxième approche, le seul algorithme d'apprentissage que nous utilisons est la méthode du *convolutional neural network* en deux dimensions, qui s'applique sur les données transformées en images.

Dans ce cas, il y a deux étapes principales :

- **Chargement des échantillons et transformation en image** : implémenté dans *AudioPictures.py*, il s'agit ici simplement d'extraire plusieurs images d'un enregistrement sonore. Les différents types d'images considérés sont :
  - Le graphe de l'amplitude en fonction du temps
  - Le graphe de la fréquence
  - Le graphe du gain en décibel
  - Le graphe de la puissance
  - Le graphe des MFCC
- **Entrainement et application de l'algorithme d'apprentissage** : La deuxième phase consiste en l'entraînement du *concolutional neural network*, implémenté dans le fichier *ConvNet2D.py*, puis au test de sa précision.

6 Résultats

6.1 Première approche : jeux de données classiques

Dans cette partie, les résultats sont présentés pour un jeu de donnée de 2104 enregistrements de 10 secondes de personnes parlant 9 langues différentes : Français, Anglais, Espagnol, Allemand, Italien, Arabe, Coréen, Polonais, et Roumain. Après découpe des enregistrements, on obtient 8416 échantillons, dont 70% sont utilisés pour l'entraînement, et le reste pour tester les performances des algorithmes. Le but est de trouver les coefficients qui permettent d'identifier au mieux la langue de l'orateur, ainsi que les algorithmes les plus appropriés.

6.1.1 Présentation des résultats

Le tableau 7 présente la précision que nous avons obtenu pour les différents algorithmes évoqués plus haut, et différentes variables extraites de nos échantillons. L'objectif des algorithme est de prédire si l'orateur parle Français ou non.

- Variables :
  - *MFCC* : Plus de 200 coefficients correspondant à 13 MFCC calculés sur plusieurs fenêtres de chaque échantillon
  - *MFCCmv* : 26 coefficients correspondants à la moyenne et à la variance de chaque type de MFCC sur l'échantillon
  - *Form* : Les cinq premiers pics de fréquences dans le spectrogramme de l'échantillon
  - *Rand* : 8 coefficients choisis aléatoirement servant de témoin

- Algorithmes :
  - *Random Forest* : Technique classique d'apprentissage
  - *Grad Boosting* : Technique classique d'apprentissage
  - *MLP* : Multi-layer perceptron avec 3 couches internes
  - *CNN* : Convolutional neural network avec trois couches de convolution en une dimension et deux couches de fully-connected. Cet algorithme n'a été appliqué qu'aux *MFCC* et *MFCC+Form* car il nécessite un nombre suffisant de données en entrée pour être efficace.

	MFCC	Form	MFCC+Form	MFCCmv	MFCCmv+Form	Rand
Random Forest	0.9461	0.9497	0.9556	0.9707	0.9786	0.9012
Grad Boosting	0.9512	0.556	0.9628	0.9758	**0.9813**	0.9107
MLP	0.9346	0.9528	0.9093	0.9758	0.9754	-
CNN	0.9504	-	0.9528	-	-	-

Figure 7: Taux d'échantillons correctement classés en fonction des variables et des algorithmes utilisés

6.1.2 Analyse des résultats

Premièrement, on peut remarquer que les taux d'échantillons correctement classés sont globalement très bons (tous supérieurs à 90%). Cependant, ce constat est également valable pour les échantillons témoins de coefficients aléatoires. Ceci peut s'expliquer par le biais introduit par le jeu de données : seulement un peu plus de 10% des échantillons sont en langue française. Ainsi les algorithmes, pour minimiser l'erreur, choisissent de classer l'intégralité des échantillons test en *non-français*, ce qui donne déjà un bon taux d'erreur. En annexe se trouvent les courbes ROC (Receiver Operating Characteristic) de ces estimateurs. Cette visualisation permet de s'affranchir de ce biais.

Deuxièmement, on peut noter que les deux réseaux de neurones ont des résultats corrects, mais que l'algorithme de Gradient Boosting les surpasse la plupart du temps. Cela est sûrement dû au manque de données. Les réseaux de neurone choisis sont de grande taille et nécessitent beaucoup d'échantillons (et de temps) pour être entraînés.

