universität wien

MAGISTERARBEIT

Titel der Magisterarbeit

Large Vocabulary Continuous Speech Recognition Systems, Maximum Mutual Information Estimation and Switching Regimes

Verfasser

Markus Cozowicz

Angestrebter akademischer Grad

Magister der Sozial- und Wirtschaftswissenschaften (Mag.rer.soc.oec.)

Wien, im Mai 2009

Studienkennzahl lt. Studienblatt:	A 066 951
Studienrichtung:	Magisterstudium Statistik
Betreuer:	O. Univ.-Prof. Dr. Georg Ch. Pflug

Contents

Acknowledgments

I want to thank Sail Labs Technology AG for sponsoring this thesis and providing machines and corpora for experiments. Thanks goes to Andreas Türk for providing his in-depth knowledge of the HTK toolkit. Many thanks to Gerhard Backfried and Norbert Pfanner for supporting the Sail Labs speech recognizer. I would like to thank Jürgen Riedler for sharing his knowledge on the arabic language. Thanks for guidance and constructive criticism to Prof. Friedrich Leisch. Finally I would like to thank Prof. Georg Ch. Pflug for providing me with interesting topic of switching regimes.

Abstract

German

Diese Arbeit gibt eine allgemeine Einführung in den Bereich der automatisierten Spracherkennung mit Hilfe von Hidden Markov Modellen (HMM). Es wurde eine vollständige Trainingsumgebung von Sprachmodellen inklusive Erzeugung von Mix Modellen unter Verwendung des Hidden-Markov-Toolkit (HTK) und eines Spracherkenners von Sail Labs' erstellt. Um die Erkennungsrate zu erhöhen, wurde Maximum Mutual Information (MMI) Parameterschätzung implementiert. Ein 93 Stunden umfassender arabischer Broadcast News Korpus wurde für die Experimente verwendet. Eine Verbesserung der Erkennungsrate durch MMI am verwendeten Korpus konnte nicht festgestellt werden, es wird aber vermutet, dass die nötige Modell Umwandlung um HTK trainierte Modelle in Sail Labs Spracherkenner zu verwenden, dafür verantwortlich ist. An einem einfach Modell, in Form von Switching Regime Modellen, wurden die aus der Spracherkennung bekannten Algorithmen analysiert.

English

This thesis presents a general introduction to automatic speech recognition based on Hidden Markov models (HMM). Using the Hidden-Markov-Toolkit (HTK) and Sail Labs' speech recognizer a complete trainings environment including mixture model training was created. To improve accuracy Maximum Mutual Information (MMI) estimation was implemented. Experiments were carried out using a 93h Arabic broadcast news corpus. MMI estimation could not improve the accuracy on the Arabic corpus, but it is presumed that model transformations needed for usage of HTK trained models in Sail Labs' speech recognizer are responsible. Based on a simple model, namely a switching regime model, algorithms used for speech recognition were analysed.

Keywords: speech recognition, hidden Markov models, maximum mutual information training, discriminative training, switching regimes.

Lebenslauf

Persönliche Daten

Name:	Markus Cozowicz
Adresse:	Bleichergasse 1/5, 1090 Wien
Telefon:	(0664) 283 36 56
E-Mail:	eisber@eisber.net
Geboren:	26.04.1980 in Wien

Wissenschaftlicher Werdegang

Abschluss	Universität	Studium
08/2007	TU Wien	Data Engineering und Statistik Bakkelaureat
08/2006	TU Wien	Software Engineering Master
11/2004	TU Wien	Software Engineering Bakkelaureat

Beruflicher Werdegang

Zeitraum	Unternehmen	Tätigkeit
seit 2000	Sail Technology AG	Forschung & Entwicklung
seit 2004	Research Industrial Software Engineering (RISE)	Entwicklung
2006-2008	Microsoft Österreich	Consultant
2006	Austrian Research Institute for Artificial Intelligence (OFAI)	Forschung & Entwicklung
2003-2007	Fachhochschule des BFI Wien	Lektor
1999-2001	Koen Electronic Media Agency (KEMA)	IT Administration
1999-2001	WIFI Wien	EDV Trainer

Fremdsprachen

Deutsch:	fliessend
Englisch:	fliessend

1 Motivation

The Sail Labs Technology AG created a near real time speech recognizer and trained acoustic models for various languages including English, German, French, Arabic and many more. Business demands constant improvement of the recognizer and models.

Supporting training of models using a public available toolkit and the increasing accuracy is such an important improvement. One of the standard methods to increase accuracy is model estimation using the Maximum Mutual Information (MMI) criteria.

This thesis describes the complete training process of statistical models for Sail Labs' speech recognizer. As the targeted toolkit does not support MMI estimation, it is extended. A detailed description of the implementation of MMI estimation is given. The accuracy of the current Arabic model is not as excellent as the English model and therefore experiments are carried out using the Arabic corpus.

2 Introduction

Automatic speech recognition has been researched for more than 30 years. Today a state of the art toolkit, namely the Hidden-Markov-Toolkit [1] (HTK), is available for the public. Based on this toolkit and the included HTK Book [7] a general introduction to speech recognition is given. To improve accuracy the toolkit is extended to support maximum mutual information estimation.

The speech recognition itself is performed using Sail Labs' [2] speech recognizer and therefore the interaction with HTK and associated tools are described. The automatic speech recognition task is commonly split into three parts [2].

- A **Front-End** transforming the speech signal into feature vectors containing spectral and/or temporal information. Common methods are fast Fourier transform (FFT), linear predictive coding (LPC) and Mel Frequency Cepstral Coefficients (MFCCs).
- An **Acoustic Unit Matching System** matches units of features. Units can be words or sub-words, such as phonemes or syllables. Based on the task (e.g. single digit or continuous speech recognition) the unit size is chosen. Continuous speech recognition typically uses triphones (a phoneme with a left and a right context).

[1] http://htk.eng.cam.ac.uk/
[2] http://www.sail-technology.com

- **Language Model:** Apart from basic acoustic information, knowledge about the grammatical structure of a language is used. The structure can either be obtained by specifying a formal grammar of the language or gathering statistics from big text corpora (e.g. word bi-grams which is the number of occurrences of word W_1 followed by word W_2).

2.1 Human Speech

When humans speak, air is pressed out of the lungs, passing the vocal cords and finally either the mouth or the nose. Different sounds are generated by varying all the previously mentioned parts, called the vocal tract.

The smallest linguistic unit is called a phoneme [6]. Each word can be represented by multiple pronunciations consisting of a sequence of phonemes. The International Phonetic Association (IPA) [1] provides a standardized alphabet. A list of phonemes for the Arabic language used in all experiments is given in Table 1.

Besides the IPA alphabet, the table also lists Sail Labs' (SL) internal phoneme representation derived from Buckwalter encoding and Arabic sample letters. To overcome HTK limitations certain phones are re-mapped and listed in the *HTK* column.

Furthermore the phonemes are categorized into consonants in Tables 2 and vowels in Table 3 categorized by linguistic criteria. v and u denote voiced and unvoiced. The Sail Labs' internal encoding is used for phonemes in Tables 2 and 3.

2.1.1 Place of Articulation

The different phonemes are formed by constriction of the vocal tract. This constriction can happen at different places. A subset used for the Arabic language is

- **bilabial**: The sound is formed by the closure of the lips.
- **labiodental**: The lower lip is pressed against the upper teeth to produce the sound.
- **interdental**: Interdental consonants are produced by placing the blade of the tongue against the upper incisors.
- **dental**: Dentals are articulated with either the lower or the upper teeth, or both, rather than with the gum ridge.
- **dentalveolar**: Dentalveolars are articulated with the flexible front part of the tongue.
- **alveolar**: The alveolar ridge, which is just behind the top front teeth is touched by the tongue.

SL	IPA	HTK	Arabic	SL	IPA	HTK	Arabic
A	/ʔ/, /aː/		ا	d	/d/		د
l	/l/		ل	$	/ʃ/	X	ش
Y	/ʔ/		ئ	2	/ʕ/	B	ع
n	/n/		ن	6	/ð/	C	ذ
s	/s/		س	?	/ʔ/	G	ع
H	/ħ/		ح	q	/q/		ق
b	/b/		ب	D	/ɗ/		ض
x	/χ/		خ	f	/f/		ف
t	/t/		ت	T	/ŧ/		ط
r	/r/		ر	g	/ʁ/		غ
k	/k/		ك	O	/θ/		ث
w	/w/, /uː/		و	S	/s̻/		ص
j	/ʒ/		ج	Z	/ʐ/		ظ
m	/m/		م	J	/ʔ/		أ
z	/z/		ز	BRT			
h	/h/		ه	GRB			
y	/j/, /iː/		ي	sil			
W	/ʔ/		ؤ				

Table 1: Phoneme Alphabet

- **palatal**: When the tongue touches the middle part of the palate.
- **velar**: The back part of the palate is touched by the tongue.
- **uvular**: Uvulars are articulated with the back of the tongue against or near the uvula.
- **pharyngeal**: The root of the tongue is pressed against the pharynx.
- **glottal**: This sound is produced by using the glottis.

A complete list can be found in [25].

2.1.2 Manner of Articulation

Another classification criteria is the manner of articulation, or how the vocal tract constricts. Again, the subset used for the Arabic language is

		bi-labial	labio-dental	inter-dental	dental	dent-alveolar	al-veolar
plosive	v	b			d		D
	u				t		
fricative	v			6,Z		z	
	u		f	O		s	S
nasal		m				n	
trill							r
lateral							l

		palatal	velar	uvular	pharyngeal	glottal
plosive	v					J,W,Y
	u		k	q	2	?
fricative	v	j		g		
	u	$			H	h

		non-emphatic	emphatic
plosive	v	d,W,Y	D,2
	u	t,?	T,q
fricative	v	z,k,j,J	Z,g
	u	s,$,h	S,x,H

Table 2: Sail Labs encoded categorized consonants

- **plosive**: Plosives are produced by stopping the airflow in the vocal tract.
- **fricative**: The vocal tract is nearly closed to produce these phonemes.
- **nasal**: If the air leaves the vocal tract through the nose instead of the mouth.
- **trill**: The sound is produced by vibrations between the articulator and the place of articulation.
- **lateral**: Laterals are pronounced with an occlusion made somewhere along the axis of the tongue, while air from the lungs escapes at one side or both sides of the tongue.

Additional information can be found in [26].

	open	close
front	A	y
back		w

Table 3: Sail Labs encoded categorized vowels

2.2 Signal Transformation

The very rich information of the human speech found in the audio signal is split into various features. The Mel Frequency Cepstral Coefficients (MFCCs) are commonly used for speech recognition ([7]). The following description is based on [4]. To extract the features from the audio signal several steps have to be performed:

1. **Divide into frames**: The signal is divided into frames by applying an overlapping windowing function in fixed intervals (e.g. 10ms). Typically a Hamming window defined as

$$w(n) = \begin{cases} 0.54 - 0.46\cos\left(\frac{2\pi n}{M}\right), & 0 \leq n \leq M \\ 0, & \text{otherwise,} \end{cases}$$

   where $M + 1$ ist the window size and $w(n)$ are the coefficients, is used and removes edge effects at the start and end of the frame. Oppenheim & Schafer [3] give a description of commonly used window functions in signal processing. This process generates a Cepstral feature vector for each frame.

2. **Discrete Fourier Transform** for each frame.

