Automatische Spracherkennung

3 Vertiefung: Drei wichtige Algorithmen – Teil 2

Soweit vorhanden ist der jeweils englische Fachbegriff, so wie er in der Fachliteratur verwendet wird, in Klammern angegeben.

Beispiele von Computerkommandos und Skizzenanweisungen sind in `Schreibmaschinenschrift` wiedergegeben.

3.2 Statistische Modelle – HMM

Um das Problem der mangelnden Generalisierung beim direkten Pattern-Match zu umgehen, wurde versucht, den Sprachprozess selbst als statistische Quelle zu modellieren. Siehe dazu die Diskussion der bedingten Wahrscheinlichkeit und des statistischen Ansatzes der Spracherkennung mit Bayes-Formel im Abschnitt 2, Teil 3.

Im folgenden werden einige mögliche statistische Modelle zur Modellierung der Akustik (Term $p(s|W)$ in der Bayes-Formel) mit steigender Komplexität skizziert. Das Problem der Verkettung von linguistischen Klassen zu Ketten (W) wird hier vorerst nicht betrachtet, nur die akustische Modellierung für eine linguistische Klasse, im Folgenden L genannt. Ziel ist also die Modellierung des Ausdrucks $p(s|L)$.

3.2.1 Einfache 'single-state'-Modelle - GMM

In der Literatur werden solche einfachen Modelle auch oft 'gaussian mixture models' (GMM) genannt. Sie werden hauptsächlich in der Sprecherverifikation und der Erkennung von Sprechereigenschaften (z.B. Geschlecht) verwendet.

Prinzip GMM:

Für jede Klasse L (Phonem, Silbe, ...) wird ein statistisches Modell $p(s|L)$ trainiert, beim Test wird das unbekannte Muster mit jedem Modell verglichen und das Modell mit der höchsten Wahrscheinlichkeit ausgewählt (Max-Regel).

Im Gegensatz zu bisherigen Verfahren (z.B. DTW) müssen hier also keine größeren Mengen von Mustervektoren abgespeichert werden, sondern unabhängig von der Trainigsdatenmenge nur genau ein Modell[1] pro Klasse L.

Training:

Es wird für jede Klasse eine *mehrdimensionale und multi-variate Wahrscheinlichkeitsdichtefunktion* $p(s|L)$ (WDF) aus dem Trainingsmaterial berechnet. Üblicherweise modelliert man diese WDF

[1] Ein Modell wird beschrieben durch seine Modellparameter wie Mittelwerte.

durch gewichtete Addition von mehreren Gaußverteilungen; die einzelnen Gaussfunktionen einer multi-variaten WDF nennt man auch 'Moden' (modes).[2]

```
Skizze:

Gelabelte
Merkmals-  ->  Clusterung  ->  Mittelwert + Varianz  ->  Moden
vektoren                       f. jedes Cluster

    gewichtete
->  Summe der    =   WDF  p(s|L)     L: Lautklasse
    Moden                            s: Merkmalsvektor
```

Kleiner Exkurs in die Statistik: Gaussverteilung

Eindimensional: Nur ein Merkmal, 1 Mode = Normalverteilung

$$p(x|P) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\frac{(x-m)^2}{\sigma^2}}$$

mit:

x : eindimensionale Zufallsgröße

P : Mode-Nummer (Bedingung)

M : Anzahl der Beobachtungen von x im Training

m : Mittelwert $m = \frac{1}{M}\sum_{i=1}^{M} x_i$

σ : Standardabweichung $\sigma^2 = \frac{1}{M-1}\sum_{i=1}^{M}(x_i - m)^2$

Mehrdimensional: Merkmalsvektor mit N Merkmalen, 1 Mode = mehrdimensionale Normalverteilung

Statt x (scalar) $\rightarrow \vec{x}$ (Vektor, hier: Merkmalsvektor),
statt Standardabweichung $\sigma \rightarrow$ Kovarianzmatrix C,
statt Mittelwert $m \rightarrow$ Mittelwertsvektor $\vec{m}$

[2] Man muss nicht notwendigerweise Gaußfunktionen verwenden, aber diese haben sich sehr bewährt.

$$p(\vec{x}|P) = \frac{1}{\sqrt{(2\pi)^N |C|}} e^{-\frac{1}{2}(\vec{x}-\vec{m})' C^{-1} (\vec{x}-\vec{m})}$$

mit:

$\vec{x}$: Merkmalsvektor der Dimension N

P : Mode-Nummer (Bedingung)