Enfin, on peut discuter l'influence des variables choisies sur la précision des prédictions. On remarque que les moyennes et variances des MFCC sont celles qui donnent les meilleurs résultats. Cependant, l'ajout des formants permet d'améliorer encore la précision. Nous avons essayé de mettre en évidence la contribution de chaque variable à ce résultat en utilisant la fonction *feature-importance* du classifieur Gradient Boosting de Scikit Learn. La figure 8 montre que les deux variables les plus significatives sont des moyennes de MFCC. Les cinq formants sont tous parmi les variables les plus pertinentes, tandis que les variances ont une influence moindre.

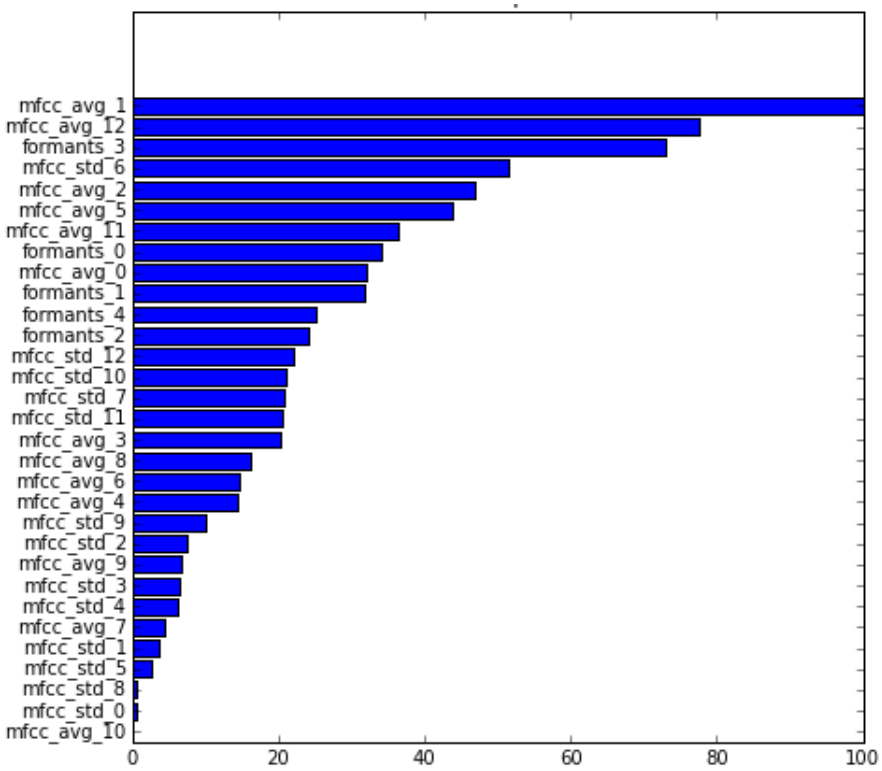

Figure 8: Importance relative des variables dans la prédiction de l'estimateur Gradient Boosting sur le jeu de données *MFCCmv+Form*

6.2 Deuxième approche : cartographie du son

6.2.1 Résultats

Nous présentons ici les premiers résultats dans le cas où le jeu de données est une cartographie extraite de l'échantillon sonore. Dans les cas présentés ci-après, l'ensemble de données est constitué de 307 échantillons dont 219 sont utilisés pour l'entraînement et 88 sont utilisés pour la validation. Précisons que cet ensemble de données est constitué de deux langues, l'espagnol et le polonais.

Dans ce cas, nous obtenons la matrice de confusion suivante (avec en ligne, le label prédit, en colonne, le label réel) :

	Polonais	Espagnol
Polonais	66	0
Espagnol	2	20

La courbe ROC associée est présentée sur la figure 9.