3. **Log of amplitude spectrum**, discards the phase information but retains the amplitude information. According to [5] the amplitude is most important for speech perception.

4. **Mel-scaling and smoothing**: The spectrum is smoothed and more important areas are emphasized by transforming the data to the Mel-scale. Pitch is not perceived in a linear manner and therefore the Mel-scale is based on a mapping between actual frequency and perceived pitch. This scale is linear up to 1kHz and logarithmic above.

5. **Discrete Cosine Transform** (DCT): The components of the Mel-spectral vectors of each frame are highly correlated. To decorrelate the components the DCT [23] is applied.

For all experiments a 45 dimensional Cepstral feature vector is calculated for frames of 10ms. A more detailed description can be found in [2].

2.3 Modeling Phonemes

Based on MFCCs different approaches to model phonemes have been developed. Hidden Markov models are predominantly used for modeling phonemes. Such systems are IBM ([9], [10]), CMU ([12], [13]), Philips [11], BBN/LIMSI ([14], [15]) and HTK [16]. Recent development effort was put into discriminative training of HMMs using as Maximum Mutual Information (MMI) estimation (see chapter 3.4) and Minimum Classification Error (MCE) [19]. Chapter 3 gives a detailed description on HMMs. As an alternative neural networks are used in [17] or combinations of HMMs and neural networks [18].

3 Hidden Markov Model

This section describes the statistical model in use for speech recognition. The hidden Markov model (HMM) is based on the assumption that the speech signal can be well characterized as a parametric random process. The basic theory was published by Baum [8] in the late 1960s and a complete description can be found in [2].

3.1 Markov Model

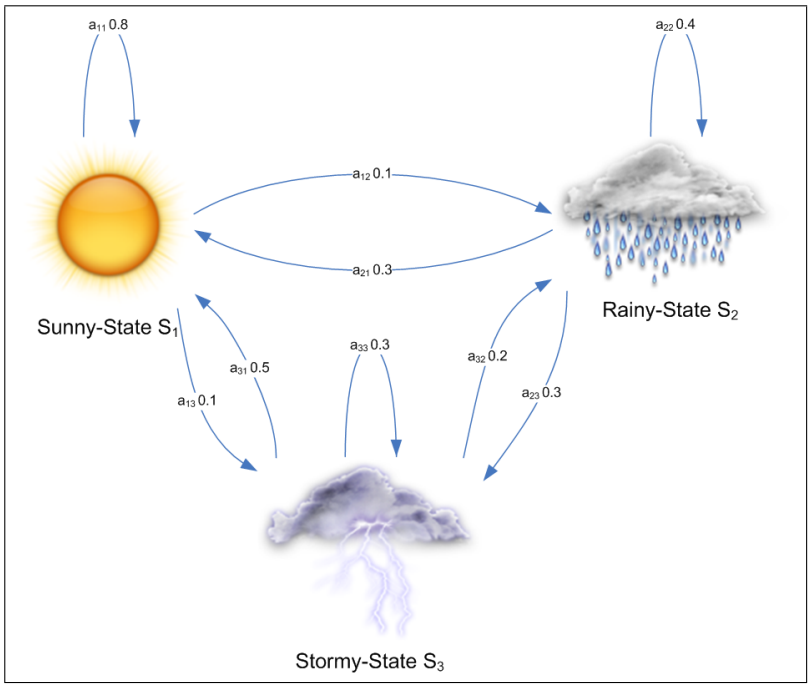

Figure 1: A Markov model of the weather

A Markov model consists of a set of N distinct states. In Figure 1 a weather system with states sunny, rainy and stormy is presented. The system changes its current active state in regular intervals according to probabilities associated to each state. The time instants associated with the state changes are denoted as $t = 1, 2, ...$, the active state is denoted by r_t. The time period for the weather system is a single day. A complete probabilistic description of the system would require the active state and all the preceding states that lead to the current active state. In the case of a first order Markov chain it is assumed that the current state depends only on a single preceding state.

$$P[r_t = j|r_{t-1} = i, r_{t-2} = k, ...] = P[r_t = j|r_t{-}1 = i]$$

Therefore only state-transition probabilities a_{ij} involving two states need to be specified of the following form

$$a_{ij} = P[r_t = j|r_{t-1} = i], \qquad 1 \leq i, j \leq N$$

with these properties

$$\begin{aligned} a_{ij} &\geq 0 \qquad \forall i, j \\ \sum_{j=1}^{N} a_{ij} &= 1 \qquad \forall i \end{aligned}$$

The transition probabilities of the weather example are found on top of the arrows between the states. Finally initial probabilities for the first state defined as

$$\begin{aligned} \pi_i &\geq 0 \qquad 1 \leq i \leq N \\ \sum_{i=1}^{N} \pi_i &= 1 \end{aligned}$$

are required.

The output of an *observable* Markov model are the visited states as time passes by. The states correspond directly to the observed events. A question answered by the model in figure 1 could be:

What is the probability of this week's forecast

Sunny-Sunny-Rainy-Stormy-Rainy

according to the model?

The observation sequence, **O**, is defined as

$$\begin{array}{rlccccc} \mathbf{O} & = (& \mathit{Sunny}, & \mathit{Rainy}, & \mathit{Rainy}, & \mathit{Stormy}, & \mathit{Rainy} &) \\ & = (& S_1, & S_2, & S_2, & S_3, & S_2 &) \\ \mathbf{t} & = (& 1, & 2, & 3, & 4, & 5 &) \end{array}$$

corresponding to this week's forecast. The probability of $P(\mathbf{O}|Model)$ is

$$\begin{aligned} P(\mathbf{O}|Model) &= P(S_1, S_2, S_2, S_3, S_2|Model) \\ &= P[S_1]P[S_2|S_1]P[S_2|S_2]P[S_3|S_2]P[S_2|S_3] \\ &= \pi_{S_1} a_{12} a_{22} a_{23} a_{32} \\ &= (1.0)(0.1)(0.3)(0.3)(0.2) \\ &= 0.0018, \end{aligned}$$

that is, the probability π_{S_1} of being in state S_1 at time 1 multiplied by a_{11} accounting for the transition from state S_1 to state S_2 and so on. π_{S_1} is 1 because $t = 1$ is today.

3.2 Hidden Markov Model

The association between events and states in Markov models are deterministic. Markov models are extended so that the observation, the speech signal, is a probabilistic function of the state. The result is a double stochastic process, where the underlying process, the state transitions, cannot be observed directly, but through another stochastic process that produces the sequence of observations.

A hidden Markov model consists of

1. a distinct number of states **N**;
2. a state-transition probability distribution $A = \{a_{ij}\}$ as specified for Markov models;
3. M observation symbols and a subset $V = \{v_1, v_2, ..., v_M\}$ per state;
4. an observation symbol probability distribution, $B = \{b_j(k)\}$, in which

$$b_j(k) = P[o_t = v_k|r_t = j], \quad 1 \leq k \leq M,$$

defines the symbol distribution in state $j, j = 1, 2, ...N$;

5. an initial state distribution $\Pi = \{\pi_i\}$, in which

$$\pi_i = P[r_1 = i], \quad 1 \leq i \leq N.$$

For convenience the compact notation $\lambda = (A, B, \Pi)$ is used. Applying the above to continuous speech recognition, a single HMM models units of features (e.g. phonemes or syllables).

1. States represent the sequence of audio.

2. The transition probability distribution defines a graph by which the HMM might be passed through.

$$\begin{aligned} a_{ij} &= 0 \quad \forall i > j, \\ a_{ij} &> 0 \quad \forall i \leq j + 1, \\ & \quad\; 1 \leq i, j \leq N \end{aligned}$$

   The graph is normally forward only and provides the ability to skip the next state as presented in Figure 2.

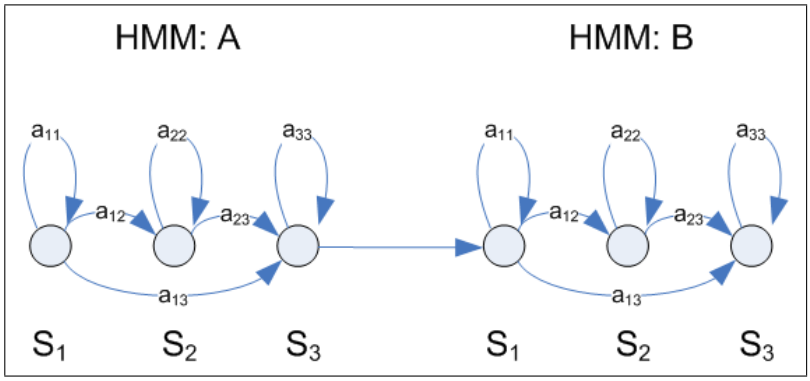

Figure 2: Two HMMs with three states for the phone A and B

   The corresponding transition matrix for phone A is given by

	S_1	S_2	S_3
S_1	a_{11}	a_{12}	a_{13}
S_2	0	a_{22}	a_{23}
S_3	0	0	a_{33}

   During training and recognition consecutive HMMs for phonemes are connected at there start/end states.

3. The observation symbols are signal features as presented in chapter 2.2.

3.3 Three Basic Problems

HMMs introduce three basic problems: evaluating the

1. probability (or likelihood) of a sequence of observations;
2. best sequence of model states;
3. model parameters to fit to the observations.

3.3.1 Probability of an Observation Sequence

The observation sequence **O** and the state sequence **r** are vectors of length T:

$$\mathbf{r} = (r_1, r_2, ...r_T)$$

$$\mathbf{O} = (o_1, o_2, ...o_T)$$

The probability of an observation sequence given the model λ is defined as

$$P(\mathbf{O}|\lambda) = \sum_{Q} P(\mathbf{O}|\mathbf{r}, \lambda)P(\mathbf{r}|\lambda)$$

which is the sum over the joint probability of state sequence **r** given the model λ and the observation sequence **O** given the state sequence **r** and λ for all possible state sequences Q.

The probability of the state sequence **r** is given by

$$P(\mathbf{r}|\lambda) = \pi_{r_1} a_{r_1 r_2} a_{r_2 r_3} ... a_{r_{T-1} r_T}$$

which is the product of the initial probability π_{r_1} of being in state r_1 and the product of transition probabilities $a_{r_i r_j}$ according to the state sequence **r**. The beginning of the path is modeled by π_{r_1} and for each further transition from r_i to r_j, transition probabilities $a_{r_i r_j}$ must be factored in. Given a state sequence **r** the probability of the observation sequence **O** is

$$P(\mathbf{O}|\mathbf{r}, \lambda) = b_{r_1}(o_1) b_{r_2}(o_2) ... b_{r_T}(o_T),$$

the product of observation symbol probabilities for each state r_i. Therefore

$$\begin{aligned} P(\mathbf{O}|\lambda) &= \sum_{Q} P(\mathbf{O}|\mathbf{r}, \lambda)P(\mathbf{r}|\lambda) \\ &= \sum_{r_1, r_2, ..., r_T} \pi_{r_1} b_{r_1}(o_1) a_{r_1 r_2} b_{r_2}(o_2) ... a_{r_{T-1} r_T} b_{r_T}(o_T) \end{aligned}$$

which describes the process as follows: Initially the state r_1 is selected with probability π_{r_1} emitting the symbol o_1 with probability $b_{r_1}(o_1)$. Then the

process continues from state r_1 to r_2 with probability $a_{r_1 r_2}$ and emitting symbol o_2 with probability $b_{r_2}(o_2)$. The process stops at state r_T. This process is computationally infeasible as it requires $2TN^T$ calculations. The Forward Procedure is more efficient.

