UNIVERSITÄT PADERBORN
Die Universität der Informationsgesellschaft

FAKULTÄT FÜR
ELEKTROTECHNIK,
INFORMATIK UND
MATHEMATIK

Institut für Elektrotechnik und Informationstechnik
Universität Paderborn
Fachgebiet Nachrichtentechnik
Prof. Dr.-Ing. Reinhold Häb-Umbach

Master's Thesis

Non Parametric Bayesian Acoustic Model Discovery for phoneme classification

by

Tanuj Jain
Matr.-Nr.: 6750612

Supervisor: Dipl.-Ing. Oliver Walter
Filing date: October 26, 2015
Number: EIM/NT - 3/15

Eidesstattliche Erklärung

Ich erkläre hiermit an Eides statt, dass ich die vorliegende Arbeit selbstständig und unter Verwendung der angegebenen Hilfsmittel angefertigt habe; die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht.
Die Arbeit wurde bisher noch keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Paderborn, den	______________________	______________________
	(Datum)	(Unterschrift)

Acknowledgement

I would sincerely like to thank my supervisor Dipl.-Ing. Oliver Walter for his immense help and patience throughout the duration of the thesis. I would also like to thank Prof. Dr.-Ing. Reinhold Häb-Umbach for providing me with this opportunity of working on this exciting topic.

A word of gratitude also goes out to the entire department of Fachgebiet Nachrichtentechnik and fellow students who showed patience while my long duration simulations ran on multiple computers.

Abstract

Speech produced using a specific language can be broken down into basic acoustic units referred to as phonemes. A combination of these sub-word acoustic units i.e. phonemes, produces meaningful units such as words. The task of recognizing acoustic structure in speech is performed seamlessly by human auditory system, but reproducing this ability in a machine is a challenging pursuit. What makes the problem more complex is the unavailability of labelled training data bases.

The aim of this thesis is to formulate an algorithm for the automatic discovery of phonemes from unlabelled speech recordings and learn the parameters of those phoneme models.

The non-parametric acoustic modelling task comprises 3 basic sub-tasks:

- Segmentation: Obtaining speech segments directly from speech samples.
- Clustering: Clustering of similar segments together.
- Learning: Learning the model parameters for each cluster.

Each cluster is modelled using a Hidden Markov Model (HMM). HMMs are ideally suited for modelling of data having temporal coherence such as speech data. The task of segmentation is carried out using Viterbi algorithm. The method used to cluster and learn the model parameters is called Gibbs Sampling which is applied in conjunction with Bayesian inference.

Contents

Introduction

Understanding acoustic structures in a language is critical to the process of language acquisition in humans [Kuh04]. Teaching machines to do the same is an interesting venture since it will allow us to create intelligent systems that comprehend human speech. Such a system which is language independent can be used in any part of the world on any language.

The first step towards solving this problem is to understand the phonetic structure of speech [KSH+06], [Fry77], [LCSSK67], [Eim75], [Ste00]. Phonemes [Oha74] are the building blocks of human speech. Also referred to as sub word acoustic units, different phonemes combine to give rise to meaningful entities like words. For example, the English word *'Chef'* is composed of three phonemes: /ʃ/,/e/ and /f/. Therefore, as a first step, it is important to obtain these sub word units which differ depending upon the language. Once the phonetic structure has been learnt, one can solve the next set of problems which are related to understanding the lexical content and grammatical structure of a given language.

One of the barriers in developing an Acoustic Speech Recognition (ASR) system for recognition of phonemes is the absence of labelled training data for a given language. Therefore, the problem falls under the category of unsupervised acoustic speech recovery. Making an ASR system which is language independent requires the number of phonemes to be variable since each language has a different number of phonemes. This makes the problem non-parametric in nature in this case.

The problem of unsupervised non-parametric acoustic unit discovery is thus a problem of determination of the phonetic structure of a language with no language specific information or labelled databases available. The sub acoustic unit discovery is a task that comprises 3 basic sub-tasks [GG06]:

- Segmentation: Obtaining phoneme segments directly from a continuous stream of speech samples.
- Clustering: Clustering of similar segments together.
- Learning: Learning the model parameters for each cluster.

Earlier approaches treated these sub problems as independent. This implied that the segmentation step was independent of the clustering and learning steps [LSJ88], [GG06], [CL11]. The clustering step was also treated independently of the learning step. This means that the variables involved in determining the segmentation boundaries had no part to play in the clustering step or the learning of model parameters and vice-versa. In this thesis, the task has been approached using non-parametric Bayesian techniques wherein the sub tasks of clustering and learning have dependence on each other. Thus, the variables involved in clustering step are also involved in the process of learning the attributes of these clusters.

A Hidden Markov Model or HMM provides an effective way to model speech data due to its ability to cater to temporally correlated data [JR91], [Rab89]. Each phoneme is hence, modelled using an HMM. The concern is then the determination of the number of HMMs and individual attributes of each HMM [LG12]. Baum-Welch algorithm has been the traditional Expectation Maximization approach towards determination of HMM parameters [Rab89]. Chapter-2 deals with an explanation of the theory behind Hidden Markov Models and also discusses the Baum-Welch algorithm in detail. However, it doesn't provide a way to infer the number of HMMs for a given system. In this thesis, Dirichlet non parametric modelling has been used to determine the number of HMMs in a given a system. A sampling approach called Gibbs sampling [CG92] has been used to obtain the attributes of the discovered HMMs.

Gibbs sampling is essentially a Markov Chain Monte Carlo (MCMC) method of sampling that uses conditional densities of random variables to sample from their joint distribution. All the model parameters are considered to be a part of a single joint distribution. Gibbs sampling is then applied iteratively to sample the values of these parameters from the conditional distribution of each parameter given all the other parameters. Cluster labels for each phoneme sequence and the attributes of each HMM that models an individual phoneme class are determined using Gibbs sampling in this thesis. Chapter-3 discusses the Gibbs sampling theory.

Gibbs sampling works by using conditional distributions on the model parameters given all the other model parameters. Using a prior on the model parameters to attain their posterior conditional distribution integrates Bayesian learning approach with Gibbs sampling. Therefore, the problem can now be summarised as follows: Determination of segments, clusters and cluster parameters by application of Viterbi algorithm and Gibbs sampling in conjunction with Bayesian Learning on HMMs.