M : Anzahl der Beobachtungen von $\vec{x}$ im Training

$\vec{m}$: Mittelwertsvektor $\vec{m} = \frac{1}{M} \sum_{i=1}^{M} \vec{x}_i$

C : Kovarianzmatrix $= \begin{vmatrix} \sigma_{11} & \sigma_{21} & \cdots \\ \sigma_{12} & \sigma_{22} & \cdots \\ \cdots & & \sigma_{NN} \end{vmatrix} = \frac{1}{M-1} \sum_{i=1}^{M} (\vec{x}_i - \vec{m})(\vec{x}_i - \vec{m})'$

σ_{11} : Varianz in der ersten Dimension (Quadrat der Standardabweichung)

σ_{12} : Kovarianz zwischen erster und zweiter Dimension

$|C|$: Determinate von C

C^{-1} : Invertierte (Inverse) von C

Da es sich um eine Wahrscheinlichkeitsdichtefunktion (WDF) handelt, muss das Integral über alle möglichen Zufallsvektoren $\vec{x}$ gleich 1 sein:

Normierung: $\int_{\vec{x}} p(\vec{x}|P) \equiv 1$ für jedes P

Ein Mode ist eine mehrdimensionale Funktion mit Integral 1, welche die Wahrscheinlichkeit für das Auftreten eines Merkmalsvektors $\vec{x}$ berechnet, unter der Voraussetzung, dass $\vec{x}$ von diesem Mode P stammt (bedingte Wahrscheinlichkeit; vgl. Kapitel 'Bayes' in Abschnitt 2, Teil 3)

Die WDF ist die gewichtete Summe (Überlagerung) mehrerer solcher Moden, wobei die Gewichte g_p (Summe = 1) dafür sorgen, dass die Normierungsbedingung auch hier eingehalten wird. Die WDF drückt die Wahrscheinlichkeit aus, dass ein Merkmalsvektor $\vec{x}$ von einer bestimmten Lautklasse L stammt.

WDF : $p(\vec{x}|L) = \sum_{p=1}^{P} g_p p(\vec{x}|p)$

Normierung der Gewichte : $\sum_{p=1}^{P} g_p \equiv 1$

Beispiel:

WDF für die Klasse L ='ja' mit zwei gaußförmigen Moden:

$$p(\vec{x}|L) = 0.67 \frac{1}{\sqrt{(2\pi)^N |C_1|}} e^{-\frac{1}{2}(\vec{x}-\vec{m_1})' C_1^{-1} (\vec{x}-\vec{m_1})} + 0.33 \frac{1}{\sqrt{(2\pi)^N |C_2|}} e^{-\frac{1}{2}(\vec{x}-\vec{m_2})' C_2^{-1} (\vec{x}-\vec{m_2})}$$

Einfacher statischer GMM-Erkenner:
Sei $L = 2$ (2 Klassen, z.B. 'ja' und 'nein'), $N = 2$ (Dimensionalität der Merkmalsvektoren, z.B. die Merkmale 'energy' und 'zero crossing') und M('ja') = 9, M('nein') = 8 (Anzahl der beobachteten Merkmalsvektoren pro Klasse) und $P = 2$ (Anzahl der Moden/Gaußglocken pro Klasse).

```
      Skizze:

9 gelabelte
 Merkmals- -> Clusterung -> Mittelwert+Varianz -> 2 Moden -> Gewichte 4/9 5/9 ->
 vektoren    in 2 Cluster    f. jedes Cluster    Klasse 'ja'
   'ja'        (4 + 5)      (m11,C11) (m12,C12)

-> WDF p(x|'ja') = 4/9 gauss(m11,C11) + 5/9 gauss(m12,C12)


8 gelabelte
 Merkmals- -> Clusterung -> Mittelwert+Varianz -> 2 Moden -> Gewichte 2/8 6/8 ->
 vektoren    in 2 Cluster    f. jedes Cluster   Klasse 'nein'
  'nein'       (2 + 6)      (m21,C21) (m22,C22)

-> WDF p(x|'nein') = 2/8 gauss(m21,C21) + 6/8 gauss(m22,C22)
```

D.h. beim Training clustern wir die beobachteten Merkmalsvektoren *innerhalb jeder Klasse* in zwei Cluster (z.B. mit LBG), berechnen aus jedem Cluster mit Mittelwertsvektor und Kovarianzmatrix zwei Moden und bilden aus diesen Moden für jede Klasse eine WDF.