The Forward Procedure defines the forward variable $\alpha_t(i)$ as

$$\alpha_t(i) = P(o_1, o_2, ..., o_t, r_t = i|\lambda) \tag{1}$$

that is, the probability of the partial observation, $o_1, o_2, ..., o_t$ (until time t) and state i at time t, given the model λ. $\alpha_{1(i)}$ at time 1 for state i is defined as

$$\alpha_1(i) = \pi_i b_i(o_1), \qquad 1 \leq i \leq N.$$

The remaining $\alpha_t(j)$ are recursively defined as:

$$\alpha_{t+1}(j) = \Big[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\Big] b_j(o_{t+1}), \qquad \begin{matrix} 1 \leq t \leq T-1 \\ 1 \leq j \leq N \end{matrix}$$

Instead of evaluating all possible paths, similar to first order Markov models, all previous paths $\alpha_t(i)$ are merged into $\alpha_{t+1}(j)$ weighted by the corresponding transition probabilities a_{ij}. Based on equation 1, one can easily see that

$$\begin{aligned} P(\mathbf{O}|\lambda) &= \sum_{i=1}^{N} P(o_1, o_2, ..., o_T, r_T = i|\lambda) \\ &= \sum_{i=1}^{N} \alpha_T(i). \end{aligned}$$

The computation requires only N^2T compared to $2TN^T$. The key difference between the straight forward definition and the Forward Procedure, is that all possible state sequences at time $t-1$ will merge into $\alpha_t(i)$ for $t > 1$.

The Backward Procedure defines the backward variable $\beta_t(i)$ in a very similar manner to $\alpha_t(i)$:

$$\beta_t(i) = P(o_{t+1}, o_{t+2}, ..., o_T|r_t = i, \lambda)$$

This is the probability of the partial observation sequence from $t+1$ to the end, given state i at time t and the model λ. As with $\alpha_t(i)$, $\beta_t(i)$ is defined inductively as:

$$\begin{aligned} \beta_T(i) &= 1 && 1 \leq i \leq N \\ \beta_t(i) &= \sum_{j=1}^{N} a_{ij} b_j(o_{t+1}) \beta_{t+1}(j) && t = T-1, T-2, ..., 1 \end{aligned}$$

As with the α variable, succeeding paths are merged into $\beta_t(i)$. The following problems can now be solved by using the α and β variable.

3.3.2 Choosing the Optimal State Sequence - Viterbi Algorithm

To find the optimal state sequence one has to define an optimality criterion. Choosing only the best state for each time t might not result in a valid path, as state transition probabilities might be zero. Thus one needs to find the best valid path $\mathbf{r} = (r_1, r_2, ..., r_T)$. This can be done by using the Viterbi algorithm. A quantity for scoring a path is given by

$$\delta_t(i) = \max_{r_1, r_2, ..., r_{t-1}, r_t} P(r_1 r_2 ... r_{t-1}, r_t = i, o_1 o_2 ... o_t | \lambda)$$

Similar to the Forward Procedure, $\delta_{t+1} i$ is defined as

$$\begin{aligned} \delta_1(i) &= \pi_i b_i(o_1) \\ \delta_{t+1}(j) &= [\max_i \delta_t(i) a_{ij}] \cdot b_j(o_{t+1}). \end{aligned}$$

Each $\delta_{t+1}(i)$ is formed by re-using the maximum of $\delta_t(i)$ weighted by the corresponding transition probability a_{ij}. To actually determine the optimal state sequence, one has to keep track of each $\max \delta_t(i)$ and collect $r_{\arg\max[\delta_t(i)]}$ using backtracking.

3.3.3 Estimation of Model Parameters

A far more difficult problem arises when it comes to estimation of model parameters. There is no closed form solution to maximize the probability of the observation sequence given a model, but an iterative procedure commonly known as the Baum-Welch Method or EM (expectation-maximization) method ([27], [28], [29], [30], [31]) exists.

At first, two probabilities need to be defined:

$$\begin{aligned} \gamma_t(i) &= P(r_t = i | \mathbf{O}, \lambda) \\ \xi_t(i, j) &= P(r_t = i, r_{t+1} = j | \mathbf{O}, \lambda) \end{aligned}$$

$\gamma_t(i)$ is the probability of being in state r_i at time t and $\xi_t(i, j)$ is the probability of being in state r_i at time t and in state r_j at time $t + 1$. $\gamma_t(i)$ can be expressed through $\alpha_t(i)$ and $\beta_t(i)$ as produced by the Forward and Backward Procedures:

$$
\begin{aligned}
\gamma_t(i) &= P(r_t = i|\mathbf{O}, \lambda) \\
&= \frac{P(r_t = i, \mathbf{O}|\lambda)}{P(\mathbf{O}|\lambda)} \\
&= \frac{P(r_t = i, \mathbf{O}|\lambda)}{\sum_{i=1}^{N} P(\mathbf{O}, r_t = i|\lambda)} \\
&= \frac{P(o_1, o_2, ..., o_t, r_t = i|\lambda) P(o_t, o_{t+1}, ..., o_T|r_t = i, \lambda)}{\sum_{i=1}^{N} P(\mathbf{O}, r_t = i|\lambda)} \\
&= \frac{\alpha_t(i)\beta_t(i)}{\sum_{i=1}^{N} \alpha_t(i)\beta_t(i)}
\end{aligned}
$$

$\alpha_t(i)$ accounts for the partial observation sequence $o_1, o_2, ..., o_t$ and $\beta_t(i)$ for $o_t, o_{t+1}, ..., o_T$. $\xi_t(i, j)$ can be expressed in terms of α and β

$$
\begin{aligned}
\xi_t(i, j) &= P(r_t = i, r_{t+1=j}|\mathbf{O}, \lambda) \\
&= \frac{P(r_t = i, r_{t+1=j}|\mathbf{O}, \lambda)}{P(\mathbf{O}|\lambda)} \\
&= \frac{P(r_t = i, r_{t+1=j}|\mathbf{O}, \lambda)}{\sum_{i=1}^{N} P(\mathbf{O}, r_t = i|\lambda)} \\
&= \frac{P(o_1, o_2, ..., o_t, r_t = i|\lambda) a_{ij} b_j(o_{t+1}) P(o_{t+1}, o_t, ..., o_T|r_t = j, \lambda)}{\sum_{i=1}^{N} P(\mathbf{O}, r_t = i|\lambda)} \\
&= \frac{\alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}{\sum_{i=1}^{N} \alpha_t(i)\beta_t(i)}
\end{aligned}
$$

Comparing $\xi_t(i, j)$ and $\gamma_t(i)$, one can easily see that a specific path can be modeled by adding the required transition probability a_{ij} and the output probability $b_j(o_{t+1})$ to the product.

The re-estimation of parameters of an HMM are defined using $\gamma_t(i)$ and $\xi_t(i)$:

$$\begin{aligned}
\bar{\pi}_i &= \gamma_1(i) \\
\bar{a}_{ij} &= \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \\
\Gamma(k,t) &= \begin{cases} 1, & \text{if } v_k = o_t \\ 0, & \text{otherwise} \end{cases} \\
\bar{b}_j(k) &= \frac{\sum_{t=1}^{T-1} \Gamma(k,t)\gamma_t(j)}{\sum_{t=1}^{T-1} \gamma_t(j)}
\end{aligned}$$

Updating the model parameters λ as shown above is the maximization step of the EM algorithm. Baum and his colleagues [8] have proven that a re-estimated model $\bar{\lambda} = (\bar{A}, \bar{B}, \bar{\pi})$ defines either a critical point of the likelihood function or $\bar{\lambda}$ is more likely than λ.

3.3.4 Continuous Observation Densities in HMMs

Until now only discrete output symbols have been considered, but in speech recognition one has to deal with a continuous signal. According to [2] reestimation formulas exist if the density used to replace the observation symbol probabilities B, is a finite mixture of log-concave or elliptically symmetric densities. Usually Gaussian mixtures with M components are used to model the signal. Parameter estimation as implemented in HTK [7] starts with a single Gaussian per feature with mean vector μ and covariance matrix $\mathbf{U}$ and is split according to the following procedure: The weight c_{jk} of the j-th state and k-th mixture component is first halved and then the mixture is cloned. The two identical mean vectors are then perturbed by adding 0.2 standard deviations to one and subtracting the same amount from the other. In the second step, the mixture component with the largest weight is split as above. This is repeated until the required number of mixture components are obtained.

$$b_j(o) = \sum_{k=1}^{M} c_{jk} \mathcal{N}(\mathbf{o}, \mu_{jk}, \mathbf{U}_{jk}), \qquad 1 \leq j \leq N, 1 \leq k \leq M$$

As before, o is the observation, c_{jk} is the k-th mixture weight at state j and $\mathcal{N}$ a Gaussian distribution with mean vector μ_{jk} and covariance matrix $\mathbf{U}_{jk}$. To ensure consistent updating of the model, the mixture weights must adhere to stochastic constraints.

$$\begin{aligned}\sum_{k=1}^{M} c_{jk} &= 1, \quad 1 \leq j \leq N \\ c_{jk} &\geq 0, \quad 1 \leq j \leq N, 1 \leq k \leq M\end{aligned}$$

For parameter estimation the EM algorithm is used. The expectation step calculates the α and β variable in the same manner by the Forward/Backward Algorithm for continuous observation densities as it was done for discrete observations. The maximization step estimates a_{ij} as previously presented, but the components of $b_j(k)$ are given by

$$\bar{c}_{jk} = \frac{\sum_{t=1}^{T} \gamma_t(j,k)}{\sum_{t=1}^{T}\sum_{k=1}^{M} \gamma_t(j,k)}$$

$$\bar{\mu}_{jk} = \frac{\sum_{t=1}^{T} \gamma_t(j,k) \cdot \mathbf{o}_t}{\sum_{t=1}^{T} \gamma_t(j,k)}$$

$$\bar{\mathbf{U}}_{jk} = \frac{\sum_{t=1}^{T} \gamma_t(j,k) \cdot (\mathbf{o}_t - \mu_{jk})(\mathbf{o}_t - \mu_{jk})'}{\sum_{t=1}^{T} \gamma_t(j,k)}$$

The mixture weight $\bar{c}_{jk}$ is the ratio of the expected number of times in state j using the k-th mixture and the expected total number of times in state j.

$$\gamma_t(j,k) = \left(\frac{\alpha_t(j)\beta_t(j)}{\sum_{j=1}^{N} \alpha_t(j)\beta_t(j)} \right) \left(\frac{c_{jk}\mathcal{N}(\mathbf{o}, \mu_{jk}, \mathbf{U}_{jk})}{\sum_{m=1}^{M} c_{jm}\mathcal{N}(\mathbf{o}, \mu_{jm}, \mathbf{U}_{jk})} \right)$$

$\gamma_t(j)$ is multiplied by the weighted output probability to accommodate for the mixture components resulting in $\gamma_t(j,k)$, that is the probability of being in state j using the k-th mixture component.

3.4 Maximum Mutual Information Estimation

3.4.1 Introduction

In general, maximum likelihood estimation (MLE) as presented in the previous chapter is used to train HMMs due to the fact that it is an optimal estimate (unbiased with minimum variance).