1.1 Problem Description

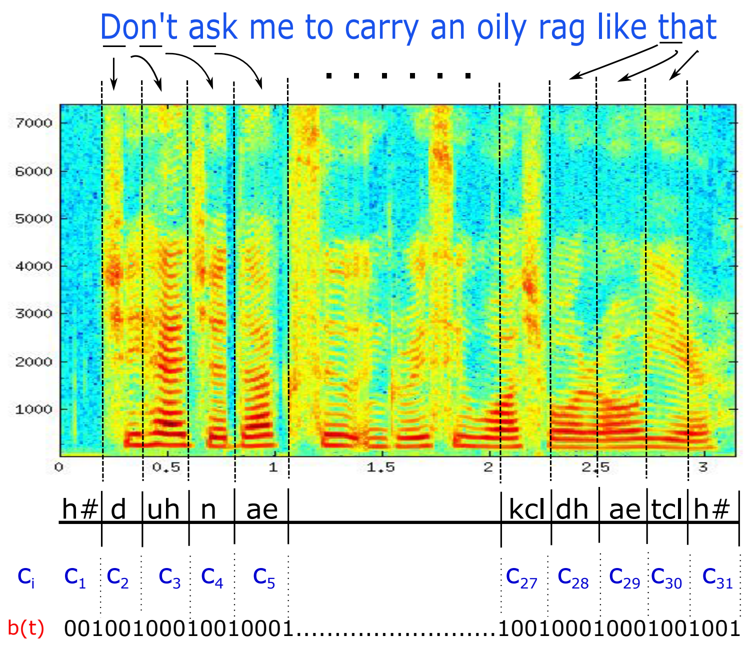


Figure 1.1: Spectrogram of a TIMIT recording

The figure 1.1 shows a spectrogram [OT01] of an utterance from the TIMIT database [GC$^+$93]. The figure also approximately maps the parts of the spectrogram to the utterance of the phoneme within the sentence. Thus, the first phoneme in the utterance is */h#/* (indicaing silence) followed by */d/* and so on. The following challenges need to be addressed:

1. **Segmentation**: The boundaries between the phonemes in a given recording need to be determined. A set of boundary variables are used to accomplish this task. The boundary variable corresponding to the last sample in a sequence is set to 1. This is shown in the figure using $b(t)$. As can be seen, the value of $b(t)$ for the samples which are the last in a given phoneme sequence is set to 1 while the value of other boundary variables remain 0.

2. **Clustering**: Similar phonemes need to be assigned the same cluster label. Each cluster is modelled using a Hidden Markov Model (HMM). The cluster labels (c_i), which denote the cluster label of the i-th sequence are shown in the figure. In the figure, there are 31 phonemes i.e., 31 sequences. Thus, there are 31 cluster labels c_1 to c_{31}. The goal of the clustering step is to assign same values to the cluster labels for the sequences that correspond to the same phoneme. Therefore,

for the figure 1.1, the cluster labels of the sequences corresponding to phoneme */ae/* should have the same value i.e., c_5 and c_{29} should be assigned the same values. This is also true for sequences 1 and 31 which correspond to phoneme */h#/*. Therefore, labels c_1 and c_{31} should also have the same value. This signifies that these sequences belong to the same HMM.

3. **Learning**: After similar sequences have been clustered together, the attributes of the HMM characterising the cluster need to be learned. For this, all the phoneme sequences which have the same values of the cluster label are used to learn the attributes of an HMM that defines that particular cluster. For instance, sequences with cluster label c_5 and c_{29} are used to learn the attributes of a particular HMM while that with cluster label c_1 and c_{31} are used to learn attributes of a different HMM and so on.

Chapter-4 describes the system and discusses each of the above steps in detail.

A variety of useful metrics can be used to evaluate the quality of clustering [Pet06], [RV96], [Tow71]. One of the metrics is cluster purity [MN03]. Cluster purity gives a measure of how pure a cluster is i.e., how well the majority class is represented in an individual cluster. Cluster purity metric can also be calculated over all the learnt clusters to give a single number which gives information about the purity of all the clusters as a whole. The higher the cluster purity, the better is the clustering performance. For determining the performance of the segmentation algorithm, *recall* [MGT03] is calculated to get the percentage of true phoneme boundaries discovered by the segmentation algorithm. These measures, along with the simulation scenarios is discussed in chapter-5. The results of various experiments are presented in chapter-6.

Chapter-7 discusses the summary of the work done in this thesis and presents the conclusions drawn from the experiments performed. Some suggestions regarding the future work is also provided in the chapter.

A general discussion on Hidden Markov models (HMM) along with the Baum-Welch algorithm is presented in the next chapter.

Hidden Markov Models

Hidden Markov Model or HMM is a type of stochastic signal model that has been used to model phonemes in this thesis. Speech data has inherent temporal continuity. HMMs lend themselves well to speech data modelling as they can model data having temporal continuity [JR91]. This chapter discusses the basics of HMM with special regards to various problems HMMs pose. A solution to these problems is also presented since these will be used at various stages in the thesis.

Before understanding the Hidden Markov Model approach to modelling, one must understand the basics of Markov chains, which is presented in the next section.

2.1 Markov Chains

Consider a system that can be characterized at a given time as being in one of **N** distinct states namely, $s_1, s_2, s_3, \cdots, s_N$. This is depicted in figure 2.1.

At different time instances, denoted by $t = 1, 2, \cdots$, the system may be in a particular state, denoted by q_t, depending upon a set of probabilities. These probabilities, which are also the parameters of the Markov chain models, are:

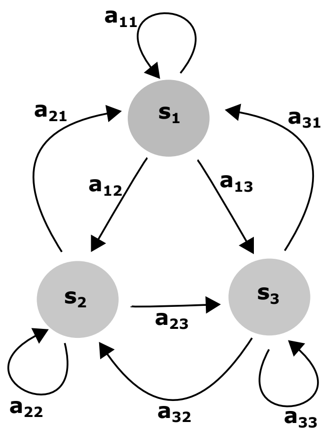


Figure 2.1: State Transition Diagram

1. Transition Probabilities (a_{ij}): The probability of being in a particular state at time instant t given the states q at earlier time instants i.e., $P[q_t = s_j | q_{t-1} = s_1, q_{t-2} = s_k, \cdots]$.
   If first order assumption is used i.e., only the state at previous instant of time affects the present state, the expression becomes,
   $$P[q_t = s_j | q_{t-1} = s_i, q_{t-2} = s_k, \cdots] = P[q_t = s_j | q_{t-1} = s_i]$$
   This is denoted by a_{ij} with properties-
   $$a_{ij} \geq 0$$
   $$\sum_{j=1}^{N} a_{ij} = 1$$
   This is also shown in the figure 2.1. For example, a_{12} denotes the probability of transitioning from state-1 to state-2. For a 3 state system like the one shown in figure 2.1, this can be represented as a 3×3 matrix denoted by $\mathbf{A}$ with entries as a_{ij}.
2. Initial State Probabilities (π_i): These signify the probability of being in state s_i at first time instant. Obviously, $\sum_{i=1}^{N} \pi_i = 1$

Given the model parameters, one could obtain the probability of occurrence of a sequence of states ($\mathbf{S}$) i.e.,

$$\begin{aligned} P(\mathbf{S}|\mathbf{A}, \Pi) &= P(q_4 = s_3, q_3 = s_2, q_2 = s_2, q_1 = s_1 | \mathbf{A}, \Pi) \\ &= P(q_1 = s_3) \cdot P(q_2 = s_2 | q_1 = s_3) \cdot P(q_3 = s_2 | q_2 = s_2) \cdot P(q_4 = s_1 | q_3 = s_2) \\ &= \pi_3 \cdot a_{32} \cdot a_{22} \cdot a_{21} \end{aligned}$$

where, $\Pi = [\pi_1, \pi_2, \cdots, \pi_N]$

2.2 Extension to Hidden Markov Models

In the Markov chain model, it was assumed that the states could be directly observed. However, in many real life problems, the states are not directly observable; they are hidden. Instead, observations that correspond to these hidden states are observed. These observations are connected to the actual hidden states through a set of observation probabilities. Therefore, the actual stochastic process responsible for production of states is observable only through a set of observations which will be denoted by $\mathbf{X} = \{X_1, X_2, \cdots, X_T\}$ as shown in figure 2.2.

