2.4 Topologie der HMMs

Bevor HMM mit HTK initialisiert werden können, braucht man sog. Prototyp-Modelle, die die Struktur der HMM für jedes Phonem festlegen. Das Design der Prototyp-Modell stellt für den Benutzer eine fast unübersichtliche Freiheit an Gestaltungsmöglichkeiten dar. Die wesentlichsten Freiheitsgrade sind:

- Anzahl der emittierenden States
- Erlaubte Übergänge
- Anzahl der Mixturen (Gauß-Glocken) pro State
- Modellierung (Art) der Gaußglocken
- Anzahl der 'Daten-Ströme' (streams)

Die Anzahl der States (zusammen mit den erlaubten Übergängen) ist ein wichtiger Faktor bei der zeitlichen Modellierung: HMMs mit mehr States 'konsumieren' auch mehr Merkmalsvektoren (Frames) bei der Erkennung. Auf diese Weise kann eine Minimal-Länge für bestimmte Segmente erzwungen werden.

Frage: Wieviele States muß ein normales 'Links-Rechts-HMM' haben damit das Modell eine minimale Verweildauer von 30 msec hat?

Die Anzahl der Mixturen pro State entscheidet, ob mehr-gipflige Wahrscheinlichkeitsverteilungen besser modelliert werden können, stellt aber auch ein Problem dar, wenn zuwenig Trainingsmaterial verfügbar ist. Das Gleiche gilt für die Art der Modellierung: volle Kovarianz-Matrizen erlauben schräg liegende Gaußglocken, erfordern aber weitaus mehr Trainingsdaten zur Abschätzung.

Die Anzahl der Ströme bietet die Möglichkeit, den Merkmalsvektor zu strukturieren, so daß separate Teile in jeweils eigenen Gaußglocken modelliert werden. Ein solches Vorgehen wird in der Literatur auch bisweilen 'product code' genannt. Der Vorteil liegt darin, daß die einzelnen Ströme eine geringere Dimensionalität haben als alle Merkmale in einem Vektor und somit der zu modellierende Vektorraum kleiner wird. Andererseits ist dies nur dann erlaubt, wenn die Komponenten der einzelnen Ströme statistisch voneinander unabhängig sind.

Hier wollen wir pragmatisch vorgehen und legen einfach fest:

- Struktur: klassisches Links-Recht-Modell
- 1 Strom mit 39 Komponenten
- Anzahl der States: 2 für kurze Laute; 3 für gelängte Vokale; 4 für Diphtonge und Pausen und Müll
- 1 Mixture pro State
- Volle Kovarianz-Matrix

Daraus folgt, daß wir 3 Prototyp-Modelle brauchen: 2state, 3state und 4state.

Öffnen Sie das aktuelle HTK-Buch auf und studieren Sie das Beispiel in Fig. 7.4 (S. 108):

```
% acroread ~/HTK-KURS/Kursunterlagen/HTKBook_3.2.pdf
```

Dann erzeugen Sie sich ein Verzeichnis, in dem Sie Ihre Prototypen ablegen wollen:

```
% cd ~/HTK-KURS/GROUP#
% mkdir proto
% cd proto
```

Kopieren Sie eine Beispiel-Datei 'proto_2state' mit einem HMM mit zwei Zuständen in das Verzeichnis.

```
cp ../proto_2state proto
```

Gehen Sie in einen Editor Ihrer Wahl (am besten 'gedit') und verändern Sie das Prototypen-File namens 'proto' so, dass es drei verschiedene HMM-Prototypen enthält, die die Namen '2state', '3state' und '4state' erhalten. Beachten sie die oben angegebenen Spezifikationen. Denken Sie daran:

- Zustände werden mit 2 beginnend nummeriert, weil HTK den ersten und letzten Zustand als 'virtuell' reserviert. Entsprechend hat ein HMM mit 2 Zuständen den Eintrag '<NumStates 4>' und eine 4x4 Übergangsmatrix.
- VecSize = 39
- VecType = MFCC_E_D_A
- alle Werte sind beliebig, außer daß die Zeilen der Transitions-Matrix sich zu 1.0 addieren müssen.
- Geben Sie in den inversen Kovarianzmatrizen in der Diagonale 1.0 und in den Nebenelementen 0.2 ein.