But MLE is based on the assumption that the observed speech is produced by the HMM. The following wrong assumptions are made:

- Output independence.
- Markov model, a state at time t depends only on states $t-1$.
- Continuous probability density.

In contrast to MLE, maximum mutual information (MMI) estimation relies less on model assumptions and was successfully used by [20], [21] and [22]. The basic idea of mutual information: how much information does one random variable provide about another one? In the case of speech recognition we want to know how much information the observed audio (O) holds about the origin words (S). By maximizing the mutual information between observed audio (O) and origin words (S), the knowledge of S is maximized. The mutual information I between O and S is given by

$$I(O,S) = \sum_{O,S} P(O,S) \log \frac{P(O,S)}{P(O)P(S)}$$

where $P(O,S)$ is the joint probability of the observed audio and the origin words and $P(O)$ and $P(S)$ are the corresponding marginal distributions. It is easy to see that if O and S are independent, that is $P(O,S) = P(O)P(S)$,

$$\log \frac{P(O)P(S)}{P(O)P(S)} = \log 1 = 0 \rightarrow I(O,S) = 0$$

An alternative representation based on entropy (H) is given by

$$\begin{aligned}
I(O,S) &= H(S) - H(O|S) \\
H(O|S) &= -\sum_{o \in O} \sum_{s \in S} P(O=o, S=s) \log P(O=o|S=s) \\
&= -\sum_{o \in O} \sum_{s \in S} P(O=o, S=s) \log \frac{P(O=o, S=s)}{P(S=s)} \\
&= -\sum_{o_1 \in O} \sum_{s \in S} P(O=o_1, S=s) \log \frac{P(O=o_1, S=s)}{\sum_{o_2 \in O} P(O=o_2, S=s)}
\end{aligned}$$

The entropy of the source $H(S)$ cannot be changed, therefore only $H(O|S)$ can be adapted to maximize $I(O,S)$. The MMI criteria is defined as

$$\begin{aligned} F_{\mathrm{MMI}}(\lambda) &= \sum_{r=1}^{R} log P(S_r|O_r,\lambda) \\ &= \sum_{r_1=1}^{R} log \frac{P(O_{r_1}|S_{r_1},\lambda)P(S_{r_1})}{\sum_{r_2=1}^{R} P(O_{r_2}|S_{r_2},\lambda)P(S_{r_2})} \end{aligned} \tag{2}$$

where R is the number of utterances.

$P(S)$ is the probability of the origin word sequence, commonly referred to as the language model score. The denominator term represents the sum of probabilities of all possible word sequences. To overcome the computational infeasibility it is approximated by the N-Best hypothesis or a word lattice produced by a recognition run.

3.4.2 Estimation of Model Parameters

To define the estimation formulas we need to specify two quantities

$$\begin{aligned} \gamma_{jm} &= \sum_{t=1}^{T} \gamma_t(j,m) \\ \theta_{jm} &= \sum_{t=1}^{T} \gamma_t(j,m) \cdot \mathbf{o}_t. \end{aligned}$$

γ_{jm} is the occupation count for state j using the k-th mixture component, θ_{jm} is the weighted sum of output symbols. Two different θ and γ are used for reestimation of μ_{jm} and σ_{jm}: $\theta_{jm}^{\mathrm{num}}$ correspond to the numerator term of (2) and θ^{den} to the denominator term. The numerator part is based on a single transcription using MLE, the denominator part uses a lattice including confusability of the speech recognizer. The new $\bar{\mu}_{jm}$ and $\bar{\sigma}_{jm}^2$ are given by

$$\begin{aligned} \bar{\mu}_{jm} &= \frac{\{\theta_{jm}^{\mathrm{num}}(\mathbf{O}) - \theta_{jm}^{\mathrm{den}}(\mathbf{O})\} + D\mu_{jm}}{\{\gamma_{jm}^{\mathrm{num}} - \gamma_{jm}^{\mathrm{den}}\} + D} \\ \bar{\sigma}_{jm}^2 &= \frac{\{\theta_{jm}^{\mathrm{num}}(\mathbf{O}^2) - \theta_{jm}^{\mathrm{den}}(\mathbf{O}^2)\} + D(\sigma_{jm}^2 + \mu_{jm}^2)}{\{\gamma_{jm}^{\mathrm{num}} - \gamma_{jm}^{\mathrm{den}}\} + D} - \hat{\mu}_{jm}^2 \end{aligned}$$

where D is set on a per Gaussian level, that is for every state j and every mixture component m. D is set so that all variances σ^2 are positive. A few useful definitions to shorten the transformation.

$$\begin{aligned}\Theta_\sigma &= \theta_{jm}^{\text{num}}(\mathbf{O}^2) - \theta_{jm}^{\text{den}}(\mathbf{O}^2)\\ \Theta_\mu &= \theta_{jm}^{\text{num}}(\mathbf{O}) - \theta_{jm}^{\text{den}}(\mathbf{O})\\ \Gamma &= \gamma_{jm}^{\text{num}} - \gamma_{jm}^{\text{den}}\end{aligned}$$

D is set twice the minimum given by

$$\begin{aligned}0 &= \frac{\Theta_\sigma + D(\sigma_{jm}^2 + \mu_{jm}^2)}{\Gamma + D} - \hat{\mu}_{jm}^2\\ &= \frac{\Theta_\sigma + D(\sigma_{jm}^2 + \mu_{jm}^2)}{\Gamma + D} - \Big(\frac{\Theta_\mu + D\mu_{jm}}{\Gamma + D}\Big)^2\\ &= \frac{(\Theta_\sigma + D(\sigma_{jm}^2 + \mu_{jm}^2))(\Gamma + D) - (\Theta_\mu + D\mu_{jm})^2}{(\Gamma + D)^2}\\ &= (\Theta_\sigma + D(\sigma_{jm}^2 + \mu_{jm}^2))(\Gamma + D) - (\Theta_\mu + D\mu_{jm})^2\\ &= D^2(\sigma_{jm}^2) + D(\Gamma(\sigma_{jm}^2 + \mu_{jm}^2) + \Theta_\sigma - 2\Theta_\mu\mu_{jm}) + \Theta_\sigma\Gamma - \Theta_\mu^2\end{aligned}$$

A full derivation is found in [24].

4 MMIE Implementation

4.1 Introduction

The HTK was used to train the large vocabulary continuous speech recognition system.

At the time of writing only HTK 3.2.1 is available, which does not include MMIE. As the HTK already provides a comprehensive framework for HMM training, it is chosen as a base for MMIE implementation. This chapter focuses on the implementation of MMIE, based on the MLE implementation found in the HTK tool *HERest*. *HERest* is built from a multitude of sources files. The following list contains the source files modified or extended for MMIE:

- *HTKTools/HERest.c*: Main entry point of *HERest*. It includes loading of data files and model update.
- *HTKLib/HFB.c*: Accumulation of α and β variable of the forward/backward algorithm for transcripts.
- *HTKLib/HNet.h*: Defines data structures holding lattices (see Figure 3 and 4).

- *HTKLib/HMMI.c*: Accumulation of α and β variable of the generalized forward/backward algorithm for lattices.

The original parameter estimation and statistic accumulation of the α and β variable according to the forward/backward algorithm is done using *HERest*. It can process transcripts and calculate statistics needed for MLE. MMIE needs statistics accumulated from lattices and therefore *HERest* was generalized to support these. A comparison of a transcript and an associated lattice generated by a recognition run is found in Figure 3.

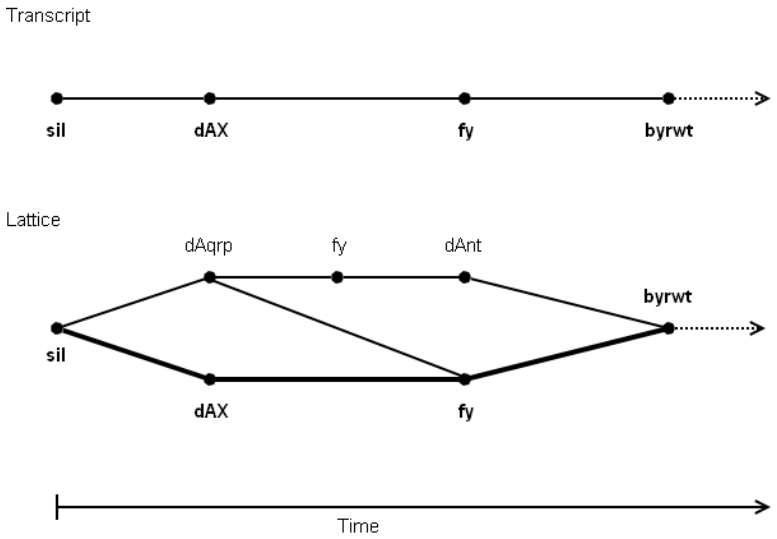

Figure 3: Transcript (MLE) vs. Lattice (MMIE)

The implementation of statistic accumulation for transcripts can be found in *HTKLib/HFB.c* and the actual parameter update is done in *HTKTools/HERest.c*. To fit into the current structure, *HTKLib/HMMI.c* contains statistic accumulation for lattices and *HTKTools/HERest.c* was extended with MMI based parameter re-estimation.

MLE statistic accumulation uses label files containing transcripts for the audio. Depending on the matching unit modeled by an HMM the transcripts are either word, monophone or triphone based. MMIE accumulation needs lattices from a recognition fitting with the matching units. Sail Labs' recognizer produces word lattices in HTKs standard lattice format (slf).

The generated lattices can be read and held in memory with functions and data structures already provided by HTK. To accumulate statistics the right transcript level is needed and the lattice is expanded to match the HMMs by HTKs *ExpandWordNet* function. The function requires a dictionary containing each word with multiple associated pronunciations. To expand each word into its pronunciation, the lattice needs to contain a pro-

nunciation ID per word. The returned *Network* structure found in *HTK-Lib/HNet.h* serves as the root for the expanded lattice.

The nodes in the expanded lattice are linked in a chain for fast unordered processing and each node contains an array of *NetLink* structures which link to the succeeding nodes. For MMIE additional information for each node during the forward/backward algorithm is needed, namely

- a list of succeeding and preceding nodes including language model scores for the transition,
- a flag if the model is active at time t,
- time based pruning information,
- a model ID q to fit the MLE implementation.

The *NetNode* structure representing each HMM was extended with *MMIInfo* structure holding the above information. Figure 4 shows the graph of the data structures in use for the first words of the already expanded lattice from Figure 3.