As can be seen from 2.2, corresponding to state $q_1 = s_1$, the observation X_1 is obtained. Here the following assumptions can be made:

1. An observation is dependent only upon the state responsible for producing it; it is independent from any other states. This implies:
   $$P(X_t | q_1, \cdots, q_t, \cdots, q_T) = P(X_t | q_t = s_i)$$

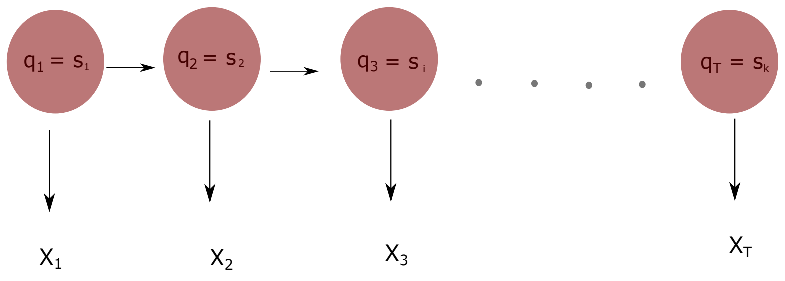


Figure 2.2: Hidden Markov Model depiction

2. An observation is independent of other observations, which leads to:

$$P(X_t|X_1, \cdots, X_{t-1}, X_{t+1}, \cdots, X_T) = P(X_t)$$

As the states are hidden, there is an observation probability associated with each observation i.e., the probability of an observation being generated by a state. For a discrete observation space $X \in \{l_1, l_2, \cdots, l_k, \cdots, l_K\}$, this is denoted by $\mathbf{B} = \{b_i(k)\}$ which defines the probability of observing the k-th (l_k) symbol in state s_i at time t i.e.,

$$b_i(k) = P(X_t = l_k|q_t = s_i)$$

For a continuous observation space, $\mathbf{B}$ is represented using a continuous probability density function (pdf) for each state. Thus, the probability of getting an observation from the pdf representing the state becomes the observation probability for that observation-state pair.

Thus the set of parameters that define an HMM are:

$$\theta = (\mathbf{A}, \mathbf{B}, \Pi)$$

A trellis diagram is often used to show the different parts of the HMM and the various HMM parameters as shown in figure 2.3. Figure 2.3 shows a three state system with the transition probabilities a_{ij} marked for $i = \{1, 2, 3\}$ and $j = 1$. Also shown are the observation probabilities $b_i(k)$ at all time instants for corresponding state i.

2.3 Basic Problems Of HMM

There are three problems that need to be solved for the HMM model presented in the last section:

(i) Evaluation problem: Given an observation sequence $\mathbf{X} = \{X_1, X_2, \cdots, X_t, \cdots, X_T\}$, compute the probability of observation sequence $P(\mathbf{X}|\theta)$ for a model with parameters θ.

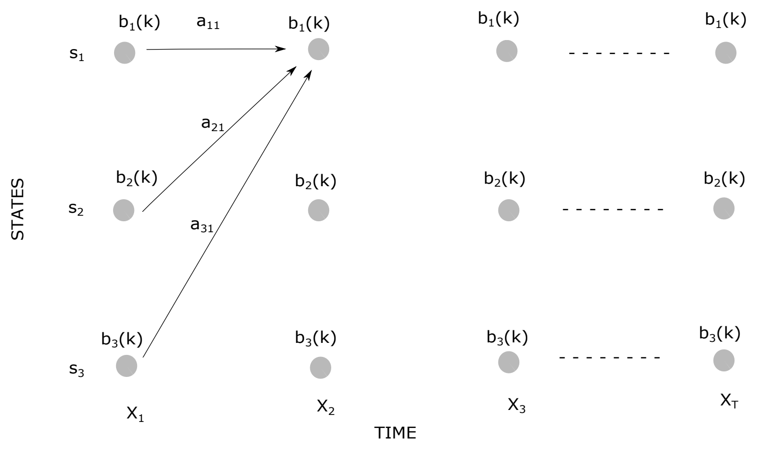


Figure 2.3: Trellis diagram

(ii) Optimality problem: Given an observation sequence $\mathbf{X} = \{X_1, X_2, \cdots, X_t, \cdots, X_T\}$ and a model θ, find the most optimal sequence of states $\mathbf{Q} = \{q_1, q_2, \cdots, q_i, \cdots, q_T\}$ that justifies the observations.

(iii) Learning problem: Given an observation sequence $\mathbf{X} = \{X_1, X_2, \cdots, X_t, \cdots, X_T\}$ and a model θ, modify the parameters in order to maximise $P(\mathbf{X}|\theta)$.

2.3.1 Solution to evaluation problem

Given: Observation sequence $\mathbf{X} = \{X_1, X_2, \cdots, X_T\}$, Model $\theta = (\mathbf{A}, \mathbf{B}, \Pi)$.
Task: To determine $P(\mathbf{X}|\theta)$.
Solution:
Consider a forward variable defined as: $\alpha_t(i) = P(X_1, X_2, \cdots, X_t, q_t = s_i|\theta)$
This denotes the probability of occurrence of partial observation sequence $\{X_1, X_2, \cdots, X_t\}$ and the state $q_t = s_i$ given the model. The target probability can be calculated as:

$$P(\mathbf{X}|\theta) = \sum_{i=1}^{N} P(\mathbf{X}, q_t = s_i|\theta) = \sum_{i=1}^{N} \alpha_T(i)$$

<u>Calculation of α</u>

α is calculated in three steps:

1. Initialization:

$$\alpha_1(i) = \pi_i b_i(X_1)$$

where i is an index over N which signifies the number of states in the HMM.

2. Induction:

$$\alpha_{t+1}(j) = \Big[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\Big] b_j(X_{t+1}), \quad 1 \le t \le T-1$$
$$1 \le j \le N$$

3. Termination:

$$P(\mathbf{X}|\theta) = \sum_{i=1}^{N} \alpha_T(i)$$

2.3.2 Solution to optimality problem

Given: Observation sequence $\mathbf{X} = \{X_1, X_2, \cdots, X_T\}$, Model $\theta = (\mathbf{A}, \mathbf{B}, \Pi)$.
Task: To determine $\mathbf{Q} = \{q_1, q_2, \cdots, q_i, \cdots, q_T\}$ i.e., the most optimal sequence of states that is associated with the observation sequence $\mathbf{X}$.
Solution: There are many different ways of approaching this problem since the definition of optimality is subjective. Three variants will be discussed here:

2.3.2.1 Variant-1: Select states q_t which are individually most likely

The optimality criterion for this variant is to select states q_t which are individually most likely. For achieving this optimality criteria, a variable γ is defined as:

$$\gamma_t(i) = P(q_t = s_i|\mathbf{X}, \theta)$$

This gives the probability of being in the state $q_t = s_i$ given all observations $\mathbf{X}$ and model θ. For calculation of γ, a backward variable is needed:

$$\beta_t(i) = P(X_{t+1}, X_{t+2}, \cdots, X_T|q_t = s_i, \theta)$$

β signifies the probability of observing all the future observations given the current state $q_t = s_i$ and the model θ.

