

Sprachverarbeitung: Übung 13

Spracherkenner mit Mustervergleich

In dieser Übung geht es darum, einen einfachen Worterkenner zu verwirklichen, der nach dem Prinzip des Mustervergleichs funktioniert. Als Muster wird die aus dem Sprachsignal ermittelte Folge von Mel-Cepstren¹ verstanden.

Wenn man Sprachsignale aufnimmt, dann sind auf der Aufnahme in Sprechpausen Hintergrundgeräusche vorhanden. Auch wenn diese Geräusche noch so leise sind, so sind sie doch vorhanden. Sie sind nicht vom Sprachsignal abhängig und können insbesondere bei jeder Aufnahme völlig verschieden sein und somit ganz unterschiedliche Merkmalsvektoren ergeben. Beim Vergleichen von Sprachmustern besteht deshalb das Problem, dass, auch wenn das Sprachsignal in den zu vergleichenden Mustern dasselbe Wort enthält, die Distanz zwischen den Mustern wegen unterschiedlicher Hintergrundgeräusche gross sein kann.

Um den Mustervergleich robuster zu machen, wird deshalb die lokale Distanz mit der lokalen Intensität² der beiden betreffenden Analyseabschnitte gewichtet. Dadurch wird der Einfluss der Hintergrundgeräusche stark gemindert. Die in dieser Übung verwendeten Sprachmuster umfassen also stets eine Folge von Mel-Cepstren und eine gleich lange Folge von RMS-Werten.

Aufgabe 1: Generieren von Referenzmustern

Grundsätzlich kann aus einem einzigen Sprachsignal mit einem Wort ein Referenzmuster für dieses Wort erzeugt werden: Es sind einfach die Mel-Cepstren und die RMS-Werte zu ermitteln. Dafür steht die Matlab-Funktion `[cep,rms] = extr_mfcc_and_rms(x,fs,nfilt,w,shift,ncep)` zur Verfügung.

Es hat sich jedoch gezeigt, dass ein Worterkenner weniger Fehler macht, wenn ein Referenzmuster nicht aus einem einzigen, sondern aus mehreren Sprachsignalen generiert werden. Weil beim Sprechen der Zufall im Spiel ist, kann aus mehreren Sprachsignalen ein Muster erzeugt werden, das ein Wort im Mittel besser repräsentiert.

Es stellt sich somit die Frage, wie aus mehreren Sprachsignalen ein Muster erzeugt werden kann. Selbstverständlich können nicht einfach die Merkmalssequenzen der Signale gemittelt werden, weil diese meistens unterschiedlich lang sind. Es muss eine Zeitnormalisation so durchgeführt werden, dass die zeitlich normierten Merkmalssequenzen übereinstimmen, d.h. die Merkmalssequenzen sollen gleich lang sein und auch die Laute zeitlich übereinstimmen. Dann kann durch

¹Die Mel-Cepstren werden hier ohne den nullten Koeffizienten verwendet, damit sie von der Lautstärke unabhängig sind.

²Als lokale Intensität gilt der RMS-Wert (root mean square) des Analyseabschnittes des Signals.

Mittelung über die zeitnormalisierten Merkmalssequenzen ein robustes Referenzmuster erzeugt werden.

Um die zu mittelnden Merkmalssequenzen der Signale gleich lang zu machen, müssen bei der Zeitnormalisation mit dem DTW-Algorithmus *asymmetrische* Pfaderweiterungen eingesetzt werden. In der Matlab-Funktion `[dst,warpcurve,distcurve] = dtw(ref,tst,refwgt,tstwgt)` werden deshalb die Pfaderweiterungen von Abbildung 12.4c im Buch auf Seite 360 verwendet.

Damit aus mehreren Signalen ein Referenzmuster erzeugt werden kann, muss also zuerst eine der Merkmalssequenzen als Zeitbasis ausgewählt werden. An diese Zeitbasis sind dann die anderen Merkmalssequenzen zeitlich anzugleichen. Eine gute Möglichkeit dieser Wahl ist, diejenige Merkmalssequenz als Zeitbasis zu nehmen, deren Distanzen zu allen anderen Merkmalssequenzen im Mittel am kleinsten sind. Der asymmetrischen Pfaderweiterungen wegen sind also bei K gegebenen Signalen für ein Wort $K * (K - 1)$ Distanzen zu ermitteln.

Das Matlab-Skript `ueb13_1_frame.m` (ist im Directory **Gegebenes** zu finden) generiert aus den Sprachsignalen³ `ueb13_sigs/sigX.Y.wav` die Referenzmuster `refpattX.mat`, wobei X von 1 bis 10 geht, was den Ziffern 1 bis 9 und 0 entspricht, und Y nummeriert die Sprachsignale mit demselben Wort. Das Matlab-Skript `ueb13_1_frame.m` verwendet jedoch beim Generieren nur ein Signal pro Referenzmuster, d.h. nur $Y = 1$.