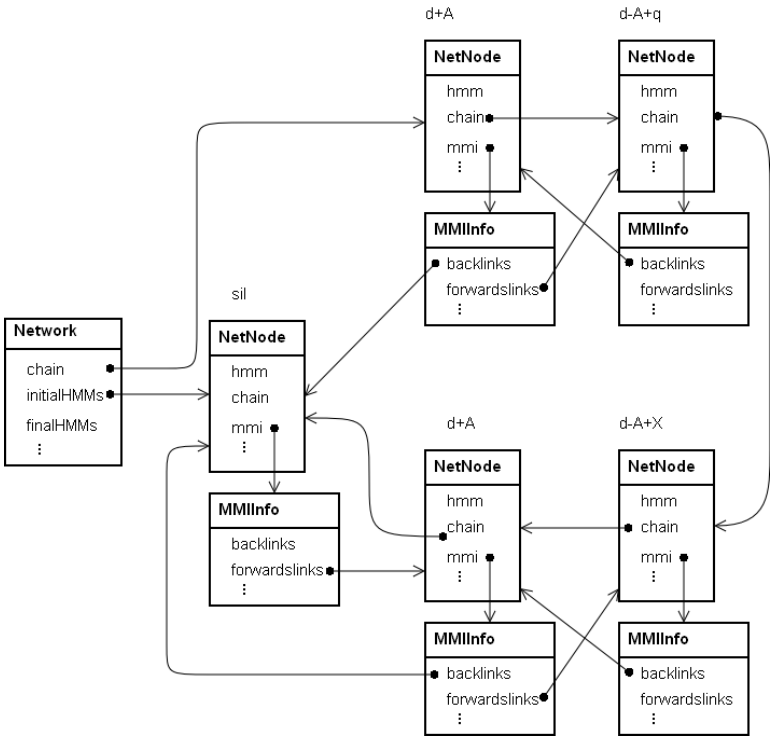

Figure 4: Data structures used for lattices

The *forwardlinks* and *backlinks* are single linked lists generated by recursively walking through the lattice. For now only a single initial and final HMM is allowed due to initialization restrictions.

The word lattice contains language model scores between each word. This score is used as a transition probability in the forward/backward algorithm. The language model score $P(W_2, W_1)$ for the word sequence W_1, W_2 is spread equally among the expanded monophone or triphone nodes.

4.2 Backward Procedure

HERest calculates first the β variable and in a following step the α variable. The MLE implementation assigns Q identifiers ordered by time to each HMM. This numbering is used to process the HMMs in the correct order and to prune the number of active HMMs at each time t. For more details on pruning see chapter 4.5.

Each HMM has an identifier q and N states with an associated transition probability matrix a. The first and last state are always non-emitting states. Due to the time-based numbering, successors and predecessors are referenced by $q+1$ and $q-1$ respectively. For the case of lattices, multiple successors and predecessors may exist and therefore each HMM has an associated set F and B implemented by the single linked lists *forwardlinks* and *backlinks* found in *MMIInfo*. Each link holds the previously spread language model l. The backward step starts at time $t = T$ and progresses toward $t = 1$. Before β is evaluated, the output probabilities b for each active HMM and state are computed. β for the last state of each HMM active at time $t = T$ is calculated by

$$\begin{aligned} \beta_{T,q,N} &= 1 && \text{if } q = Q \\ \beta_{T,q,N} &= \beta_{T,q+1,N} \cdot a_{q+1,1,N} && \text{if } q \neq Q \end{aligned}$$

The generalized version is given by

$$\begin{aligned} \beta_{T,q,N} &= 1 && \text{if } |F| = 0 \\ \beta_{T,q,N} &= \sum_{f \in F_q} \beta_{T,f,N} \cdot a_{f,1,N} && \text{else} \end{aligned}$$

It should be noted that $a_{1,N} > 0$ is only true for the small pause model (*sp*). Furthermore, the summation over F needs to be done in order, that is if f is part of the active set, $\beta_{T,f,N}$ needs to evaluated prior to q.

β of the emitting states for MLE and MMIE is given by

$$\beta_{T,q,i} = a_{i,N} \cdot \beta_{T,q,N} \qquad 2 \leq i < N$$

And finally $\beta_{T,q,1}$ is

$$\beta_{T,q,1} = a_{1,j} \cdot b_j \cdot \beta_{T,q,j} \qquad 2 \leq j < N$$

For the remaining time t, the β variable is given by

$$\beta_{t,q,N} \quad = \quad \beta_{t+1,q+1,1} + \beta_{t,q+1,N} \cdot a_{q+1,1,N}$$

Again $a_{q+1,1,N}$ uses the property that the q's are ordered by time and therefore the generalized version needs to adapted.

$$\beta_{t,q,N} = \sum_{f \in F_q} \beta_{t+1,f,1} \cdot l_{q,f} + \beta_{t,f,N} \cdot a_{f,1,N}$$

The calculation needs to be done in the correct order as before. Beside the adaptation for multiple successors, the sum includes the language model score $l_{q,f}$.

β of states 2 to N are given by

$$\beta_{t,q,i} \quad = \quad a_{q,i,N} \cdot \beta_{t,q,N} + \sum_{j=2}^{N-1} a_{q,i,j} \cdot \beta_{t+1,q,j} \quad i = N-1 \rightarrow 2$$

$$\beta_{t,q,1} \quad = \quad \sum_{j=2}^{N-1} a_{q,1,j} \cdot b_{q,j} \cdot \beta_{t,q,j}$$

4.3 Forward Procedure

The forward procedure calculates the α-variable and accumulates statistics for parameter estimation, iterating from time $t = 1$ to T. The accumulated statistics are an expression of α_t, α_{t-1}, β_t and β_{t-1}.

Due to pruning, only a subset of βs are calculated during the backward procedure, thus only αs for this subset need to be calculated. Each step $t = t + 1$, for $t > 1$ performs:

1. Calculate α_t
2. Accumulate statistics
3. Swap α_t and α_{t-1} to save memory

For MLE, α at time t, HMM q, state 1 is given by

$$\alpha_{t,q,1} = \alpha_{t-1,q-1,N} + \alpha_{t,q-1,1} \cdot a_{q-1,1,N}$$

where N is the number of states of HMM $q-1$. The HMM $q-1$ is before HMM q in the transcription. As lattices may contain multiple predecessors,

this and all following expressions containing references to $q-1$ must be generalized.

For MMI, $\alpha_{t,q,1}$ is given by

$$\alpha_{t,q,1} = \sum_{b \in B_q} \alpha_{t-1,b,N} \cdot l_{q,b} + \alpha_{t,b,1} \cdot a_{b,1,N}$$

$l_{q,f}$ is the spread language model score. α for the remaining states 2 to N is given by

$$\begin{aligned} \alpha_{t,q,j} &= a_{q,1,j} \cdot \alpha_{t,q,j} + \left(\sum_{i=2}^{N-1} a_{q,i,j} \cdot \alpha_{t-1,q,i} \right) \cdot b_{q,j} \quad j = 2 \rightarrow N-1 \\ \alpha_{t,q,N} &= \sum_{i=2}^{N-1} a_{q,i,N} \cdot \alpha_{t,q,i} \end{aligned}$$

4.4 Parameter Estimation

For each HMM, state and mixture, the following variables are accumulated during the forward procedure

$$x_{t,j} = \frac{\left(a_{1,j} \cdot \alpha_{t,1} + \sum_{i=2}^{N-1} a_{i,j} \cdot \alpha_{t-1,i} \right) \cdot \beta_{t,j}}{P(\mathbf{O}|\lambda)} \cdot c_{j,m} \cdot b_{j,m} \quad 1 < j < N$$

where $P(\mathbf{O}|\lambda)$ is estimated with $\beta_{1,1,1}$, that is β at time 1, for the first HMM of the transcript and state 1. Due to this estimation, a lattice must not contain multiple HMMs at the beginning. $c_{q,j,m}$ is the m-mixture weight at state j.

$$\begin{aligned} \gamma &= \sum_{t=1}^{T} x_t \\ \theta(\mathbf{O})_k &= \sum_{t=1}^{T} (\mathbf{O} - \mu_k) \cdot x_t \\ \theta(\mathbf{O}^2)_k &= \sum_{t=1}^{T} (\mathbf{O} - \mu_k)^2 \cdot x_t \end{aligned}$$

μ and σ of each mixture component k are updated by

$$\hat{\mu}_k = \mu_k + \frac{\theta(\mathbf{O})_k}{\gamma}$$

$$\hat{\sigma}_k = \frac{\theta(\mathbf{O}^2)_k}{\gamma}$$

For MMIE the accumulated γ, μ and σ based on lattices are stored separately. MMI estimation of μ and σ uses transcript and lattice accumulations, where γ^{num}, μ^{num}, σ^{num} refer to the transcript and γ^{den}, μ^{den}, σ^{den} to the lattice. The estimation is given by

$$\hat{\mu}_k = \frac{(\theta(\mathbf{O})_k^{\text{num}} + (\gamma^{\text{num}} \cdot \mathbf{O}) - \theta(\mathbf{O})_k^{\text{den}} + (\gamma^{\text{den}} \cdot \mathbf{O}) + D \cdot \mu_k)}{\gamma^{\text{num}} - \gamma^{\text{den}} + D}$$

$$\hat{\sigma}_k = \frac{(\theta(\mathbf{O}^2)_k^{\text{num}} - \theta(\mathbf{O}^2)_k^{\text{den}}) + (D \cdot \sigma_k + \mu_k^2)}{\gamma^{\text{num}} - \gamma^{\text{den}} + D} - \hat{\mu}^2$$

where D is found in chapter 3.4.2.

4.5 Pruning

Pruning is a vital part of the trainings process. Consider the sentence

"The weather is very warm".

At time $t = 1$ only the word "the" needs to be looked at and not all the other words occurring in the sentence. As the time advances during the forward/backward algorithm only words, respectively the associated HMMs, in a certain time window should be considered. This strategy has the advantages of severely reducing the required amount of computation and assures proper alignment of the transcript and the audio data. Suppose all words are considered at every time and the current position in the audio is at the beginning of the word "weather". Both words "weather" and "warm" have some phonetic similarity at the beginning and therefore the α and β variable would be increased, although the audio data actually corresponds to the word "weather". To avoid this situation the following two pruning techniques are used:

- Threshold based
- Time based

Threshold based A HMM q is removed from the current active set as soon as

$$max_q(\beta_{q,t,i}) - \beta_{q,t,i} < \epsilon$$

where i is the state number and ϵ is a pruning threshold. In some cases the forward/backward algorithm fails to process an utterance due to a very tight threshold. Therefore *HERest* provides three pruning parameters: start threshold, step size and maximum threshold. The start threshold specifies the initial threshold. If the forward/backward algorithm fails, the threshold is increased by the step size and the utterance is processed again. The threshold is increased until the maximum threshold is reached or the utterance was processed successfully. Unlike the transcript implementation, the lattice version needs to prune each HMM separately. The transcript version defines a pruning beam specifying boundaries based on HMM IDs, which correlate directly to their chronological occurrence in the transcript. As a lattice holds several possible transcripts, multiple HMMs occur at the same time. Therefore no chronologically ordered IDs for each HMM can be given and pruning needs to be applied to each HMM part of the active set. The main purpose of threshold based pruning is speed and memory improvement.

Time based Each HMM has an associated minimum processing duration. Based on this minimum duration, the transcript version creates two vectors for the previously mentioned pruning beam boundaries, q_{lo} and q_{hi}. As soon as the sum of minimum durations of all predecessors of an HMM is larger than time t, the HMM can be included in the active set/pruning beam of the forward procedure. Likewise for the backward procedure, the sum of minimum durations of all successors of an HMM needs to be smaller than $T - t$, where T is the time span of the current utterance. This models the idea, that an HMM should not be used as long as all its successors or predecessors respectively, have not been processed. For the generalized lattice case, the earliest usage of an HMM is defined as the sum of minimum durations of the all HMMs on shortest path from the first or last HMM to the one in question. The time based pruning needs to be equal for forward and backward, especially in connection with *sp*. If the timing is off-by-one the forward and backward procedure could produce unequal $\alpha_{t=T}$ and $\beta_{t=1}$.

4.6 Descriptive Statistics

To validate and compare the MMIE implementation to the existing MLE, intermediate values have been visualized. The HMMs and there states are ordered by time form the y-axis. Each data point is either the value of α, β or $\alpha \cdot \beta$ for the combination of HMM, state and time.