Calculation of β

1. Initialization:

$$\beta_T(i) = 1, \quad 1 \le i \le N$$

2. Induction:

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(X_{t+1}) \beta_{t+1}(j),$$
$$t = T-1, T-2, \cdots, 1,$$
$$1 \le i \le N$$

Calculation of γ

γ can now be calculated as:

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{\sum_{i=1}^{N} \alpha_t(i)\beta_t(i)}$$

Now the most likely state for a time instant t can be obtained as:

$$q_t = \underset{i}{\operatorname{argmax}}(\gamma_t(i))$$

The drawback of this optimality criteria is that it ignores the states that precede or succeed the present state; only the most likely state at an instant is determined without any regard to the sequence of states. This may lead to a sequence of states which is invalid or highly unlikely.

2.3.2.2 Variant-2: Viterbi algorithm

This variant overcomes the drawbacks of variant-1 by attempting to find the most likely sequence of states that justify a given set of observations. It is known as *Viterbi algorithm.*

The Viterbi Algorithm requires definition of two variables:

1. Probability variable δ: This variable contains the probability that accounts for the path with the highest probability upto the time instant t with $q_t = s_i$ defined as:

$$\delta_t(i) = \max_{q_1, q_2, \cdots, q_{t-1}} P(q_1, q_2, \cdots, q_t = s_i, X_1, X_2, \cdots, X_t|\theta)$$

   This can be calculated using induction:

$$\delta_{t+1}(j) = \left[\max_i \delta_t(i) a_{ij}\right] \cdot b_j(X_{t+1})$$

2. Path Variable Ψ: This variable keeps track of the state which is most likely to have produced the current state. It is denoted by $\Psi_t(j)$. For each time instant t, this is an array keeping track of most likely last state in the previous time instant that produced the current state $q_t = s_i$.

Viterbi algorithm is carried out in four steps:

1. Initialization:

$$\delta_1(i) = \pi_i b_i(X_1) \quad 1 \le i \le N$$

$$\Psi_1(i) = 0$$

2. Recursion:

$$\delta_t(j) = \max_{1\leq i\leq N} \Big[\delta_{t-1}(i)a_{ij}\Big] b_j(X_t), \quad 2 \leq t \leq T$$

$$1 \leq j \leq N$$

$$\Psi_t(j) = \operatorname*{argmax}_{1\leq i\leq N} \Big[\delta_{t-1}(i)a_{ij}\Big], \quad 2 \leq t \leq T$$

$$1 \leq j \leq N$$

3. Termination: Once the values of $\delta_T(i)$ have been calculated for all states, the state with the highest value of $\delta_T(i)$ becomes the most probable last state in the sequence. The $\Psi_t(i)$ corresponding to this state is then selected to obtain the most likely state in the previous state that generated this state at this time instant. This is stored in q_T^* and corresponding path probability is stored in variable P_{max}:

$$P_{max} = \max_{1\leq i\leq N} \Big[\delta_T(i)\Big]$$

$$q_T^* = \operatorname*{argmax}_{1\leq i\leq N} \Big[\delta_T(i)\Big]$$

4. Path backtracking: After obtaining q_T^*, backtracking upto the first time instant is done to obtain the most likely sequence of states:

$$q_t^* = \Psi_{t+1}(q_{t+1}^*), \quad t = T-1, T-2, \cdots, 1$$

An example of Viterbi algorithm is shown in figure 2.4 for a three state system with four observations ($T = 4$). As can be seen from the figure, the value of variable δ_1 is the the highest. The path stored in variable $\psi_4(1)$ is shown in colour red as $\psi_4(1) = \{s_2, s_2, s_1, s_1\}$. This is the most probable sequence of states according to the Viterbi algorithm.

2.3.2.3 Variant-3: Backward sampling

This variant is a technique referred to as *Backward sampling.* As the name suggests, it is an approach that combines sampling with backtracking. A vector $\underline{v}_t = \{v_t(1), v_t(2), \cdots, v_t(N)\}$ is defined for $t \in \{1, T-1\}$ with $v_t(i)$ given as:

$$v_t(i) = P(X_1, X_2, \cdots, X_t, q_t = s_i, q_{t+1} = s_j|\theta)$$

$$= P(X_1, X_2, \cdots, X_t, q_t = s_i|\theta) \cdot P(q_{t+1} = s_j|q_t = s_i)$$

$$= \alpha_t(i) \cdot a_{ij}$$

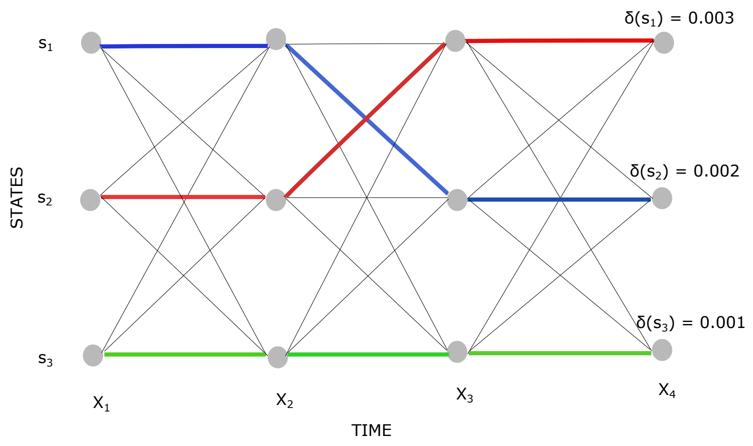


Figure 2.4: Viterbi algorithm with paths for different most likely end states

with s_j known for $t = T$.

The state of the sample at the last instant T can be sampled using the set of probabilities $\{\alpha_T(j)\}$. In specific problems, the state of the last sample might be known due to model assumptions. Once this vector is obtained for a time instant t, the vector can be normalised and its entries can be treated as inputs to a Categorical distribution [ML06] i.e.,

$$P(q_t = s_i|X_1, X_2, \cdots, X_t, q_{t+1} = s_j, \theta) = Cat(q_t|v_t(i = 1), v_t(i = 2), \cdots, v_t(i = N))$$

Here the entries of the vector $\underline{v}_t$ have been normalised so that they sum to 1.