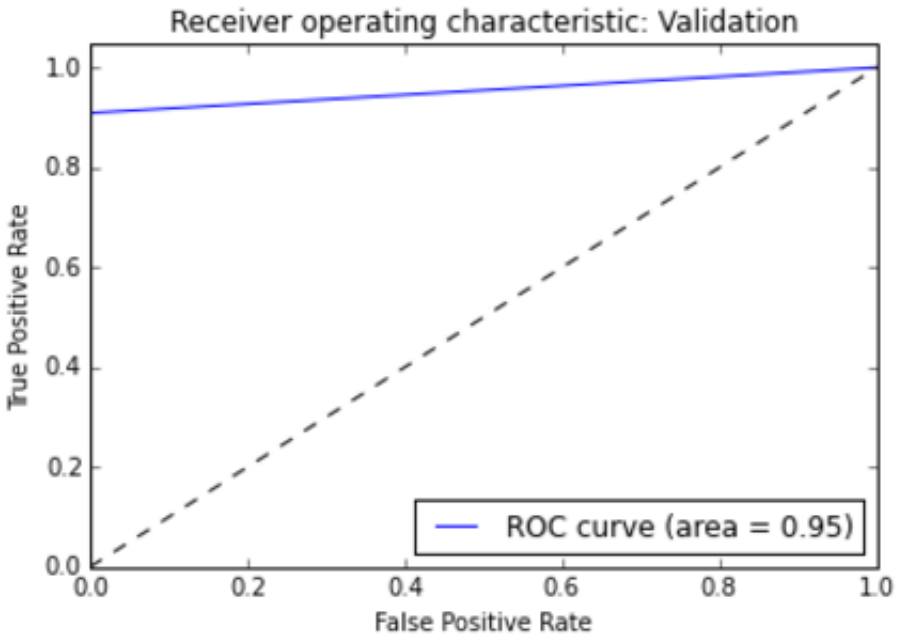

Figure 9: Courbe ROC obtenue dans le cas du CNN à 2 dimensions

6.2.2 Analyse des résultats

Le *convolutional neural network* présente la particularité d'être long à entraîner, par conséquent nous ne sommes pour l'instant pas parvenus à obtenir plus de résultats. Il s'agit donc d'être prudent pour juger de la qualité des résultats observés. De plus, précisons qu'une des limites de cette technique d'apprentissage est la difficulté à analyser les résultats, en particulier les variables prépondérantes. Par conséquent, le risque est que ce réseau de neurones se base sur le locuteur pour faire sa prédiction, plutôt que sur la langue.

Ceci constitue d'ailleurs une approche de détection de fraude : enregistrer une empreinte vocale afin d'identifier un individu sur une liste noire.

6.2.3 Remarque sur l'initialisation de réseau de neurones

Une étape cruciale de l'apprentissage profond est l'initialisation du réseau de neurones. En effet, une mauvaise initialisation peut conduire à des résultats moins bons, la raison étant une saturation des couches inférieures du réseau. Cette théorie a notamment été développée par X. Glorot et Y. Bengio dans [3].

Nous avons vérifié expérimentalement ces résultats en initialisant de deux manières les coefficients du réseau de neurones : dans un premier temps en tirant les coefficients selon une loi normale centrée réduite (ce qui correspond théoriquement à une mauvaise initialisation); dans un deuxième temps selon une loi normale centrée de variance $\frac{1}{n}$ où n est le nombre de coefficients du réseau de neurones. Dans le premier cas, l'erreur initiale (avant entraînement) est environ 3 à 5 fois plus grande que dans le deuxième cas (cf. Fig 10 et Fig 11 respectivement).

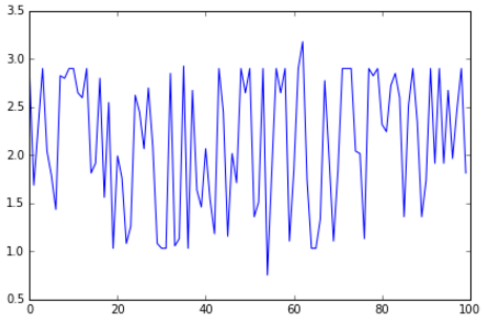

Figure 10: 100 tirages de l'erreur initiale avec une mauvaise initialisation

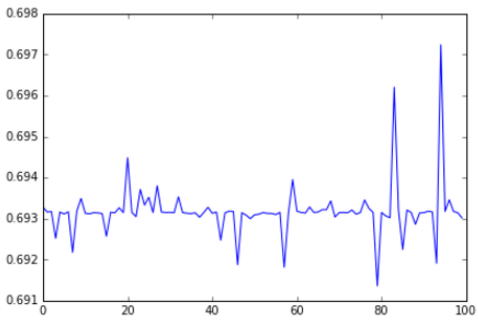

Figure 11: 100 tirages de l'erreur initiale avec une bonne initialisation

7 Avancement du projet

Notre projet étant réalisé en partenariat avec AXA, nous avons fonctionné tout au long de ces quelques mois par rendez-vous successifs dans les locaux de AXA, toutes les deux semaines approximativement. Chaque rendez-vous était l'occasion de faire un point sur ce qui avait été réalisé et de définir les nouveaux objectifs avant le prochain rendez-vous. Le calendrier d'avancement a été le suivant :