In Figure 5 the α and β according to the forward/backward procedure for a sample utterance are presented. One can see that the value of α

and β decreases with the direction of the recursive definition, that is with increasing respectively decreasing time. White areas mark combinations of HMM, state and time where no value is available due to pruning. As one can see the pruning beam is very tight and thus only a few HMMs are active at a time.

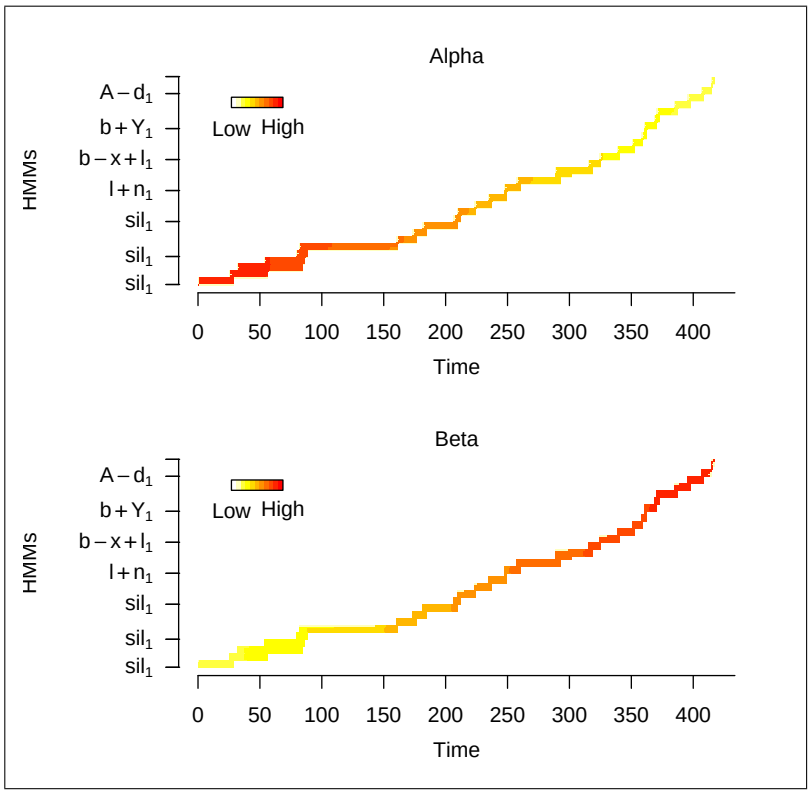

Figure 5: α/β variable for an utterance

Figure 6 presents α and β of a lattice matching the utterance of figure 5. Comparing the lattice with the utterance, one can see that the pruning beam is much wider as similar words are considered at the same time. In difference to the backward procedure calculating the β variable, the forward procedure is able to perform better alignment to the correct utterance, resulting in high values of the α variable for only a few HMMs at a time. The HMMs on the y-axis are no longer strictly ordered by time due to the internal storage structure.

In Figure 7 a detail view of an HMM and its states is presented. Each phoneme consists of a series of sounds. The fact that this series is ordered, is modeled by the transition probability matrix that defines a graph that can only be traversed forward. Thus each state models a part of the series. In Figure 7 each state has a series of darker and wider dots accounting for the amount of time it models the audio.

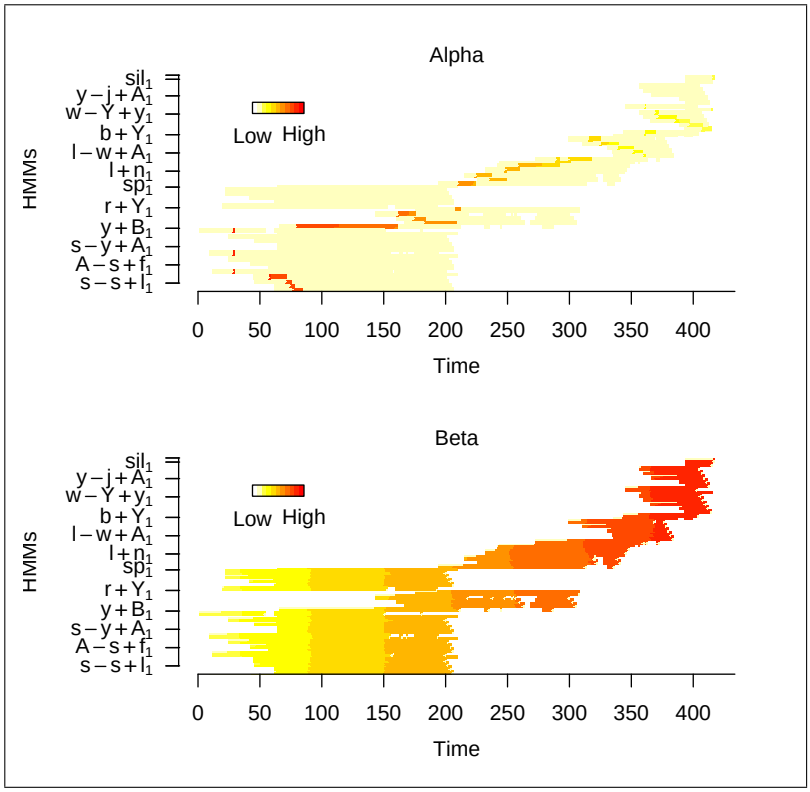

Figure 6: α/β variable for a lattice

Figure 8 presents $\alpha \cdot \beta$ of a lattice. As one can see the number of active HMMs at a time is much larger compared with the corresponding utterance. The HMMs are approximately ordered by time, but due to the structure of a lattice the path is spread across the plot.

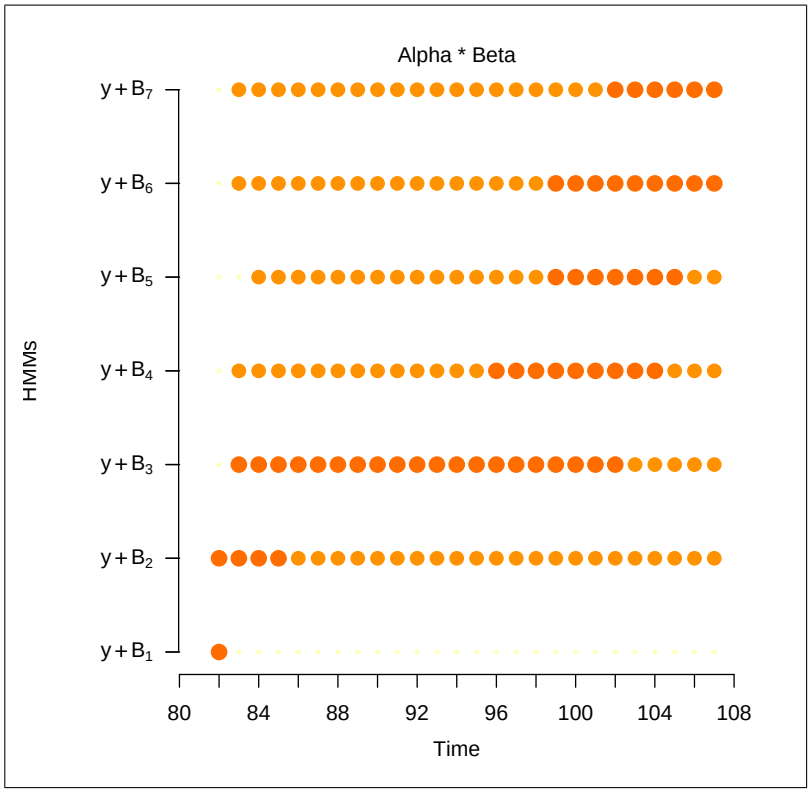

Figure 7: $\alpha \cdot \beta$ variable for an utterance

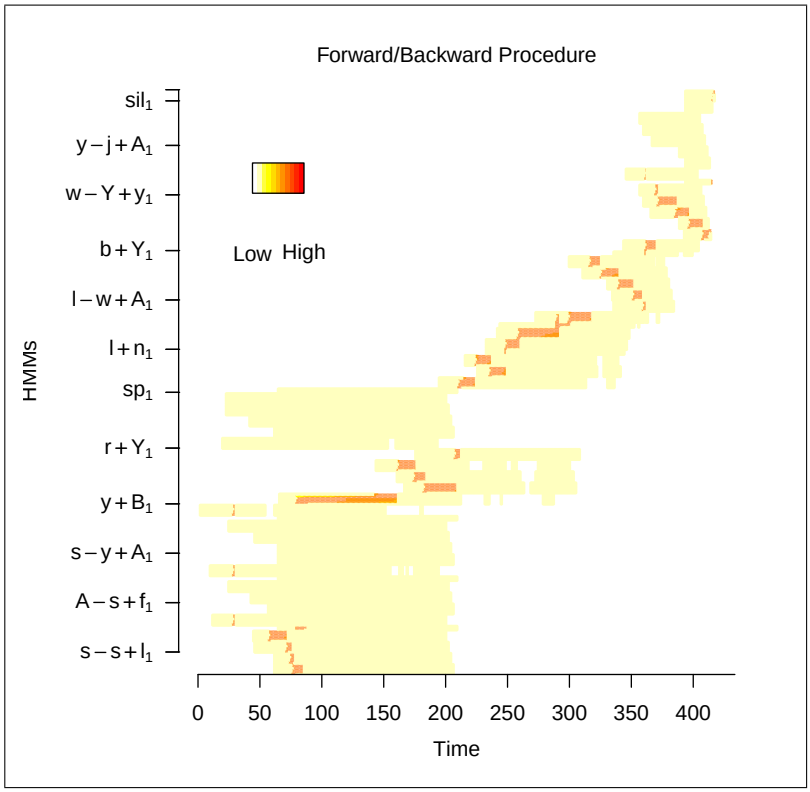

Figure 8: $\alpha \cdot \beta$ variable for a lattice

5 Switching regime estimation

Hidden Markov models are not only used in speech recognition as presented in the previous chapters, but also for switching regime modelling. The number of parameters required for a switching regime is alot smaller than for speech recognition, still the same algorithms can be utilized. The following chapters present the statistical model, estimation of parameters using Baum-Welch algorithm and analysis of state sequence estimation performance using simulation for various parameter combinations.

5.1 Model

Let $X_1, \ldots, X_T$ be an unobservable sequence of zeros and ones following the Markov transition

$$P = \begin{pmatrix} 1-p & p \\ q & 1-q \end{pmatrix}.$$

Let μ_0, μ_1 resp. σ_0, σ_1 be means and standard deviation parameters. The observable sequence $Y_1, \ldots, Y_T$ follows the distribution

$$Y_t \sim N(\mu_{X_t}, \sigma^2_{X_t}).$$

The problem is to estimate the following parameters:

$$p, q, \mu_0, \mu_1, \sigma_0, \sigma_1, X_T.$$

5.2 Estimation

As previously presented in chapter 3.3.3 the Baum-Welch algorithm can be used to estimate Hidden Markov model parameters. For the above model, parameters are estimated by

$$\begin{aligned} \gamma_t(i) &= \frac{\alpha_t(i)\beta_t(i)}{\sum_{i=1}^{N} \alpha_t(i)\beta_t(i)} \\ \xi_t(i,j) &= \frac{\alpha_t(i)a_{ij}b_j(y_{t+1})\beta_{t+1}(j)}{\sum_{i=1}^{N} \alpha_t(i)\beta_t(i)} \\ \bar{\pi}_i &= \gamma_1(i) \end{aligned}$$

$$
\begin{aligned}
\bar{p} &= \frac{\sum_{t=1}^{T-1} \xi_t(0,1)}{\sum_{t=1}^{T-1} \gamma_t(0)} \\
\bar{q} &= \frac{\sum_{t=1}^{T-1} \xi_t(1,0)}{\sum_{t=1}^{T-1} \gamma_t(1)} \\
\bar{\mu}_i &= \frac{\sum_{t=1}^{T} \gamma_t(i) y_t}{\sum_{t=1}^{T} \gamma_t(i)} \\
\bar{\sigma}_i &= \frac{\sum_{t=1}^{T} \gamma_t(i)(y_t - \mu_i)^2}{\sum_{t=1}^{T} \gamma_t(i)}
\end{aligned}
$$

where i, j denotes the state number.