Sampling from this distribution gives the state q_t. An example of this variant is shown in figure 2.5. A trellis with a three state HMM and four observations is shown along with the corresponding forward variable $\alpha_t(j)$ for all states and time instants. If the last state is assigned as the state-3, then the state at time instant $t-1$ i.e., at $t = 3$ can be obtained by defining the vector $\underline{v}_3(i) = \{\alpha_3(1) \cdot a_{13}, \alpha_3(2) \cdot a_{23}, \alpha_3(3) \cdot a_{33}\}$. Now this vector is normalised and used as an input to the categorical distribution for q_3 to sample the appropriate state as:

$$q_3 \sim P(q_3|\underline{v}_3(i)) = Cat(q_3|\underline{v}_3(i))$$

2.3.3 Solution to learning problem

Given: Observation sequence $\mathbf{X} = \{X_1, X_2, \cdots, X_T\}$, Model $\theta = (\mathbf{A}, \mathbf{B}, \Pi)$
Task: To learn the model parameters $\theta = (\mathbf{A}, \mathbf{B}, \Pi)$ to maximize the probability of occurrence of the observations given the model i.e., to maximize $P(X_1, X_2, \cdots, X_T|\theta)$
Solution: The classic expectation maximization (EM) way of approaching the problem is given by *Baum Welch algorithm* [Rab89] which will be discussed next.

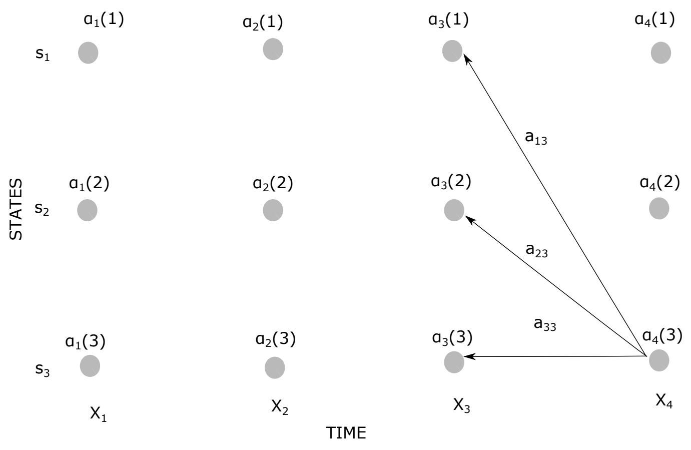


Figure 2.5: Back sampling illustration

2.3.3.1 Baum Welch Algorithm

A variable ξ needs to be calculated as:

$$\xi_t(i,j) = P(q_t = s_i, q_{t+1} = s_j | \mathbf{X}, \theta)$$

$\xi_t(i,j)$ is thus equivalent to the probability of being in state s_i at time instant t and in state s_j at time *t+1*. Using the forward variable(α) and the backward variable(β), ξ can be defined as:

$$\xi_t(i,j) = \frac{\alpha_t(i) a_{ij} b_j(X_{t+1}) \beta_{t+1}(j)}{P(\mathbf{X}|\theta)}$$

$$= \frac{\alpha_t(i) a_{ij} b_j(X_{t+1}) \beta_{t+1}(j)}{\sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_t(i) a_{ij} b_j(X_{t+1}) \beta_{t+1}(j)}$$

The denominator is just used for normalization.

$\gamma_t(i)$ can now be calculated by summing over the probabilities of all the states the system can go from a state s_i at time instant t i.e.,

$$\gamma_t(i) = \sum_{j=1}^{N} P(q_t = s_i, q_{t+1} = s_j | \mathbf{X}, \theta)$$

$$= \sum_{j=1}^{N} \xi_t(i,j)$$

The variables ξ and γ can now be used to update the model parameters.

Update of Π:

π represents the number of times the system may be in a particular state at the initial time instant. It can be calculated as:

$$\pi_i = \gamma_1(i)$$

Update of a_{ij}:

a_{ij} is the probability of the system transitioning from state s_i to s_j. It can thus be calculated as follows:

$$a_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)}$$

The above expression is the ratio of expected number of times the system makes a transition from state s_i to state s_j to the average number of times the system is in state s_i.

Update of $\mathbf{B}$:

The update of $\mathbf{B}$ depends upon whether the observation probabilities are discrete or continuous.

- Case - 1: Discrete observation probabilities:

  In this type of HMM, there are only a finite number of discrete values $L = \{l_1, l_2, ..., l_k\}$ the observation may take. This is often the case with communication systems [Mob00]. In this case, the observation probability for l_k given a state s_i can be calculated as the ratio of the average number of times the system is in state s_i with observation l_k to the average number of times the system is in state s_i i.e.,

$$b_i(k) = \frac{\sum\limits_{\substack{t=1 \\ X_t = l_k}}^{T} \gamma_t(i)}{\sum\limits_{i=1}^{T} \gamma_t(i)}$$

- Case - 2: Continuous Observation probabilities

  In this type of HMM, observations are continuous as opposed to a finite set of discrete symbols. Thus, continuous probability density functions (pdf) are required to represent these observations. One such pdf could be a Gaussian mixture model (GMM) which can be represented as:

$$b_j(X) = \sum_{m=1}^{M} w_m \mathcal{N}(X, \mu_m, \mathbf{C}_m)$$

The above density is a mixture density where w_m is the weight the m-th mixture receives out of a total of M mixtures with the property:

$$\sum_{m=1}^{M} w_m = 1$$

The parameters μ_m and $\mathbf{C}_m$ are the means and covariances respectively for the m-th mixture Gaussian pdf.

The updates for the parameters are obtained as:

$$w_{jk} = \frac{\sum\limits_{t=1}^{T} \gamma_t(j,k)}{\sum\limits_{t=1}^{T} \sum\limits_{k=1}^{M} \gamma_t(j,k)}$$

$$\mu_{j,k} = \frac{\sum\limits_{t=1}^{T} \gamma_t(j,k) \cdot X_t}{\sum\limits_{t=1}^{T} \gamma_t(j,k)}$$

$$\mathbf{C}_{j,k} = \frac{\sum\limits_{t=1}^{T} \gamma_t(j,k) \cdot (X - \mu_{j,k})(X - \mu_{j,k})^{`}}{\sum\limits_{t=1}^{T} \gamma_t(j,k)}$$

Here, $\gamma_t(j,k)$ is defined as:

$$\gamma_t(j,k) = \left[\frac{\alpha_t(j)\beta_t(j)}{\sum\limits_{j=1}^{N} \alpha_t(j)\beta_t(j)}\right] \left[\frac{w_{jk}\mathcal{N}(X_t, \mu_{j,k}, \mathbf{C}_{jk})}{\sum\limits_{m=1}^{M} w_{j,m}\mathcal{N}(X_t, \mu_{jm}, \mathbf{C}_{jm})}\right]$$

The above updates correspond to the EM way of updating parameters.

An alternative learning approach to the Baum Welch algorithm is *Gibbs sampling* and will be presented in the next chapter.

