Structures de données et algorithmes

Projet 3: Résolution de problèmes

Pierre Geurts – Jean-Michel Begon – Romain Mormont

20 avril 2017

1 Objectif

On souhaite réaliser une application de reconnaissance vocale dans le but de distinguer et identifier une suite de un ou plusieurs chiffres énoncés oralement. Ca tombe bien, un collègue vous expliquait justement qu'une méthode qui semble bien fonctionner pour déterminer si deux échantillons sonores correspondent au même contenu vocal consiste à prendre la transformée MFCC (Mel-frequency cepstral coefficients) des deux signaux et puis à vérifier si les signaux transformés sont proches en utilisant un algorithme appelé *dynamic time warping* (DTW). Cet algorithme calcule une distance entre signaux qui, paraît-il, prend en compte les variations de vitesses et de longueurs entre les échantillons sonores correspondant à un même contenu vocal.

Cela vous inspire la méthode suivante pour reconnaître des suites de chiffres :

1. On calcule la transformée MFCC de la séquence sonore à analyser, ce qui donne une série temporelle multivariée.
2. On essaie de trouver le découpage optimal en chiffres de cette série.
3. Pour chaque partie obtenue du découpage, on identifie le chiffre correspondant en la comparant à l'aide de l'algorithme DTW à une base de données de référence contenant des échantillons sonores isolés correspondant aux différents chiffres.

Afin d'obtenir les meilleurs taux de reconnaissance possibles, il est important d'intégrer le score d'identification des chiffres dans la procédure de découpage optimale, c'est-à-dire d'effectuer conjointement les deux dernières étapes.

Plus formellement, soit une séquence de N valeurs $s[1\,..\,N]$ avec $s[i] \in \mathbb{R}^m$ (correspondant à la transformation MFCC de la séquence sonore, en m coefficients) et soit une base de données $\mathcal{S}$ de référence, correspondant aux transformations MFCC de chiffres isolés. Le problème consiste à trouver le nombre de chiffres k et le découpage de la séquence en k sous-séquences $s[i_j\,..\,i_{j+1}-1]$ $(j = 1, \ldots, k)$, avec $i_j < i_{j+1}$, $i_1 = 1$ et $i_{k+1} = N + 1$ tels que pour tout j :

$$l_{min} \leq i_{j+1} - i_j \leq l_{max} \tag{1}$$

et tels que la somme suivante soit minimale :

$$\min_k \min_{o_1, o_2, \ldots, o_k \in \mathcal{S}} \sum_{j=1}^{k} \text{dtw}(s[i_j\,..\,i_{j+1} - 1], o_j), \tag{2}$$

où $\text{dtw}(x, y)$ correspond à la dissimilarité entre le signal x et le signal y telle que calculée par l'algorithme DTW. La contrainte (1) permet d'imposer que les parties de signal correspondant

aux différents chiffres ne soient ni trop courtes ni trop longues. Notez que le nombre k de chiffres de la séquence est supposé inconnu et il doit être déterminer de manière à minimiser (2). Une fois le minimum de (2) trouvé, la séquence de chiffres correspondant à l'échantillon sonore sera obtenue en récupérant la séquence des chiffres correspondant aux échantillons $o_1, o_2, \ldots, o_k$ de la base de référence qui réalise le minimum de (2).

Le projet se divise donc en deux parties :

1. La comparaison de deux signaux à l'aide de l'algorithme DTW.
2. Le découpage optimal de la séquence en chiffres.

Notez que le DTW doit pouvoir calculer une distance entre deux éléments des signaux. Nous vous proposons d'utiliser la distance absolue moyenne :

$$d(s_1[i], s_2[j]) = \frac{1}{m} \sum_{h=1}^{m} \mid s_1[i]^{(h)} - s_2[j]^{(h)} \mid \tag{3}$$

2 Analyse théorique

2.1 DTW

On vous demande de vous renseigner sur le *Dynamic time warping* avec contraintes de localité et de répondre dans votre rapport aux questions suivantes :

1. Expliquez brièvement le principe de cet algorithme.
2. Discutez de l'intérêt des contraintes de localité.
3. L'algorithme DTW est basé sur la programmation dynamique. Donnez la formulation récursive correspondante.
4. Donnez la complexité de l'algorithme en fonction des longueurs des signaux comparés et de la contrainte de localité.

Remarque : n'oubliez pas de citer vos sources !

2.2 Découpage optimal

L'algorithme de détermination de la découpe optimale se base également sur la programmation dynamique. La fonction à calculer par programmation dynamique sera notée $M(n)$ et représentera le score obtenu par le découpage optimal de la sous-séquence $s[1\,..\,n]$. $M(N)$ sera ainsi la valeur minimale de (2) sur tous les découpages et toutes les valeurs de k.

Dans votre rapport, on vous demande de répondre aux questions suivantes :

1. Formulez $M(n)$ de manière récursive en précisant bien le cas de base. *Suggestion :* Partez du problème de la découpe de la tige d'acier et modifiez le pour prendre en compte une contrainte sur les tailles des morceaux. Ensuite, ajoutez le coût lié au DTW.
2. Déduisez-en le pseudo-code d'un algorithme efficace pour calculer cette découpe.
3. Analysez la complexité au pire et au meilleur cas de votre algorithme en fonction des paramètres les plus appropriés.