As the algorithm requires starting values the following procedure was implemented:

1. Estimation of μ_i, σ_i using k-means algorithm.
2. Estimation of p, q based on state assignment τ_t of k-means, where τ_t denotes the assigned state and $1 \leq t \leq T$.

$$
\begin{aligned}
m_t(i,j) &= 1_{\tau_{t-1}=i \wedge \tau_t = j} \quad 2 \leq t \leq T \\
\bar{p} &= \frac{\sum_{t=2}^{T} m_t(0,1)}{\sum_{t=2}^{T} m_t(0,0) + m_t(0,1)} \\
\bar{q} &= \frac{\sum_{t=2}^{T} m_t(1,0)}{\sum_{t=2}^{T} m_t(1,1) + m_t(1,0)}
\end{aligned}
$$

where i, j denotes the assigned state.

5.3 Simulation

Initial experiments were performed using the R [32] package HiddenMarkov. Due to the extended run time the complete simulation was re-implemented and optimized in C++ using Intel Performance Primitives. The results were validated against the R package HiddenMarkov. To overcome numerical underflows the forward and backward variable of the Baum-Welch algorithm are stored in logarithmic representation. Furthermore $\gamma_t(i)$ is calculated by

$$\gamma_t(i) = \exp(\log \alpha_t(i) + \log \beta_t(i) - f - \log(\sum_{i=1}^{N} \exp(\log \alpha_t(i) + \log \beta_t(i) - f)))$$

with

$$f = \log \alpha_{\frac{T}{2}}(0) + \log \beta_{\frac{T}{2}}(0).$$

For a given parameter combination each experiment consists of the following steps:

1. Random generation of X_t, Y_t based on given parameters.
2. Clustering of Y_t using k-means for initial parameter estimation.
3. Baum-Welch parameter re-estimation with a maximum of 100 iterations or until the difference between two subsequent iterations in log-likelihood is smaller than 0.00001.
4. Prediction of X_t by the Viterbi algorithm (see Chapter 3.3.2).

5. Calculation of state prediction success rate and $(\bar{\mu}_i - \mu_i)^2, (\bar{\sigma}_i - \sigma_i)^2$. Since switched state labels might occur in the described procedure, the maximum of switched and non-switched state prediction success rates is reported.

5.4 Simulation Results

5.4.1 A single observation sequence

A single run based on 75 observations, with state switching probability of 0.2 for each state and $Y_t|X_t = 0 \sim N(0,1), Y_t|X_t = 1 \sim N(1,1)$, is visualized in Figure 11. Each box represents an observation generated by this process. Correctly predicated states are marked as green boxes and mispredicted states as red boxes. Furthermore the real and estimated state sequence is plotted below the graph. To the right, real and estimated densities per state are displayed. Connecting lines between each observation are only visual aids to perceive the temporal process and do not represent any intermediate values. Evidently longer phases can be predicted to a large extend, however observations between two state centers are rather easily mispredicted.

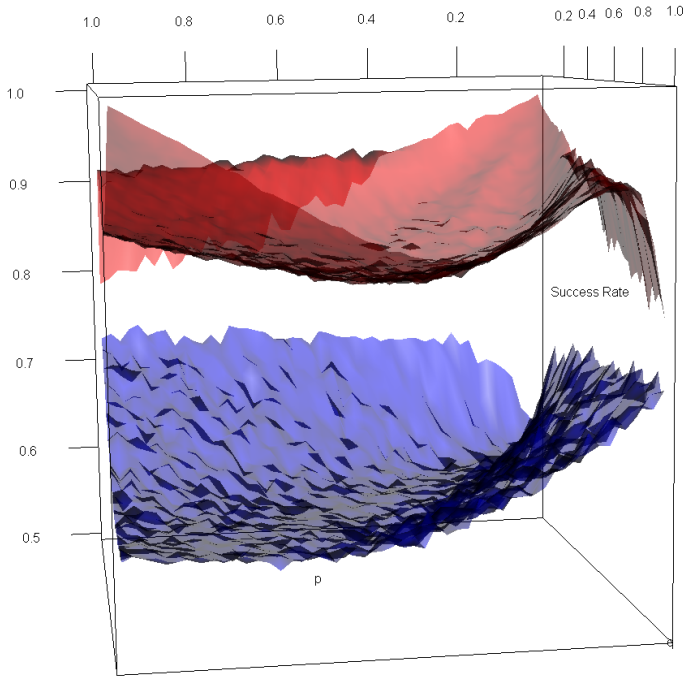

Figure 9: Success rate as a function of state switching probabilities for $\mu_1 - \mu_0 = 0.5$ *(blue)* and $\mu_1 - \mu_0 = 2$ *(red)*

5.4.2 Effect of state switching probabilities on state prediction

To investigate the effect of state switching probabilities on the state prediction success rate, experiments inside the following parameter space were performed:

Experiment	μ_0	σ_0	μ_1	σ_1	Figure	Layer Color
1	0	1	0.5	1	9	blue
2	0	1	2	1	9	red
3	0	0.5	1	1	12	green

$$p, q \in \left\{ \frac{1}{50}, \frac{2}{50}, \ldots, \frac{49}{50} \right\}$$

The average success rate of 100 repeated experiments per parameter combination is reported. Selected experiments were repeated 200 times and yielded the same results. When comparing experiment number one and two in Figure 9, it becomes evident that the delta between μ_0 and μ_1 has a major effect on the state prediction success rate. Experiment one shows an increase in the success rate, when exclusively one of the switching probabilities is small ($p, q < 0.15$). The smaller the switching probability, the more observations origin from this state and prediction improves. In Figure 10 the mean squared error (MSE) of μ_0 and μ_1 for $\mu_1 - \mu_0 = 0.5$ is shown together with the success rate. MSE decreases for the more frequent state.

In experiment number two an opposite effect of small p, q can be observed. Underrepresentation of one state in the observation sequence makes parameter estimation more difficult. For p and $q > 0.8$ the process switches very often and both states can be observed well. In combination with a larger delta ($\mu_1 - \mu_0 = 2$) the success rate increases.

An asymmetric effect is achieved by unequal standard deviation as shown in Figure 12 by experiment three. A smaller σ_0 of 0.5 makes prediction of state 0 easier and values of p, q favoring state zero increases the success rate.

5.4.3 Effect of ergodic coefficient and difference $\mu_1 - \mu_0$

In the previous experiments the most dominant effect was the difference of $\mu_1 - \mu_0$. Furthermore for equal standard deviation the state prediction success rate was symmetric in p and q. Therefore the switching probabilities are expressed through the ergodic coefficient e.

The stationary distribution of the Markov transition is given by

$$\left[\frac{q}{p+q}\ \frac{p}{p+q}\right]$$

as long as $p, q > 0$. Thus the ergodic coefficient is

$$e = \frac{\frac{q}{p+q}}{\frac{p}{p+q}} = \frac{q}{p}.$$

To perform the experiment p must be given and q can be expressed by the ergodic coefficient. In Figure 13 the state prediction success rate is plotted as a function of the ergodic coefficient and $\delta = \mu_1 - \mu_0$ for $p = 0.5$. Clearly δ is most relevant for the prediction of the underlying state sequence X_t. The experiment was carried out using different values for p and lead to the same conclusion.

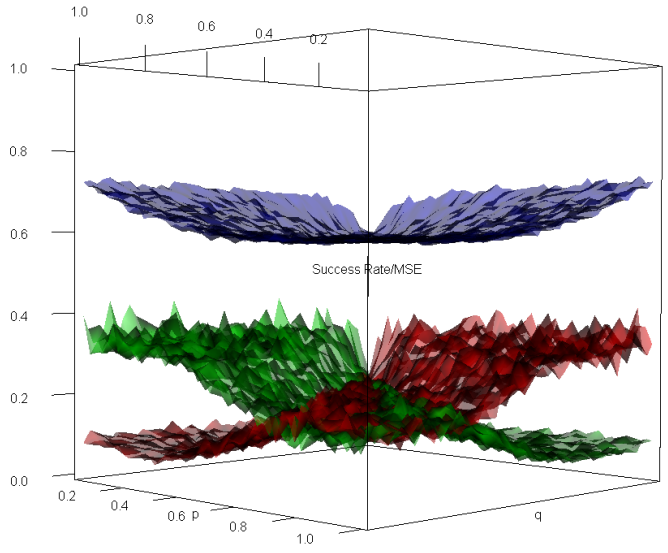

Figure 10: Success rate as a function of state switching probabilities for $\mu_1 - \mu_0 = 0.5$ *(blue)* and mean squared error of μ_0 *(red)* and μ_1 *(green)*