Erweitern Sie das Matlab-Skript `ueb13_1_frame.m` so, dass zum Generieren eines Referenzmusters `refpattX.mat` alle vorhandenen Signale `ueb13_sigs/sigX.*.wav` verwendet werden, so wie dies oben erläutert worden ist. Generieren Sie sodann die Referenzmuster für die Ziffern 0 bis 9.

Aufgabe 2: Erkennung isolierter Wörter

Wenn Sie Aufgabe 1 gelöst und die Referenzmuster erzeugt haben, dann können Sie diese im Worterkennungsprogramm `ueb13_2.m` einsetzen. Dieses Matlab-Programm liest die Sprachsignaldatei `u13_test.wav` mit einem zu erkennenden Wort ein und extrahiert daraus mit der Funktion `extr_mfcc_and_rms` die für den Vergleich benötigten Merkmale (das sogenannte Testmuster). Zudem wird das Signal als Figur 1 dargestellt.

Danach wird in einer Programmschleife nacheinander jedes der Referenzmuster `refpatt1.mat` bis `refpatt10.mat` (Ziffern 1 bis 9 und 0) geladen, das Test- und das Referenzmuster mittels der Funktion `dtw` zeitlich einander angepasst und die Distanz bestimmt. Die Warping-Kurve wird in Figur 2 dargestellt, zusammen mit den Intensitätsverläufen des Test- (links) und des Referenzsignals (darunter) und mit dem Verlauf der gewichteten (vergl. oben) lokalen Distanz entlang der Warping-Kurve (ganz unten).

Das Referenzmuster mit der kleinsten Distanz (mittlere Distanz entlang der Warping-Kurve) zum Testmuster gilt als das erkannte Wort.

Mit den in Aufgabe 1 generierten Referenzmustern sollte aus dem Signal `u13_test.wav` die Ziffer 7 erkannt werden. Wenn dies der Fall ist, dann können Sie im Programm `ueb13_2.m` die mit `%>>` auskommentierte Zeile zum Aufnehmen eines Testsignals aktivieren und so das Programm als Online-Worterkenner einsetzen.

³Diese Signale können mit den Matlab-Befehlen `[s,fs] = audioread('Gegebenes/ueb13_sigs/sigX.Y.wav');` und `sound(s,fs)` abgespielt werden.

Zum Aufnehmen eines Testsignals erscheint im Figur-Fenster 3 eine einfache Benutzerschnittstelle. Sie können damit ein Sprachsignal, also eine gesprochene Ziffer aufnehmen, überprüfen und – wenn Sie mit der Aufnahme zufrieden sind – abspeichern. Das Signal steht dann in der Datei `test.wav` dem Erkenner zur Verfügung.

Hören Sie vor dem Abspeichern die Aufnahme an und kontrollieren Sie, ob das ganze Wort, aber keine Störungen davor oder danach aufgenommen worden sind. Achten Sie auch darauf, dass die Aufnahme nicht zu starke Blasgeräusche aufweist (Tip: das Mikrophon nicht direkt vor den Mund, sondern seitlich davon positionieren). Der Ziffernerkennung wird sonst oft Fehler machen.

Aufgabe 3: Verbessern des Worterkenners

Wie eingangs erwähnt, funktioniert ein Worterkennung auf der Basis eines Mustervergleichs nur dann zuverlässig, wenn die Referenzmuster und das zu erkennende Signal von derselben Person gesprochen worden sind. Dies ist bei der Anordnung von Aufgabe 2 nicht der Fall. Um den Worterkennung zu verbessern, können Sie deshalb die Referenzmuster für Ihre eigene Stimme generieren. Gehen Sie wie folgt vor:

- Nehmen Sie mit dem Matlab-Skript `ueb13_3.m` Sprachsignale der zu erkennenden Wörter auf. Hören Sie vor dem Abspeichern das aufgenommene Wort jeweils an und kontrollieren Sie wiederum, ob die Aufnahme das ganze Wort, aber keine Störungen davor oder danach enthält. Die aufgenommenen Signale werden im Subdirectory `ueb13_sigs_own` gespeichert.
- Ersetzen Sie im Matlab-Skript `ueb13_1_frame.m` die Zeile `dirnam = 'ueb13_sigs';` durch `dirnam = 'ueb13_sigs_own';`.
- Generieren Sie nun mit dem in Aufgabe 1 erweiterten Matlab-Skript `ueb13_1_frame.m` aus den aufgenommenen Signalen die neuen Referenzmuster.
- Testen Sie den Worterkennung mit `ueb13_2.m` aus.