3 Analyse empirique

Afin de valider votre implémentation, on vous demande de réaliser les expériences suivantes et de reporter et analyser les résultats dans le rapport :

1. Pour vérifier d'abord la pertinence du DTW pour l'identification de chiffres isolés, pour chacun des *échantillons de test*, déterminez *l'échantillon de référence* dont il est le plus proche. Reportez le nombre d'erreurs de reconnaissance parmi les 50 échantillons de test en fonction de la contrainte de localité. Par erreur de reconnaissance, on entend le nombre d'échantillons de test dont l'échantillon de référence le plus proche ne correspond pas au bon chiffre.
2. Pour vérifier le bon fonctionnement de l'algorithme de découpage, on vous demande de l'appliquer pour déterminer les séquences de chiffres correspondant aux 8 fichiers MFCC que nous vous avons fournis, en utilisant comme référence les mêmes échantillons de référence que pour l'expérience précédente. Précisez et justifiez les valeurs de paramètres que vous avez utilisées (l_{min}, l_{max} et contrainte de localité).
3. (Bonus) On se propose d'étudier la robustesse de l'approche par rapport au locuteur. Pour cela, enregistrez 5 échantillons sonores par chiffre de l'un d'entre vous ainsi que 8 échantillons de nombres à 4 chiffres et reproduisez les expériences 1 et 2 avec les modifications suivantes :

   (a) Pour l'expérience 1, calculez le nombre d'erreurs en utilisant vos 5 échantillons par chiffre comme références et nos 5 derniers échantillons par chiffre comme échantillons de test et ensuite en utilisant nos 5 premiers échantillons par chiffre comme référence et vos 5 échantillons par chiffre comme échantillons de test.

   (b) Décodez nos 8 séquences de chiffres en utilisant vos 5 échantillons comme référence et décodez vos 8 séquences de chiffres en utilisant nos 5 premiers échantillons par chiffre comme référence.

   Concluez par rapport à la robustesse de l'approche.

Pour la question bonus, vous devez vous enregistrer et puis transformer les signaux. Vous pouvez utiliser la bibliothèque python `librosa` (`https://github.com/librosa/librosa`) pour la transformée MFCC. Dans toutes vos expériences, nous vous suggérons de fixer le nombre m de coefficients des transformée MFCC à 13.

4 Implémentation

Nous vous demandons d'implémenter les fonctions suivantes :

`dtw` qui calcule la distance DTW de deux signaux. Cette fonction est spécifiée dans le fichier `dynamicTimeWarping.h`.

`predictDigit` qui calcule score DTW minimum entre un signal et les signaux de la base de données de référence. Cette fonction est spécifiée dans le fichier `predictDigit.h`

`bestSplit` qui calcule le découpage optimal. Cette fonction est définie dans le fichier `splitSequence.h`.

Afin de vous aider, nous vous fournissons les fichiers suivants :

`LinkedList.{h,c}` : une implémentation de liste liée.

`Signal.{h,c}` : les fichiers qui définissent les signaux à manipuler.

`digitRecognizer.c :` un fichier `main` qui traite de la reconnaissance d'un chiffre individuel.

`sequenceRecognizer.c :` un fichier `main` qui traite de la reconnaissance d'une séquence de chiffres.

`refDB :` les signaux constituant la base de données de référence. Celle-ci doit être un dossier contenant 10 sous-dossiers nommés de 0 à 9. Tous les fichiers `.mfcc` d'un sous-dossier seront chargés par la fonction `parseDatabase`.

`testDB :` les signaux correspondant à un chiffre isolé pour tester votre implémentation.

`sequences :` les signaux à analyser.

`wav2mfcc.py` un fichier python se basant sur `librosa` pour calculer les transformées MFCC pour les questions bonus.

5 Deadline et soumission

Le projet est à réaliser **par groupe de deux** pour le **19 mai 2017, 23h59** au plus tard. Il doit être soumis via la plateforme de soumission (`http://submit.montefiore.ulg.ac.be/`).

Le projet doit être rendu sous la forme d'une archive `tar.gz` contenant :

1. Votre rapport (7 pages maximum) au format PDF. Soyez bref mais précis et respectez bien la numération des (sous-)questions.
2. Les fichiers `dynamicTimeWarping.c`, `predictDigit.c` et `splitSequence.c`.

Vos fichiers seront évalués avec les flags habituels (`--std=c99 --pedantic -Wall -Wextra -Wmissing-prototypes - DNDEBUG`) sur les machines ms8xx en substituant adéquatement l'algorithme de recherche. Ceci implique que :

— Les noms des fichiers doivent être respectés.
— Le projet doit être réalisé dans le standard C99.
— La présence de *warnings* impactera négativement la cote finale.
— Un projet qui ne compile pas sur ces machines recevra une cote nulle (pour la partie code du projet).

Un projet non rendu à temps recevra une cote globale nulle. En cas de plagiat avéré, l'étudiant se verra affecter une cote nulle à l'ensemble des projets.

Les critères de correction sont précisés sur la page web des projets.

Bon travail !