Gibbs Sampling

This chapter presents an alternative solution to the learning problem of HMM presented in the last chapter i.e., to learn the model parameters θ given the observations ($\mathbf{X}$). This alternative is a technique called *Gibbs sampling.*

Gibbs sampling is one of the Markov chain Monte Carlo (MCMC) methods which are used to generate samples from probability distributions. Metropolis algorithm, described in [MU49] and [MRR$^+$53], formed the basis of the MCMC methods. A generalization of the aforementioned work lead to Metropolis-Hastings theorem described in [Has70]. Gibbs sampling was then used for the analysis of image data as discussed in [GG84]. MCMC methods employ the stationarity property of Markov chains to work which will be discussed in the next section. This chapter also includes an explanation of the Gibbs sampler and the way it can be used to learn the parameters of a distribution.

Bayesian inference is incorporated in many statistical problems as it gives a better solution by avoiding point estimates of quantities being estimated. It can also be used in conjunction with Gibbs sampling using *conjugate priors* which will be discussed in the last section of the chapter.

3.1 Stationarity of Markov Chains

Consider a Markov chain over a state space $\Omega = \{s_1, s_2, ..., s_N\}$ and attributes $\mathbf{A} = \{a_{ij}\}$, $\Pi_{st} = \{\pi_1, \pi_2, ..., \pi_N\}$ where Π_{st} denotes the stationary distribution. The Markov chain is said to be stationary when following condition is achieved [Fel08]:

$$\Pi_{st} = \Pi_{st}\mathbf{A} \tag{3.1}$$

This means that the Markov chain becomes stationary when the probability of appearance of a state is no longer dependent upon the transition probabilities $\{a_{ij}\}$. Therefore, it is as though the samples generated after achieving stationarity are generated from the true distribution. The stationarity condition of 3.1 also implies the following relation known as the *detailed balance equation*:

$$\pi_i \cdot a_{ij} = \pi_j \cdot a_{ji} \tag{3.2}$$

For the Markov chain to become stationary, it must fulfil two conditions:

1. Irreducibility: The Markov chain must be irreducible. This means that for a Markov chain with N states, it must be possible to reach any state from any other state i.e., there exists an integer 't' such that:

   $$P^t_{ij} > 0; \quad i, j \in 1, ..., N$$

   where P^t_{ij} denotes the probability of reaching state s_i from state s_j in t steps.

2. Aperiodicity: The Markov chain must be aperiodic. This means that for a Markov chain with N states, there is no state s_i and integer d except $d = 1$, for which $P^d_{ij} = 1$. This means that there is no guarantee of a state being observed after every d steps. This implies that the Markov chain at any time instant does not depend upon the initial state.

The example discussed above shows the construction of a Markov chain for a discrete state space. Often the state space is continuous rather than discrete. This is also the case with acoustic speech recognition. The variables involved with this problem are usually continuous in nature. Thus, the stationary distribution is also a continuous distribution. The equivalent stationarity conditions for continuous state space is:

$$\Pi(x) = \int \Pi(y) P(x|y) dy$$
$$\Pi(y) = \int \Pi(x) P(y|x) dx$$

The detailed balance equation is:

$$\Pi(x) P(x|y) = \Pi(y) P(y|x)$$

In our problem, observations are continuous in nature. The parameters of the densities that model these observations are also continuous. All these observations and parameters form a continuous state space from which a Markov chain is constructed. Markov chain Monte Carlo methods, discussed on the upcoming sections discuss how Markov chains are utilised to estimate these densities and parameters.

3.2 Markov Chain Monte Carlo (MCMC) methods

Once a Markov chain becomes stationary, the samples drawn from the distribution $\Pi\mathbf{A}$ can be interpreted as samples drawn from the true distribution $\Pi_{true} = \Pi_{st}$ over the state variables $\{s_1, s_2, ..., s_N\}$. This fact is employed by the MCMC techniques to sample a target distribution. MCMC methods are a set of methods that can be used to sample distributions which may be difficult to sample due to the lack of availability of an analytical expressions. For instance, let there be a joint distribution on random variables X and Y, $P(X, Y)$. The conditional distribution of X given Y can be found as:

$$P(X|Y) = \frac{P(Y|X) \cdot P(X)}{\int\limits_y P(Y|X) \cdot P(X)} \tag{3.3}$$

It is often difficult to obtain the denominator of an expression such as the one in equation 3.3. Thus the expression for the density of $P(X|Y)$ is known only upto a normalizing constant. Hence, drawing samples directly from this distribution is a challenging task. One of the MCMC techniques that could be used to get samples from this distribution is the *Metropolis Hastings* technique which is discussed in section 3.2.1.

Furthermore, if complete analytical expressions of the conditional distributions $P(X|Y)$ and $P(Y|X)$ are known, an MCMC technique called Gibbs sampling can be used to sample from the joint distribution of random variables X and Y i.e., from $P(X,Y)$. The process works by alternatively sampling the conditional distributions such that each random variable to be sampled is conditioned on the latest value of the other random variable(s). For example, the sample $X_i \sim P(X|Y=y_j)$ is generated in the i-th iteration which is followed by sampling of $Y_k \sim P(Y|X=x_i)$. Hence the latest value of the conditioning variable is used. This gives rise to a sequence of samples for both variables X and Y.

$$X_0, Y_0, X_1, Y_1, \cdots$$

After many sampling steps have been performed, the samples from above sequence start approximating the samples from the joint distribution $P(X,Y)$ as showed in [CG92], [Bes74] and [GS90]. A detailed explanation of working of Gibbs sampling is described in section 3.2.2.

3.2.1 Metropolis Hastings theorem:

The aim is to draw samples from the distribution of equation 3.3. Let the term on the left hand side of this expression i.e., $P(X|Y)$ be referred to as the *target distribution.* Lets define a term $P(Z)$ as:

$$\begin{aligned} & P(Z) \propto P(X|Y) \\ \implies & P(Z) \propto P(Y|X) \cdot P(X) \end{aligned}$$

Note that $p(Z)$ is not yet the true distribution from which it is required to draw samples; it is equivalent to the true distribution from which the normalizing constant is missing. Let the true distribution be denoted by $\pi(Z) = P(X|Y)$. Metropolis Hastings theorem aims to simulate a Markov chain on the random variable Z such that the chain becomes stationary after enough samples have been drawn from it. This means that the chain should be irreducible, aperiodic and thus satisfy the detailed balance equation of 3.2.