Tips: Erstellen kopieren Sie das bereits vorhandene HMM '2state' mit Copy and Paste, und erweitern Sie dann die Kopie auf drei Zustände usw.

Wir können prüfen, ob unsere Struktur von HTK akzeptiert wird, indem wir die Prototypen in den HMM-Editor HHEd laden:

```
% touch dummy.hed
% cat > proto.list
2state
3state
4state
^D
% HHEd -H proto -w dummy.mmf dummy.hed proto.list
```

(Die Zeichensequenz ^D bedeutet, daß man die Tastenkombination "Strg-D" verwendet.)

Wenn die Struktur nicht stimmt, wird HHEd eine Fehlermeldung liefern. (HHEd wird später noch ausführlich besprochen!)

```
-----------------------------------------------------------------
SIGNALISIEREN SIE DAS ENDE DER UEBUNG
-----------------------------------------------------------------
```

2.5 Bootstrapping und Viterbi-Training

Wir wollen nun für jedes phonetische Modell in der Liste PHONEME ein entsprechendes HMM aus den Trainingsdaten initialisieren und trainieren. Dieser Prozeß wird 'Bootstrapping' genannt.

Der Befehl 'HInit' führt dies für uns aus. Im Einzelnen macht HInit folgendes:

- Kopiert das vorgesehene Prototyp-Modell in ein Modell mit dem Namen der phonetischen Klasse
- liest alle Segmente mit diesem Namen aus dem MLF und dann die korrespondierenden Daten aus den vorverarbeiteten htk Files
- Teilt jedes Segment linear in soviele Unterabschnitte, wie das HMM emmitierende States hat.
- Schätzt aus allen Unterabschnitten den Mittelwerts-Vektor und die Kovarianzmatrix und schreibt diese Werte in die Modell-Struktur
- Läßt über alle Segmente einen Viterbi-Aligner laufen, der die Grenzen zwischen den Unterabschnitten optimiert.
- Schätzt aus den neuen Unterabschnitten den Mittelwerts-Vektor und die Kovarianzmatrix und schreibt diese Werte in die Modell-Struktur
- Wiederholt die letzten beiden Schritte solange, bis die Gesamt- Wahrscheinlichkeit des Modells in Bezug auf alle Segmente konstant bleibt (Konvergenz) oder eine maximale Anzahl von Wiederholungen (Iterationen) erreicht ist.
- Schreibt die letzten Schätzwerte in ein neues MMF

Die letzten 4 Schritte nennt man auch 'Viterbi training'. Dieser Schritt ist bereits relativ rechenintensiv.

Die Befehle zur Initialisierung des Modells /_2:/ lauten also:

```
% cd ~/HTK-KURS/GROUP#
% mkdir hmm1
% nice HInit -i 10 -l '_2:' -H proto/proto -o '_2:' -v 0.0001 \
? -I TRAIN.mlf -S ../TRAIN.slist -M hmm1 -A -T 1 3state
```

Während der Prozeß läuft, lesen Sie bitte die Erläuterungen über die Optionen und Argumente von HInit:

```
-i #   : max. Anzahl der Iterationen im Viterbi training
-l str : Segmentname, nach denen im MLF gesucht wird
-H nam : Filename des HMM-Makro-Files (MMF) mit den Prototyp-Modellen
-o str : Name des Modells im Output (Makro $\sim$h)
-v #   : minimale Varianz oder Kovarianz
-I nam : Filename des MLF mit den Segmentierungen des
         Trainingsmaterials
-S nam : Filename des Skriptfiles mit den Signalfilenamen des
         Trainingmaterials
-M nam : Name des Directories, in dem das neue Modell gespeichert wird
-A     : Wiederhole den Befehl auf stdout (für Logging)
-T #   : Trace Level: gib Informationen über deine Arbeit
2state : Einziges Argument: Name des Prototypen, der verwendet
         werden soll
```

Der Befehl HInit erzeugt im Zieldirectory hmm1 ein File, das genauso heißt wie das Prototypen-File, also 'proto'. Dieses muß man nun umbennen in den entsprechenden Phonem-Namen:

```
% mv hmm1/proto hmm1/_2:
```

Schauen Sie sich das Modell an:

```
% less hmm1/_2:
```

Was fällt Ihnen an den Übergangswahrscheinlichkeiten auf?