Test (Erkennung)

Die Merkmalsvektoren des unbekannten Sprachsignals $X = (\vec{x}_1, \vec{x}_2, \cdots, \vec{x}_K)$ mit K Vektoren werden nacheinander einzeln in die WDFs beider Klassen eingesetzt (d.h. die Formel der WDF wird für einen Merkmalsvektor ausgerechnet) und man erhält für jeden Merkmalsvektor zwei bedingte Wahrscheinlichkeiten, dass er von dem einen oder dem anderen Modell stammt. Dann werden einfach alle bedingten Wk des einen Modell aufmultipliziert und genauso die des anderen Modells. (Die Annahme dabei ist, dass die einzelnen Merkmalsvektoren statistisch unabhängige Beobachtungen sind.)

$$p(X|L = \text{'ja'}) = \prod_{k=1}^{K} p(\vec{x}_k|L = \text{'ja'})$$

und

$$p(X|L = \text{'nein'}) = \cdots$$

Man erhält also für jedes Modell $L =$ 'ja' und $L =$ 'nein' eine Gesamtwahrscheinlichkeit, dass die beobachtete Vektorfolge X von ihm stammt. Zuletzt wird das Modell (und damit die Klasse) als erkannt ausgewählt, das die höhere Wk hat (MAX-Entscheidung).

Eigenschaften GMM

- statistische Modelle generalisieren viel besser als Muster (Warum?)
- der Rechenaufwand (beim Test) ist unabhängig von der Anzahl der Trainingsvektoren
- der Speicherbedarf für das Modell ist konstant und unabhängig von der Anzahl der Trainingsvektoren
- relativ robust
- Nachteil: Die Reihenfolge der Vektoren (also die Sprachdynamik) wird nicht berücksichtigt

Beispiel aus der Praxis : Sprecherverifikation

Sprecherverification (speaker verification) bedeutet die Überprüfung einer Sprecheridentität anhand einer Sprachprobe: ein Sprecher muss sich bei einem Sprecherverifikations-System (SV) registrieren, indem er eine Sprachprobe abgibt (i.d.R. 1-3 Sätze), mit welchen dann ein Modell der Sprache dieses Sprecher berechnet wird (Training). Im Testfall gibt der Sprecher vor, eine bestimmte Identität zu haben, und wird dann anhand einer angegebenen Stimmprobe identifiziert (zugelassen) oder abgelehnt.

Am Institut für Phonetik regelt ein solches System dem Zugang zur Bibliothek. Es verwendet pro registriertem Sprecher zwei GMM: ein GMM wird auf die Trainingsstichprobe des Sprechers trainiert, das andere auf die Sprache von allen anderen registrierten Sprechern (sog. Weltmodell). Im Testfall berechnet zunächst das Sprecher-GMM die Wahrscheinlichkeit, dass die Sprachprobe von dem Sprecher der vorgegebenen Identität stammt. Dann berechnet das Weltmodell die Wahrscheinlichkeit, dass die Spracheprobe NICHT vom Sprecher mit der vorgegebenen Identität stammt. Dann werden beide Wahrscheinlichkeiten verglichen: ist die Wk des Sprecher-GMMs höher, wird der Sprecher zugelassen, und die Türe öffnet sich; andernfalls wird er abgewiesen.

Die Sprache wird in diesem SV-System mit 16kHz abgetastet, so dass das Sprachspektrum von 0-8kHz gut abgedeckt ist. Es werden 12 MFCC und die log. Energie aus einem Fenster der Breite 20msec und alle 5msec berechnet. Die GMM bestehen aus 20 Gaußglocken mit diagonalisierten Kovarianzmatrizen (d.h. die Gaußverteilungen liegen achsenparallel; Kovarianzen zwischen den Merkmalen werden nicht modelliert).

In einer Simulation mit 12 Sprechern ergab sich eine mittlere 'False Acceptance Rate' (d.h. ein fremder Sprecher wird irrtümlich zugelassen) und eine mittlere 'False Rejection Rate' (d.h. ein echter Sprecher wird irrtümlich abgewiesen) von jeweils 0,7%.

3.2.2 Statistischer Erkenner mit Dynamik - HMM

Das im vorangegangenen Abschnitt geschilderte statistische Modell GMM lässt sich so erweitern, dass die typische Dynamik eines Lauts, d.h. die Abfolge der Merkmalsvektoren über der Zeit in der statistischen Bewertung mit berücksichtigt wird.