The first step in Metropolis Hastings theorem is to select a *proposal distribution* $Q(Z_{t+1} = z'|Z_t = z)$ and simulate draws from it. The proposal distribution gives the probability of selecting $Z = z'$ given that the last draw was $Z = z$. It can be any distribution that is fully known. It is further assumed that it is easy to draw samples from this proposal distribution. Once a sample is drawn from this proposal distribution, it is accepted to be the part of the Markov chain on Z with an acceptance probability $W(Z_{t+1} = z'|Z_t = z)$ i.e., the transition from the state z to z' is accepted with an acceptance probability of $W(Z_{t+1} = z'|Z_t = z)$. If the transition is rejected, Z_{t+1} is set to z. Thus, the transition probabilities are given as:

$$\begin{aligned} a(Z = z'|Z = z) =& W(Z_{t+1} = z'|Z_t = z) \cdot Q(Z_{t+1} = z'|Z_t = z) \\ a(Z = z|Z = z) =& 1 - \sum_{z' \neq z} a(Z = z'|Z = z) \end{aligned}$$

For the Markov chain to be stationary, the detailed balance equation of 3.2 needs to be satisfied:

$$\pi(z) \cdot a(Z = z'|Z = z) = \pi(z') \cdot a(Z = z|Z = z')$$

$$\begin{aligned} \implies \pi(z) \cdot W(Z_{t+1} = z'|Z_t = z) \cdot Q(Z_{t+1} = z'|Z_t = z) = \\ \pi(z') \cdot W(Z_{t+1} = z|Z_t = z') \cdot Q(Z_{t+1} = z|Z_t = z') \end{aligned}$$

Rearranging the terms, we can get the ratio of the acceptance probabilities for transition from z to z' and vice-versa:

$$\frac{W(Z_{t+1} = z'|Z_t = z)}{W(Z_{t+1} = z|Z_t = z')} = \frac{\pi(z') \cdot Q(Z_{t+1} = z|Z_t = z')}{\pi(z) \cdot Q(Z_{t+1} = z'|Z_t = z)}$$

In the above equation, the ratio $\frac{\pi(z')}{\pi(z)}$ can be replaced by the ratio of the known un-normalized densities since the normalizing constant cancels out in the ratio i.e.,

$$\frac{W(Z_{t+1} = z'|Z_t = z)}{W(Z_{t+1} = z|Z_t = z')} = \frac{P(z') \cdot Q(Z_{t+1} = z|Z_t = z')}{P(z) \cdot Q(Z_{t+1} = z'|Z_t = z)}$$

This can be expressed for $W(Z_{t+1} = z'|Z_t = z)$ as:

$$W(Z_{t+1} = z'|Z_t = z) = min\left[1, \frac{p(z') \cdot Q(Z_{t+1} = z|Z_t = z')}{P(z) \cdot Q(Z_{t+1} = z'|Z_t = z)})\right] \tag{3.4}$$

Therefore, the value of Z_{t+1} can be determined as:

$$Z_{t+1} = \begin{cases} z' & \text{with probability } min\left[1, \frac{P(z') \cdot Q(Z_{t+1}=z|Z_t=z')}{P(z) \cdot Q(Z_{t+1}=z'|Z_t=z)})\right] \\ z, & \text{with probability } 1 - min\left[1, \frac{P(z') \cdot Q(Z_{t+1}=z|Z_t=z')}{P(z) \cdot Q(Z_{t+1}=z'|Z_t=z)})\right] \end{cases}$$

3.2.2 Gibbs Sampling:

Gibbs sampling is an MCMC method of sampling distributions if the following conditions are met:

1. Multivariate distribution needs to be sampled.

2. The conditional distribution of one random variable (corresponding to a dimension of the random vector) given all the other random variables (forming other dimensions of the random vector) is available for all the dimensions of the random vector whose distribution needs to be sampled.

Consider a random vector $\underline{Z} = \{Z_1, Z_2, ..., Z_D\}$ having a joint distribution $P(Z_1, Z_2, ..., Z_D)$ which is also the target distribution from which samples are to be drawn. Let Z^t_{-d} denote all the random vector dimensions except Z_d at time instant t. It is further assumed that the conditional distributions of Z_d i.e., $P(Z_d|Z_{-d})$ is known for all D. The Gibbs sampler extends the Metropolis Hastings theorem by setting the proposal distribution Q for each dimension d according to the following criteria:

$$Q(Z_d^{t+1} = z'|Z_d^t = z) = \begin{cases} P(z_d^{'}|z_{-d}) & \text{if } z_{-d}^{'} = z_{-d} \\ 0 & \text{otherwise} \end{cases} \tag{3.5}$$

These assumptions change the expression for acceptance probability. The second term in the *min* function of equation 3.4 can be written as:

$$G = \frac{P(z') \cdot Q(Z_d^{t+1} = z|Z_d^t = z')}{P(z^t) \cdot Q(Z_d^{t+1} = z'|Z_d^t = z)}$$

Using the equation 3.5, the expression for G now becomes:

$$\begin{aligned} G &= \frac{P(z') \cdot P(Z_d^{t+1} = z_d|Z_{-d}^t = z'_{-d})}{P(z^t) \cdot P(Z_d^{t+1} = z'_d|Z_{-d}^t = z_{-d})} \\ &= \frac{P(z'_d|z'_{-d}) \cdot P(z'_{-d}) \cdot P(z_d^t|z'_{-d})}{P(z_d^t|z_{-d}^t) \cdot P(z_{-d}^t) \cdot P(z_d|z_{-d}^t)} \\ &= \frac{P(z'_d|z'_{-d}) \cdot P(z'_{-d}) \cdot P(z_d^t|z'_{-d})}{P(z_d^t|z'_{-d}) \cdot P(z'_{-d}) \cdot P(z_d|z'_{-d})} \\ &= 1 \end{aligned}$$

Therefore, the equation 3.4 for Gibbs sampler becomes:

$$W(Z_{t+1} = z'|Z_t = z) = min[1, G] = 1$$

This means that every new proposal is accepted with probability 1. The algorithm for Gibbs sampling can thus be formulated.

Here, the stationarity property of the Markov chains can be exploited to have a random initialisation of the random variables in the first step of the algorithm since the Markov chain becomes stationary regardless of the initial values of the random variables involved.

3.2.2.1 Example: Gibbs sampling

As an example of Gibbs sampling, a bivariate Gaussian needs to be sampled:

$$(X_1, X_2) \sim \mathcal{N}(x_1, x_2; \boldsymbol{\mu}, \boldsymbol{\Sigma})$$

Algorithm 1 Gibbs sampling algorithm

```
1: Initialize $z^0_{1,2,....,D}$
2: for $t = 0$ to $T - 1$ do
3:     Sample $z_1^{t+1} \sim P(z_1|z_2^t, z_3^t, ...., z_D^t)$
4:     Sample $z_2^{t+1} \sim P(z_2|z_1^{t+1}, z_3^t, ...., z_D^t)$
5:     Sample $z_3^{t+1} \sim P(z_3|z_1^{t+1}, z_2^{t+1}, z_4^t...., z_D^t)$
6:     .
7:     .
8:     Sample $z_D^{t+1} \sim P(z_D|z_1^{t+1}, z_2^{t+1}, z_3^{t+1}...., z_{D-1}^{t+1})$
9: end for
```