- 05/11/15 : réunion de lancement, définition des objectifs et de l'enjeu du projet
- jusqu'au 20/11/15 : prise en main du traitement du signal avec python
- jusqu'au 11/12/15 : état de l'art, recherche de variables significatives dans l'étude de la voix humaine, comparaison des différents types de réseaux de neurones
- jusqu'au 29/01/16 : prise en main de la librairie Theano pour la conception d'un réseau de neurone, et comparaison de différents algorithmes pour la reconnaissance de langue sur des jeux de donnée 1D.
- jusqu'au 20/02/16 : consolidation de nos programmes, étude de la possibilité de données 2D

8 Pistes de développement

Nous avons présenté deux approches dans la recherche de solution du problème. Nous présentons ici quelques pistes, d'une part pour améliorer ces approches, d'autres part pour les compléter :

- Dans le cas de la première approche, il peut être intéressant de rajouter certaines variables. Par exemple, nous n'avons pas considéré la médiane des MFCC et les percentiles. Il peut être intéressant de rechercher l'information qu'ils contiennent. D'autre part, notre fonction qui extrait les formants n'est pas optimale. En effet, nous aimerions obtenir les arguments des k premiers pics du spectre en fréquence de l'échantillon. Or, notre approche est simplifiée, puisque nous nous contentons actuellement de renvoyer les k plus grandes valeurs du tableau contenant les valeurs discrètes du spectre en fréquence. Par conséquent, nous ne détectons

peut-être pas les pics, mais simplement les k premières valeurs maximales, sans que celles-ci soient des maxima. Une solution peut-être d'utiliser une solution d'analyse topologique, en calculant la persistance des pics de la fonctions et en ne conservant que les pics de persistance maximale.

- Nous pouvons également évoquer d'autres approches que celles que nous avons choisies. Par exemple, on peut avoir d'utiliser une technique d'apprentissage non supervisé, en se plaçant dans un espace dont la dimension est égale au nombre de *features*, puis en ajoutant un axe, qui représente la langue. En exagérant les distances inter-langues, cela peut nous permettre de détecter des *clusters* inattendus.

- Citons enfin une dernière possibilité d'ouverture de ce projet. Si l'on revient à l'objectif initial, à savoir la détection de fraude, certaines des approches présentées dans ce rapport devraient être revues. Notamment, une différence fondamentale entre détection de langue et détection de fraude peut être que l'enchaînement joue un rôle beaucoup plus important dans le cas de la fraude. En effet, on peut attendre des schémas de fraudes, qui se lisent dans le temps. Pour prendre en compte cette dimension supplémentaire, il serait intéressant d'implémenter le troisième type de réseau de neurones présenté : le *recurrent neural network*. Celui-ci est en effet capable d'apprendre sur l'enchaînement des variables. On peut également s'attendre à ce que les *features* soient un peu différents. Nous avons évoqué dans ce rapport l'utilisation des silences, qui pourraient venir renforcer l'information contenue dans la voix, tirée des MFCC.

9 Annexe : Courbes ROC pour la première approche

Ci-dessous sont présentées les courbes ROC (*Receiver Operating Characteristic*) de *Random Forest* et *Gradient Boosting* appliqués à deux jeux de données :

1. Coefficients aléatoires
2. Moyenne et variance des MFCC + 5 premiers formants

La courbe ROC représente le taux de vrais positifs en fonction du taux de faux positifs. Alors que les taux d'échantillons bien classés étaient relativement proches (autour de 90% pour des coefficients aléatoires et autour de 97% pour l'autre jeu de données), les courbes ROC mettent en évidence que pour le premier jeu de données, la prédiction est très mauvaise. En effet, une courbe proche de la diagonale signifie que les échantillons classés positivement ont autant de chance d'être effectivement positifs que l'inverse.