Wiederholung der Grundlagen

Ein HMM ist ein Graph mit Zuständen ('states') und Übergängen ('transitions'). Ein voll vernetztes HMM nennt man *ergodisch* und hat alle Übergänge, die möglich sind (auch alle Selbstübergänge). Ist die Zahl der möglichen Übergänge eingeschränkt, spricht man meistens z.B. von sog. 'Links-Rechts-Modellen'

```
Skizze:

Klassisches ergodisches und links-rechts-Modell
```

Jeder Übergang ist mit einer Übergangs-Wahrscheinlichkeit a_{xy} vom Zustand x zum Zustand y belegt. Die Summe aller von einem Knoten abgehenden Übergangswahrscheinlichkeiten muss 1 sein (logisch: irgendwohin muss man ja gehen).

In jedem Zustand wird ein statistisches Modell gespeichert, das in Abh. des Merkmalsvektor $\vec{x}$ eine bedingte WDF modelliert. In diesem Fall ist die WDF nicht nur durch das Modell L (Klasse) bedingt, sondern auch noch durch den Zustand Q innerhalb des Modells L.

$p(\vec{x}|L, Q)$ beschreibt also die Auftretenswahrscheinlichkeit für $\vec{x}$ unter der Bedingung, dass es Zustand Q des Modells L passiert. Im Allgemeinen wird diese WDF durch eine gewichtete Summe von Gaußglocken (Moden) wie oben besprochen realisiert.

Es gibt zu jedem HMM einen *Einsprungs-Vektor* $\vec{e}$, der die Wahrscheinlichkeiten enthält, dass das Modell in einem bestimmten Zustand betreten wird. Entsprechend gibt es auch einen *Aussprungs-Vektor*, der die Wken enthält, dass das Modell aus einem bestimmten Zustand verlassen werden kann.
Beim Links-Rechts-Modell kann oft nur der erste Zustand betreten und der letzte verlassen werden.

Es gibt Algorithmen, die ...

- ... für eine Sammlung von Merkmalsvektorfolgen einer Klasse (z.B. eines Phonems) das optimale HMM (Übergänge und WDFs) berechnen.
  Diesen Vorgang nennt man 'Training' des HMM. Die beiden wichtigsten Trainingsalgorithmen sind
  - *Baum-Welch* : alle möglichen Pfade durch das HMM werden berücksichtigt

  - *Viterbi-Training* : nur der jeweils wahrscheinlichste Pfad durch das HMM wird berücksichtigt (Näherung)

- ... für eine einzelne Merkmalsvektorfolge $X = (\vec{x}_1, \vec{x}_2, \cdots, \vec{x}_K)$ die sog. *Erzeugungswahrscheinlichkeit* $p(X|L)$ berechnen. Diesen Vorgang nennt man auch 'Test' des HMM. Die beiden wichtigsten Test-Algorithmen sind
  - *Forward-Backward* : alle möglichen Pfade durch das HMM werden berücksichtigt
  - *Viterbi-Decoding* : nur der jeweils wahrscheinlichste Pfad durch das HMM wird berücksichtigt (Näherung)

Das Viterbi-Decoding ist anschaulich leichter zu verstehen: Beim Viterbi-Decoding berechnet sich die Erzeugungswk $p(X|L)$ aus dem Produkt der Übergangswk längs des Pfades durch das HMM und den WDFs in den dabei durchlaufenen States in Bezug auf die dabei zugeordneten Merkmalsvektoren. Der Pfad, d.h. die Zuordnung von Merkmalsvektoren auf die Zustände wird dabei so gewählt, dass die Erzeugungswahrscheinlichkeit maximal wird. Der Viterbi-Algorithmus muss also nicht nur die Wken aufmultiplizieren, sondern vor allem auch den *besten* Pfad finden. Dieses Problem ist sehr ähnlich wie das Problem beim bereits besprochenen Dynamic Time Warping.

HMM-Design in der ASR

Für jeden Sprachlaut/Silbe/Wort benötigt man ein links-rechts-HMM mit typischerweise 3 - 20 Zuständen. Die Anzahl der Zustände bestimmt implizit die mittlere Verweildauer in einem Modell. Dadurch können z.B. Mindestdauern für bestimmte Laute erzwungen werden.

```
Skizze:

Links-rechts-Modelle verschiedener Länge, min. Verweildauern
```

Die Anzahl der Zustände ist manchmal auch motiviert durch die mehr oder weniger bekannte intrinsische Dynamik des zu modellierenden Sprachlauts.