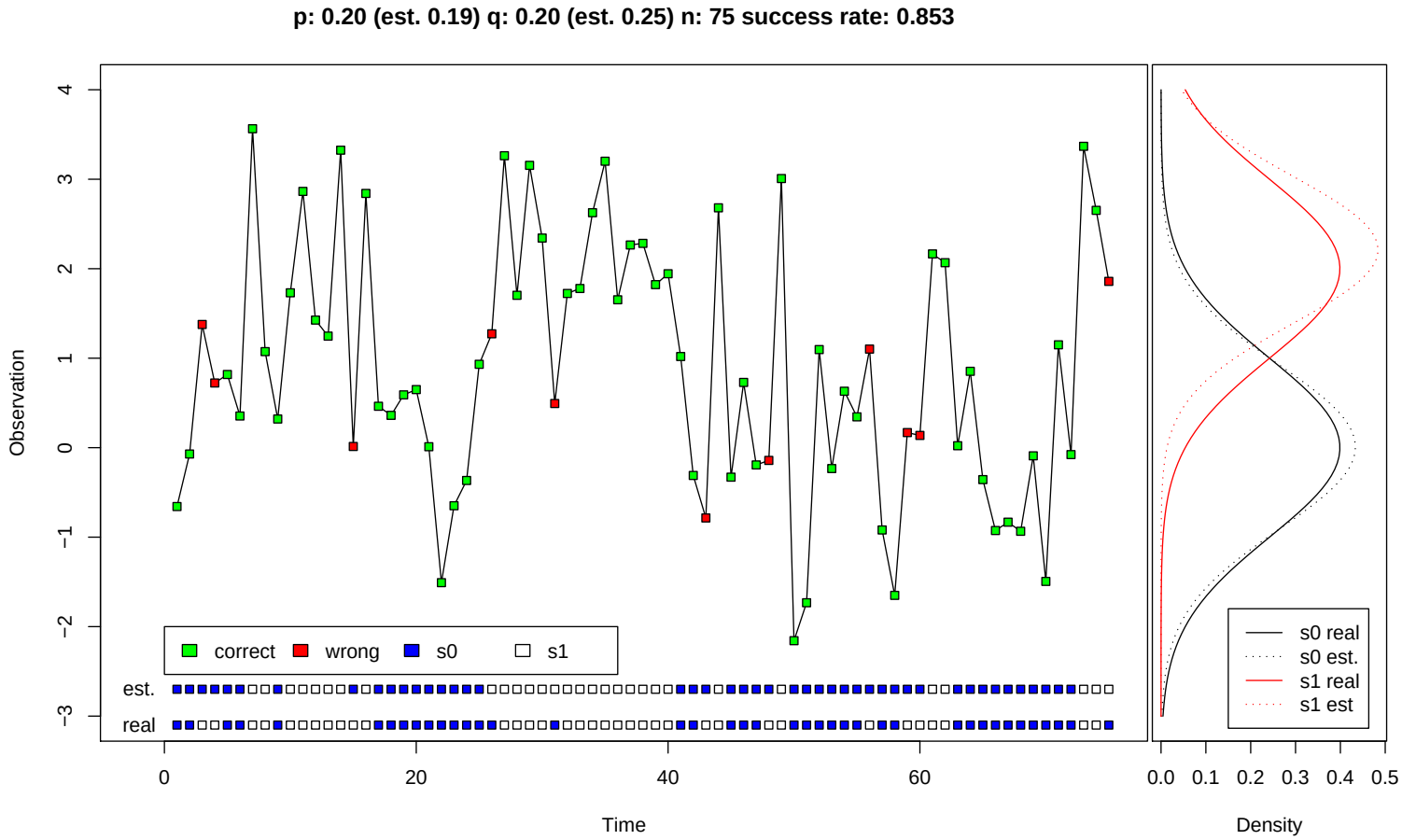

Figure 11: A single run with 75 observations

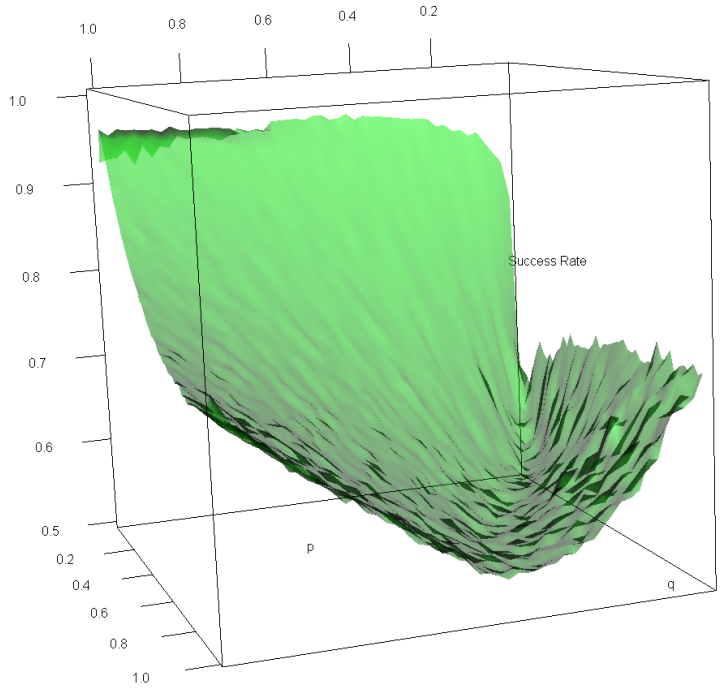

Figure 12: Success rate as a function of state switching probabilities for $Y_t|X_t = 0 \sim N(0, 0.5), Y_t|X_t = 1 \sim N(1, 1)$

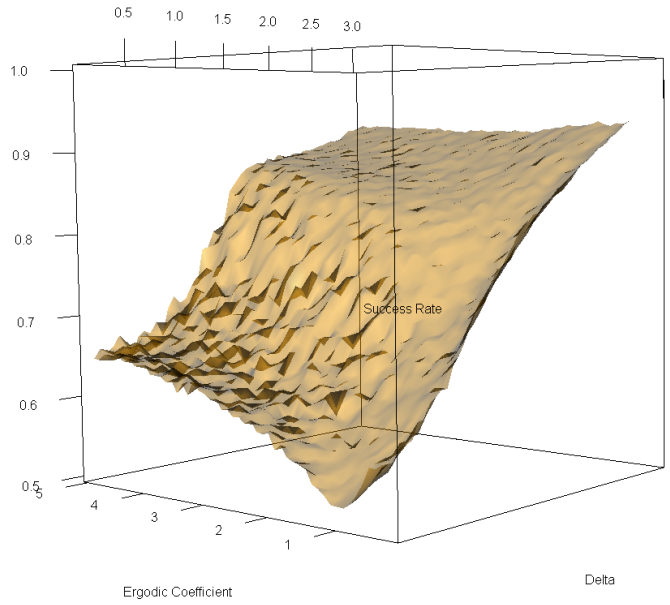

Figure 13: Success rate as a function of ergodic coefficient and $\delta = \mu_1 - \mu 0$ for $p = 0.5$

References

[1] http://www.arts.gla.ac.uk/ipa/ipa.html

[2] L. Rabiner and B. H. Juang: Fundamentals of Speech Recognition. New Jersey: Prentice Hall (1993)

[3] A. V. Oppenheim and R. W. Schafter: Zeitdiskrete Signalverarbeitung. München, Wien: Oldenbourg Verlag (1992).

[4] B. Logan: Mel Frequency Cepstral Coefficients for Music Modeling. International Symposium on Music Information Retrieval (2000).

[5] S. Mallat and W. L. Hwang: Singularity detection and processing with wavelets. IEEE Trans. Inform. Theory, vol. 38, pp. 617-643 (1992).

[6] J. P. Olive, A. Greenwood and J. Coleman: Acoustics of American English Speech - A Dynamic Approach. Springer Verlag (1993).

[7] S. Young, G. Evermann, T. Hain et al: The HTK Book. Cambridge University Engineering Department (2002).

[8] L. E. Baum: An inequality and associated maximization technique in statistical estimation for probabilistic functions of Markov processes. Inequalities, 3 (1972).

[9] M. Kumar, N. Rajput, and A. Verma: A large-vocabulary continuous speech recognition system for Hindi. IBM Journal of Research and Development, Volume 48, Number 5/6, (2004).

[10] L.R. Bahl, S. Balakrishnan-Aiyer, M. Franz, P.S. Gopalakrishnan, R. Gopinath, R. Novak, M. Padmanabhan, S. Roukos. The IBM Large Vocabulary Continuous Speech Recognition System for the ARPA NAB News Task. In Proceedings of ARPA Spoken Language System Technology Workshop, pp 121-126, (1995).

[11] P. Beyerlein, X. Aubert, R. Haeb-Umbach, M. Harris, D. Klakow, A. Wendemuth, S. Molau, H. Ney, M. Pitz and A. Sixtus. Large vocabulary continuous speech recognition of Broadcast News - The Philips/RWTH approach. In Speech Communication, Volume 37, pp 109-131, (2002).

[12] M.Y. Hwang. Subphonetic Acoustic Modeling for Speaker-Independent Continuous Speech Recognition. Ph.D. thesis, Tech Report No. CMU-CS-93-230, Computer Science Department, Carnegie Mellon University, (1993).

[13] P. Lamere, P. Kwok, E.B. Gouvêa, B. Raj, R. Singh, W. Walker, P. Wolf. The CMU Sphinx-4 Speech Recognition System. Proc. of the ICASSP 2003, (2003).

[14] R. Schwartz, T. Colthurst, N. Duta, H. Gish, R. Iyer, C.-L. Kao, D. Liu, O. Kimball, J. Ma, J. Makhoul, S. Matsoukas, L. Nguyen, M. Noamany, R. Prasad, B. Xiang, D.-X. Xu, J.-L. Gauvain, L. Lamel, H. Schwenk, G. Adda, L. Chen. Speech recognition in multiple languages and domains: the 2003 BBN/LIMSI EARS system. Acoustics, Speech, and Signal Processing, 2004. Proceedings. (ICASSP '04). (2004).

[15] L. Nguyen, T. Anastasakos, F. Kubala, C. LaPre, J. Makhoul, R. Schwartz, N. Yuan, G. Zavaliagkos, Y. Zhao. The 1994 BBN/BYBLOS Speech Recognition System. In Proceedings of ARPA Spoken Language Systems Technology Workshop, pp. 77-81, (1995).

[16] G. Evermann, H.Y. Chan, M.J.F. Gales, T. Hain, X. Liu, D. Mrva, L. Wang, P.C. Woodland. Development of the 2003 CU-HTK Conversational Telephone Speech Transcription System. Proceedings ICASSP 2004, Montreal, (2004).

[17] I. Kirschinger, H. Tomabechi and J.-I. Aoe. Recent Advances in Continuous Speech Recognition Using the Time-Sliced Paradigm. International Workshop on Soft Computing in Industry, Muroran, Japan, (1996).

[18] E. Trentin, M. Gori. Robust combination of neural networks and hidden Markov models for speech recognition. Neural Networks, IEEE Transactions on , vol.14, no.6pp. 1519- 1531, (2003).

[19] W. Macherey, L. Haferkamp, R. Schlüter and H. Ney. Investigations on Error Minimizing Training Criteria for Discriminative Training in Automatic Speech Recognition. In the 9th European Conference on Speech Communication and Technology (Interspeech 2005), Lisbon, Portugal, (2005).

[20] V. Valtchev. Discriminative Methods in HMM-based Speech Recognition. PhD Thesis, Cambridge University, Department of Electrical Engineering, (1995).

[21] S. Kapadia. Discriminative Training of Hidden Markov Models. PhD Thesis, Cambridge University, Department of Electrical Engineering, (1998).

[22] P.C. Woodland and D. Povey. Large scale discriminative training for speech recognition. Proc. ISCA ITRW ASR2000, (2000).

[23] N. Marhav and C.-H. Lee: On the asymptotic statistical behavior of empirical cepstral coefficients. IEEE Transactions and Signal Processing 41, (1990-1993).

[24] Y. Normandin: Maximum Mutual Information Estimation of Hidden Markov Models. Automatic Speech and Speaker Recognition: Kluwer Academic Publishers (1996).

[25] http://en.wikipedia.org/wiki/Place_of_articulation

[26] http://en.wikipedia.org/wiki/Manner_of_articulation

[27] A.P. Dempster, N.M. Laird and D.B. Rubin. Maximum-likelihood from incomplete data via the em algorithm. J. Royal Statist. Soc. Ser.B., 39, (1977).

[28] R. Redner and H. Walker. Mixture densities, maximum likelihood and the em algorithm. SIAM Review, 26(2), (1984).

[29] Z. Ghahramami and M. Jordan. Learning from incomplete data. Technical Report AI Lab Memo No. 1509, CBCL Paper No. 108, MIT AI Lab, (1995).

[30] M. Jordan and R. Jacobs. Hierarchical mixtures of experts and the em algorithm. Neural Computation, 6:181-214, (1994).

[31] C.F.J. Wu. On the convergence properties of the em algorithm. The Annals of Statistics, 11(1):95-103, (1983).

[32] R Development Core Team. R: A language and environment for statistical computing. R Foundation for Statistical Computing, (2008).