Université Paul Sabatier TOULOUSE III

M2R – IRR

BE 2 : Traitement Automatique de la Parole

Partie I : Programmation Dynamique

Reconnaissance de la parole par Dynamic Time Warping (DTW ou Programmation Dynamique)

La Reconnaissance Automatique de la Parole a pour objectif de retrouver le message prononcé dans un signal acoustique. Ce message peut correspondre à une séquence de phonèmes, à un mot, à une phrase, etc.

Dans cette étude vous réaliserez une RAP en mots isolés dans un contexte de vocabulaire restreint.

La méthode de reconnaissance utilisée est fondée sur un alignement temporel dynamique ou programmation dynamique.

1. Système de reconnaissance en mots isolés par DTW

Le système de reconnaissance en mots isolés a pour but de mettre en correspondance un signal acoustique inconnu *O* avec un dictionnaire de références $\{R_1, \ldots, R_n\}$. La reconnaissance est achevée en retenant le mot du dictionnaire le plus proche du signal inconnu au sens du critère de distance *D* choisie.

L'algorithme d'alignement temporel dynamique (ou programmation dynamique) calcule le chemin permettant d'aligner temporellement deux séquences numériques distinctes ainsi que la distance séparant les deux séquences.

Soit $O = \{O_1, \ldots, O_k\}$ et $R = \{r_1, \ldots, r_p\}$ deux séquence numériques de longueurs différentes. Dans un premier temps la matrice des distances locales entre les deux vecteurs est calculée pour chaque couple (o_i, r_j). Dans la pratique la distance euclidienne est souvent utilisée.

A partir de la matrice de distances locales, l'algorithme de programmation dynamique établit le parcours minimisant la distance totale entre les vecteurs *O* et *R* selon le critère d'optimisation suivant :

$$c_{ij} = \min \begin{Bmatrix} c_{i,j-1} + d_{i,j} \\ c_{i-1,j-1} + 2 * d_{i,j} \\ c_{i-1},j + d_{i,j} \end{Bmatrix}$$

C'est à dire que la minimisation du chemin total peut être décomposée en succession de minimisations locales.

2. Mise en œuvre d'un système de reconnaissance de mots

Présentation du système

Le système de reconnaissance en mots isolés que vous allez mettre en œuvre se décompose en :

- une étape d'acquisition du signal audio ;
- une étape d'extraction des vecteurs de paramètres acoustiques ;

- une DTW qui permet de calculer la distance entre le mot prononcé et un mot du dictionnaire ;
- une étape de décision ;
- l'affichage du mot reconnu sur le terminal de l'utilisateur ;
- la possibilité de visualiser les résultats de la DTW sur l'ensemble du dictionnaire afin d'évaluer la robustesse de la reconnaissance.

Dictionnaire

Le dictionnaire sera composé des mots nécessaire pour faire une calculatrice vocale (l'idéal serait de prendre en compte les chiffres de 0 à 99). Elle devra, au minimum, comprendre les mots suivants : plus, moins, multiplié, divisé, un deux, trois, quatre cinq, six, sept, huit, neuf, dix.

Spécifications

Vous développerez ce système sous Matlab, en prenant en compte les spécifications suivantes :

- des signaux d'entrée et du dictionnaire seront à enregistrer au format wav.
- les paramètres acoustiques sont des paramètres cepstraux calculés sur des fenêtres de 30ms toutes les 15 ms à l'aide de la fonction rceps() de Matlab. Les fenêtres glissantes sont gérées grâce à la fonction buffer(). La fonction réalisant une extraction des vecteurs acoustiques ***extractionCoeffCepstraux()*** vous est fournie.
- les vecteurs acoustiques des signaux de référence ne sont calculés qu'une seule fois puis stockés dans une matrice.

Réalisation

Vous devez implémenter les fonctions ***dtw(), decision(), visualisation(), affichage() et recoglobale()***

- La fonction ***dtw()*** prend deux arguments en entrée : la matrice de coefficients cepstraux du signal à reconnaître et la matrice de coefficients cepstraux d'un signal de référence. Cette fonction renvoie la mesure du coût total, la matrice de coûts et la matrice des chemins possibles. Vous utiliserez la distance euclidienne entre chaque couple de vecteurs de paramètres acoustiques. Le développement de cette fonction pourra s'appuyer sur ***l'algorithme de programmation dynamique***. La visualisation de la matrice des coûts pourra se faire par l'appel suivant : `figure, imagesc(chemin), xlabel('Test'), ylabel('Ref')`
- La fonction ***decision(),*** prend en entrée la liste des distances calculées pour l'ensemble du dictionnaire et la liste de mot du dictionnaire et renvoie la distance minimale et le mot associé.
- La fonction ***visualisation()*** permet de visualiser la robustesse de la reconnaissance en comparant les mesures des distances calculées par ***dtw()***.
- La fonction ***recoglobale()*** réalise tous les couples de calcul de distance et appelle la fonction de décision et d'affichage et de visualisation.

Vous devez également enregistrer une occurrence des mots par chaque membre de votre binôme. Une mise en commun de tous les mots enregistrés par tous les étudiants est possible : veuillez nommer les mots de cette façon là : nommot_prenom_2014.wav. Ils seront alors mis a disposition sous moodle.

Partie II : Modèles de Markov cachés

Reconnaissance de la parole par modèles de Markov Cachés (HMM : Hidden Markov Model)

Vous devez maintenant réaliser l'application réalisée en partie I en utilisant une modélisation des mots par modèles de Markov cachés. Vous utiliserez pour cela la boite à outil HTK (Hidden Tool Kit distribué par http://htk.eng.cam.ac.uk/).

1. Des scripts ont été écrits pour vous faciliter l'utilisation du toolkit. Cette partie du BE va se dérouler sous Linux. Vous pouvez les installer en exécutant la commande suivante dans un terminal : `~farinas/installM2IRR` : cela va installer dans le répertoire courant un dossier `BE_TAP_HMM/` contenant tout le nécessaire pour réaliser les manipulations.
2. Vous devez dans un premier temps déterminer la topologie de vos modèles de mots par HMM. Je vous conseille d'utiliser un état émetteur par état stable au niveau du signal. Pour les probabilités de transition, initialisez-les à 0.8 quand vous rebouclez sur l'état et 0.2 vous passez à l'état suivant. Ensuite créer un fichier par modèle dans le répertoire `Modeles/gabarits/` (nom du fichier = nom du mot en toutes lettres). Pour vous aider, vous pouvez regarder les modèles fournis : 3etats, 4etats, 5etats, 6etats, 7etats.
3. Ensuite il est nécessaire de réaliser l'initialisation de vos modèles (script `Bin/initialisation`) puis de réaliser un apprentissage (script `Bin/apprentissage`).
4. Effectuez ensuite un décodage des mots de la liste de test (script `Bin/reconnaissance`). Calculez ensuite des scores sur la reconnaissance (script `Bin/scores`)
5. Modifiez la grammaire pour pouvoir prendre en compte la connexion des mots et réaliser des phrases correspondant à des commandes pour la calculette. Enregistrez des fichiez de test, en marquant une pause marquée entre les mots. Le modèle de pause vous est fourni et est a intégrer dans la grammaire.

Réalisez un rapport détaillant le fonctionnement des modèles de Markov cachés. Intéressez-vous en particulier aux questions suivantes :

- Quelle est la paramétrisation qui est utilisée ?
- Quels sont les caractéristiques des lois attachées aux états ?
- Comment est réalisée l'initialisation, l'apprentissage et la reconnaissance des HMM ?
- Que faudrait-il faire pour réaliser une modélisation phonétique des mots ?