Skizze:

Beispiel Plosiv mit drei Zuständen; Diphthong mit 5 Zuständen

Die Suche nach dem optimalen Pfad durch ein HMM kann mit Hilfe einer *trellis*-Darstellung anschaulich gemacht werden:

Skizze:

HMM mit drei Zuständen, Vektorfolge mit 6 Vektoren: trellis

In jeden Zustand Q des HMM wird eine WDF wie im vorherigen Abschnitt über GMM beschrieben gespeichert. Auch in HMM werden am häufigsten multi-variate und multi-dimensionale Gaussfunktionen verwendet. Die Wahl der Dimensionalität (Merkmale) und der Anzahl der Moden bestimmt wie beim GMM die Zahl der freien Parameter des Gesamtsystem.

Beispiel

Zustände: 48 Phoneme mit je 3 Zuständen = 144 Zustände
Merkmale: 12 Mel-Cepstral-Koeffizienten + Energie + 1. Ableitung + 2. Ableitung = 39 Merkmale (Dimension)
Moden: WDFs mit je 8 Gaußglocken

Parameter pro Mode:	Mittelwertvektor	39
	Kovarianz-Matrix 39x39	1521
	Summe	1560
Parameter pro Zustand:	8 Moden x 1560	12480
Parameter pro HMM:	3 Zustände x 12480	37440
	9 Übergangswk	9
	Summe	37449
Parameter des System:	48 Modelle x 37449	1797120

D.h. um das oben definierte System zu trainieren, müssen knapp 2 Mio Parameter bestimmt werden. Da es sich um statistische Parameter handelt, rechnet man mit mindestens 100-1000 mal

so vielen Trainingsdaten, um diese einigermaßen sicher abschätzen zu können. Das entspricht in unserem Beispiel einer Gesamtdauer von 128 Stunden Trainingssprache.

Training eines HMM (nur anschaulich)

Bei Training von HMM wird im Prinzip wie folgt vorgegangen:

1. Ein Stück Sprachsignal aus dem Trainingsmaterial (Segment) wird per Labelung einem bestimmten HMM zugeordnet. Die Labelung und Segmentierung kann per Hand oder auch durch ein automatisches Verfahren erfolgen.

   `Skizze:`

   `Aufteilung des Trainingskorpus auf die Klassen`

2. Jedes Segment wird zunächst in Q (Anzahl der States) gleich lange Teile zerschnitten und die Teile jeweils einem State zugewiesen. Die WDF in jedem State wird dann aus dem so gesammelten Material berechnet (Clusterung, Mittelwerte und Kovarianzen, Gewichte). Die Übergangswahrscheinlichkeiten des HMM werden beliebig vorbesetzt. Diesen Vorgang nennt man *Initialisierung* oder *bootstrapping* der HMM.

   `Skizze:`

   `HMM mit drei Zuständen, 2 Vektorfolgen mit 6 und 8 Merkmalsvektoren`

3. Der Trainingsalgorithmus nimmt nun wieder alle Segmente dieser Klasse und berechnet alle möglichen (Baum-Welch) oder nur die statistisch beste (Viterbi) Zuordnung von Merkmalsvektoren zu Zuständen für jedes Segment im Trainingsmaterial (eine solche Zuordnung wird

auch 'Bester Pfad' oder 'alignment' genannt). Dabei verwendet er zur Bewertung der unterschiedlichen möglichen Pfade die WDF und die Übergangswk der Bootstrap-HMM. Dies wird für alle Trainingsdaten ('Segmente') dieses Modells wiederholt und die Zuordnung von Merkmalsvektoren pro Zustand ('mapping') abgespeichert.

```
Skizze:

HMM mit drei Zuständen, Vektorfolge mit
   6 Merkmalsvektoren, lattice innerhalb des HMM
```

4. Dann schätzt der Algorithmus aus den zugeordneten Merkmalsvektoren die Parameter des HMM neu ab. Diese sind:
   + Mittelwertsvektoren und Kovarianzen der Moden
   + Gewichte der einzelnen Moden
   + Übergangswahrscheinlichkeiten

Die Punkte 3 und 4 werden iterativ wiederholt und die zunehmende Performanz des Modells nach jeder Iteration an einer unabhängigen Stichprobe (development test set) kontrolliert. Wenn die Erkennungsleistung stagniert oder sogar absinkt, wird der Trainingsprozess abgebrochen.