Jetzt haben wir erst ein Phonem initialisiert. Es empfiehlt sich daher die ganzen 44 Initialisierungsbefehle in ein Shell-Skript zu schreiben und nacheinander automatisch abzuarbeiten. Ein solches Shell-Skript steht in Ihrem Gruppen-Directory in 'INIT'. Rufen Sie es NICHT auf, sondern schauen Sie es sich an und versuchen Sie die Befehle zu verstehen.

```
% less INIT
```

Wie Sie sehen, kann dieses Shell-Skript wie folgt aufgerufen werden:

```
% cat init.script | ./INIT
```

Dazu brauchen Sie aber zunächst ein File 'init.script', welches pro Zeile den Eintrag:

```
'label' TAB 'prototyp-name'
```

enthält. Also z.B.:

```
@ 2state
OY 4state
usw.
```

Achtung: Das Pausenmodell '<p:>' sollte 4 states haben! Ebenso das 'Müllmodell '<usb>'. Schreiben Sie sich das entsprechende 'init.script' und starten Sie den Prozeß als Hausaufgabe (Erzeugung dauert ein paar Stunden!) in den Hintergrund:

```
% cd ~HTK-KURS/GROUP#
% cat init.script | ./INIT >& INIT.log &
```

Sie können den Fortschritt des Bootstraps anhand des Logfiles INIT.log überprüfen. Geben dazu ein:

```
% less INIT.log
```

und dann den Großbuchstaben 'F' (Verlassen mit Strg-C und 'Q')

Schauen Sie sich die 'Average LogP' der Modelle '_2:' und '<usb>' an. Warum sind diese Wahrscheinlichkeiten so extrem unterschiedlich?

```
################################################################
```

Wenn Ihre Hausaufgabe fehlerfrei funktioniert hat, sollte sich jetzt im Subdir 'hmm1' ein Set von Phonem-HMM befinden.

```
% ls hmm1
```

Wieviele sind es? Welche Modelle wurden nicht erzeugt? Warum nicht? Überprüfen Sie diese und klären Sie ev. Fehler.

Bei der Erzeugung unserer Aussprache-Lexica haben wir noch einige zusätzliche 'Phoneme' verwendet, die bisher nicht in der Phonem- Liste PHONEME vorkamen: /br:/ (Atmen), /la:/ (Lachen) und /<nib>/ (nicht-sprachlicher Bereich). Der Grund hierfür ist uns jetzt klar: Für diese Modelle haben wir keine Segmentierung in der für das Bootstrapping verwendeten MAUS-Spur. Jetzt wollen wir diese fehlenden Modelle ergänzen. Da wir (vorerst) keine Segmente zum Bootstrappen haben, kopieren wir einfach die nächst ähnlichen Modelle und benennen sie um:

```
% cd ~/HTK-KURS/GROUP#
% cd hmm1
% cp h '<nib>'
% cp '<usb>' '<la:>'
% cp h '<br:>'
```

Editieren Sie jetzt jedes der neuen Modelle und ändern Sie die Makro-Definition '∼h' in den neuen Namen, z.B.

```
~h "h"  =>  ~h "<nib>"
```

Nun muß noch die Liste der Modelle PHONEME erweitert werden. Machen Sie sich dazu eine Kopie von ../PHONEME in Ihrem Gruppen-Dir und editieren Sie diese Kopie. Bei allen zukünftigen Aufrufen verwenden Sie nur noch Ihre eigene Kopie. Kontrolle:

```
% wc -l PHONEME
```

muß jetzt 47 ergeben.

Die Modelle können als einzelne Files weiterverwendet werden. Für die zukünftige Arbeit ist es jedoch handlicher, das ganze Set in einem MMF zusammenzufassen (ganz analog wie wir die Label-Files in einem MLF zusammengefasst haben). Das geht mit dem HMM-Editor HHEd:

```
% cd ~/HTK-KURS/GROUP#
% touch dummy.hed
% HHEd -w hmm1/VM.mmf -d hmm1 dummy.hed PHONEME
% less hmm1/VM.mmf
```

Kontrolle:

```
% grep '~h' hmm1/VM.mmf | wc -l
```

muß ebenfalls 47 ergeben.

Damit haben wir ein erstes Set von HMM, das für einen Erkenner verwendet werden kann.

```
-----------------------------------------------------------------
SIGNALISIEREN SIE DAS ENDE DER UEBUNG
-----------------------------------------------------------------
```