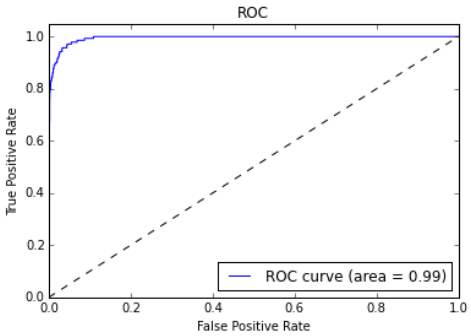

Figure 12: Courbe ROC de Gradient Boosting sur *MFCCmv+Form*

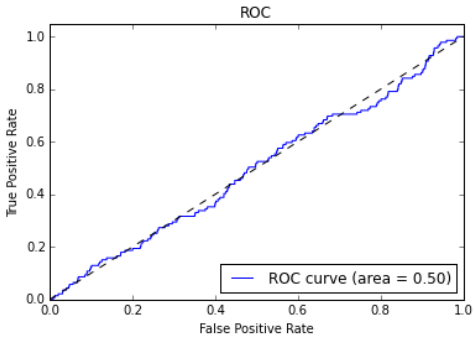

Figure 13: Courbe ROC de Gradient Boosting sur *Rand*

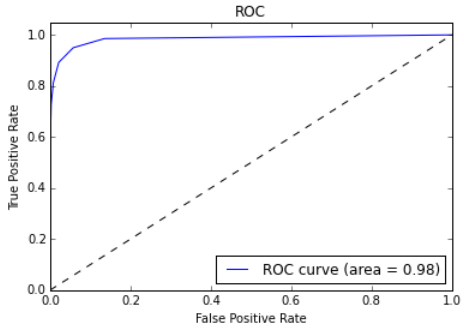

Figure 14: Courbe ROC de Random Forest sur *MFCCmv+Form*

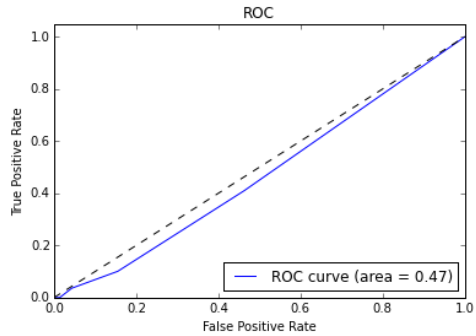

Figure 15: Courbe ROC de Random Forest sur *Rand*

References

[1] Roman Belavkin. "Feed-Forward Neural Networks". http://www.eis.mdx.ac.uk/staffpages/rvb/teaching/BIS3226/hand11.pdf.

[2] Denny Britz. "Recurrent Neural Networks Tutorial, Part 1 – Introduction to RNNs". http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/.

[3] Xavier Glorot and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks". http://machinelearning.wustl.edu/mlpapers/paper_files/AISTATS2010_GlorotB10.pdf.

[4] Alex Graves and Navdeep Jaitly. "Towards End-to-End Speech Recognition with Recurrent Neural Networks". http://www.jmlr.org/proceedings/papers/v32/graves14.pdf.

[5] Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". http://www.cs.toronto.edu/~fritz/absps/imagenet.pdf.

[6] Pham Lee Largman. "Convolutional neural network". http://papers.nips.cc/paper/3674-unsupervised-feature-learning-for-audio-classification-using-convolutional-deep-belief-networks.pdf.

[7] Stéphane Mallat, Eric Moulines, and François Roueff. *Traitement du signal*. Polycopié de cours de l'Ecole polytechnique, 2015.

[8] Michael Nielsen. "Deep learning". http://neuralnetworksanddeeplearning.com/chap6.html.

[9] Christopher Olah. "Understanding LSTM Networks". http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[10] Raul Rojas. "The Backpropagation Algorithm". http://page.mi.fu-berlin.de/rojas/neural/chapter/K7.pdf.

[11] Raunaq Shah and Michelle Hewlett. "Emotion Detection from Speech".

[12] Ergün Yücesoy and Vasif V. Nabiyev. "Gender Identification of a Speaker Using MFCC and GMM". 2013.

[13] Brigitte Zellner. "Caractérisation du débit de parole en français". 1998.