Trainingsalgorithmen gibt es in vielen verschiedenen Variationen dieses allgemeinen Prinzips. Immer aber ist es das Ziel, möglichst 'sinnvolle' (i.S. der Maximierung der Wahrscheinlichkeit) Zuordnungen von Merkmalsvektoren zu Zuständen zu erreichen und aus den dabei beobachteten Daten die Parameter der Zustände neu abzuschätzen. Nach jeder Iteration muss die Gesamt-Wahrscheinlichkeit der Modelle gegeben das Material gestiegen sein (Konvergenzkriterium des Trainingsalgorithmus). Das bedeutet aber nicht, dass auch die Gesamtwahrscheinlichkeit auf dem unabhängigen development test set steigen muss! (Warum?)

Performanzbetrachtung

Bei sehr viel Trainingsmaterial und komplizierten HMM würde die vollständige Berechnung aller möglichen Pfade durch ein Modell bei gegebener Vektorfolge viel zu lange dauern.

Beispiel

Ein HMM mit drei Zuständen (ergodisch) und ein Segment mit 20 Merkmalsvektoren ergibt schon 3486784401 mögliche Pfade, die berechnet werden müssen. Typisches Trainingsmaterial besteht aus Millionen von Segmenten, und man beachte, dass alle diese Berechnungen bei jeder Iteration neu durchgeführt werden müssen.

Daher verwenden Algorithmen wie der Baum-Welch und der Viterbi rekursive Berechnungen von Zwischenwerten in jedem Punkt der Trellis, d.h. der Aufspannung aller möglichen Pfade in einem HMM. Der anschauliche Hintergedanke ist der, dass sehr viele Pfade natürlich über bestimmte 'Teilstrecken' durch das Modell identisch sind, und daher nicht jedesmal neu berechnet werden müssen. Durch Ausnutzung dieser Redundanz wird es möglich, alle Pfade durch ein HMM wirklich vollständig zu berücksichtigen (Baum-Welch) bzw. den besten Weg durch ein Modell überhaupt zu ermitteln (Viterbi). Wie wir jetzt sofort erkennen, ist also der Viterbi-Algorithmus auch nur eine Variante der Dynamischen Programmierung.

```
Skizze:

Beispiel trellis, Ermittlung des besten Pfads mit DP-Technik
```

Test eines HMM (nur anschaulich)

Im Prinzip passiert beim Test das Gleiche wie im ersten Teil des Trainings mit dem Unterschied, dass jetzt *jedes* Modell versucht, den besten (Viterbi) oder die Summe über alle möglichen (Forward-Backward) Pfade für ein gegebenes Stück Signal zu berechnen (beim Training wurde das nur im *zugeordneten* Modell durchgeführt). Beim Viterbi bestimmt das Produkt aller WKs entlang dieses Pfades die Gesamtwahrscheinlichkeit $p(X|L)$ des Signalstücks X gegeben das Modell L.

```
Skizze:

Einfacher Phonemerkenner mit nur drei HMM, Aufspannen der trellis
```

Um den Suchraum zu begrenzen, werden Einschränkungen ('constraints') beim Erweitern der Lattice gemacht. Z.B. können bestimmte Phonemfolgen einfach nicht vorkommen (Phonotaktik) oder es werden minimale und maximale Verweildauern in einem Segment festgelegt, die nicht unter- bzw. überschritten werden dürfen (explizit time modelling).

Trotz all dieser Einschränkungen ist der Suchraum immer noch so groß, dass er in vertretbarer Zeit nicht vollständig abgesucht werden kann. Daher wird die Lattice von links nach rechts zeitsynchron aufgebaut und in jedem Zeitschritt nur eine bestimmte Anzahl von möglichen Suchpfaden (die besten) weiterverfolgt. Diesen Vorgang nennt man 'pruning' oder 'beam search' (man sucht nur innerhalb eines bestimmten Suchstrahls, eben eines 'beams'). Auch bei den Pruning-Techniken gibt es eine Vielzahl von Ansätzen; alle haben das Ziel, möglichst wenig Pfade zu untersuchen und dabei den besten Pfad nicht zu verlieren.

Eigenschaften von HMM

- Sehr gute Generalisierung (Sprecher, Dynamik, etc.)
- Robust gegen zeitliche Variationen
- Einfaches Training (links-rechts-Modelle sind 'gutartig')
- Sehr rechenintensives Training
- Wenig rechenintensiver Test (mit pruning)
- Viel Trainingsmaterial nötig, da viele freie Parameter
- Komplizierter Test für fließende Sprache (Warum?)
- Gefahr der Überadaption