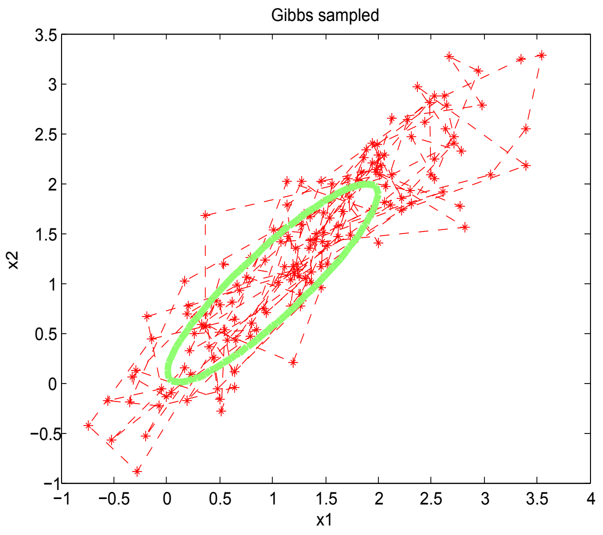


Figure 3.1: Bivariate Normal Gibbs Sampling

where $\boldsymbol{\mu} = [\mu_1, \mu_2]$ and $\boldsymbol{\Sigma} = \begin{bmatrix} \sigma_{11}^2 & \sigma_{12}^2 \\ \sigma_{21}^2 & \sigma_{22}^2 \end{bmatrix}$ represent the mean and covariance matrix of the bivariate density. The conditional distributions of x_1 and x_2 are given as:

$$P(x_i|x_j; \mu', \Sigma') = \mathcal{N}(x_i; \tilde{\mu_i}, \tilde{\sigma_i})$$

$$\tilde{\mu_i} = \mu_i + \frac{\sigma_{i,j}^2}{\sigma_{j,j}^2}(x_j - x_i)$$

$$\tilde{\sigma_i^2} = \sigma_{i,i}^2 - \frac{\sigma_{i,j}^2}{\sigma_{j,j}^2}$$

The samples generate from this bivariate Gaussian distribution are shown in figure 3.1. The figure shows the path the samples take as they are generated from conditional distributions. The green curve at the center shows the first standard deviation of the original Gaussian distribution.

3.2.2.2 Application of Gibbs sampling: The reverse problem

Gibbs sampling can also be applied in the opposite way i.e., given the samples from a distribution, Gibbs sampling can be used to sample the parameters of this distribution. A scheme for selecting the final values of the parameters can be chosen in a variety of ways. For example, all the samples of the parameters are averaged to get the final estimate of the parameter or the final sample obtained can be considered the estimate of the parameter.

Consider the reverse problem where samples (X) from a univariate Gaussian are provided and it is required to obtain the mean (μ) and precision λ (inverse of variance) of the Gaussian i.e., $X \sim \mathcal{N}(x; \mu, \lambda^{-1})$. The problem can be formulated by assuming a joint distribution on the parameters to be determined i.e., on μ and λ:

$$P(\mu, \lambda) \sim NG(\mu_0, \kappa_0, \alpha_0, \beta_0)$$

Here, a Normal-Gamma distribution is assumed over the mean and precision. $\mu_0, \kappa_0, \alpha_0, \beta_0$ are the parameters of this Normal-Gamma distribution and are assumed to be known. The next step in the sampling process is to obtain the conditional probabilities of μ and λ which can be obtained as:

$$P(\mu|\lambda, X) = \mathcal{N}(\mu; \frac{\kappa_0\mu_0 + n\bar{x}}{\kappa_0 + n}, (\kappa_0 + n)\lambda^{-1}) \tag{3.6}$$

$$P(\lambda|X) = Ga(\lambda; \alpha_0 + \frac{n}{2}, \beta_0 + \frac{1}{2}\sum_{i=1}^{n}(x_i - \bar{x})^2 + \frac{\kappa_0 n(\bar{x} - \mu_0)^2}{2(\kappa_0 + n)}) \tag{3.7}$$

where n denotes the total number of samples of X. Note that in the expression for λ, the dependence on μ has not been considered. Such choices are design choices and Gibbs sampling still works in this case. An experiment was conducted in which artificial data was generated from a univariate Gaussian density with a known mean and precision. This data was then fed as input samples for estimating the mean and precision for this density. The distributions were assumed on the mean and precision as given in equation 3.6 with $\kappa_0 = 1, \alpha_0 = 1, \beta_0 = 1$ and μ_0 initialised to a random value. It was found that after running the algorithm for a sufficiently long time, the distributions obtained for μ and λ were indeed close to the true distributions of these parameters if sufficiently large number of observation samples were available. This shows that Gibbs sampling also works in practice for the sampling of the parameters of a distribution given the samples from the distribution.

3.3 Bayesian Learning

Bayesian learning techniques are often used in statistics to give better results.For instance, in the linear regression technique, Bayesian updates improve the weight estimates to a considerable degree [Bis06]. Bayesian learning can also be applied in the conjunction with Gibbs sampling.

In the last section, a *prior* density was assumed on the random variables μ and λ with an initial known parameter set. To obtain the posterior distribution, this prior density was used with the relation:

$$Posterior \propto Likelihood \times Prior \tag{3.8}$$

The posterior here is the required conditional density. For estimating the mean and precision of a Gaussian as in the earlier example, the relation in equation 3.8 can be written as:

$$\begin{aligned}
&P(\mu, \lambda|X) \propto P(X|\mu, \lambda) \times P(\mu, \lambda) \\
\Longrightarrow\ &P(\mu, \lambda|X) \propto NG(\mu, \lambda; \mu', \kappa', \alpha', \beta') \\
&\text{with,} \\
&\mu' = \kappa_0\mu_0 + n\bar{x} \\
&\kappa' = \kappa_0 + n \\
&\alpha' = \alpha_0 + \frac{n}{2} \\
&\beta' = \beta_0 + \frac{1}{2}\sum_{i=1}^{n}(x_i - \bar{x})^2 + \frac{\kappa_0 n(\bar{x} - \mu_0)^2}{2(\kappa_0 + n)}
\end{aligned}$$

Here, the likelihood function $P(X|\mu, \lambda)$ is a Gaussian. Note that the posterior obtained in the above equation is also a normal-gamma distribution with updated parameters. Thus, the shape of the posterior distribution is same as that of the prior distribution with updated parameters. Such prior distributions for which the posterior distributions have the same shape are called *conjugate prior distributions.* For a given likelihood distribution, there are specific prior distributions that yield a posterior distribution having the same shape as the prior. A different prior distribution for the Gaussian likelihood may not yield a normal-gamma posterior in the above example. A proof that normal-gamma distribution serves as a conjugate prior for a Normal distribution is shown in the appendix section of this report. Conjugate priors simplify the task of obtaining the posterior distribution as it has the same shape as the prior distribution. Hence, conjugate priors are heavily employed throughout this thesis.

Model

In this chapter, the specific model and techniques used to formulate and solve the problem will be discussed.

As explained in the previous chapters, the problem can be subdivided into 3 tasks: segmentation, clustering and learning. The task of segmentation is dealt by using the Viterbi algorithm. Dirichlet process, along with Gibbs sampling, is used to cluster similar segments together, each of which correspond to a phoneme where each phoneme is modelled using an HMM. The parameters of each of the HMMs are then learnt using Gibbs sampling. Section 4.1 describes the problem statement in detail and presents an outline of the algorithm that has been used to solve it. The system can be modelled as a generative process that outputs the observation samples. Section 4.2 describes this generative process. Section 4.3 discusses each of the steps of the algorithm in detail.

4.1 Problem description

Figure 4.1 shows how segmentation, clustering and learning is applied onto an utterance of TIMIT database. Figure 4.2 shows the complete algorithm.

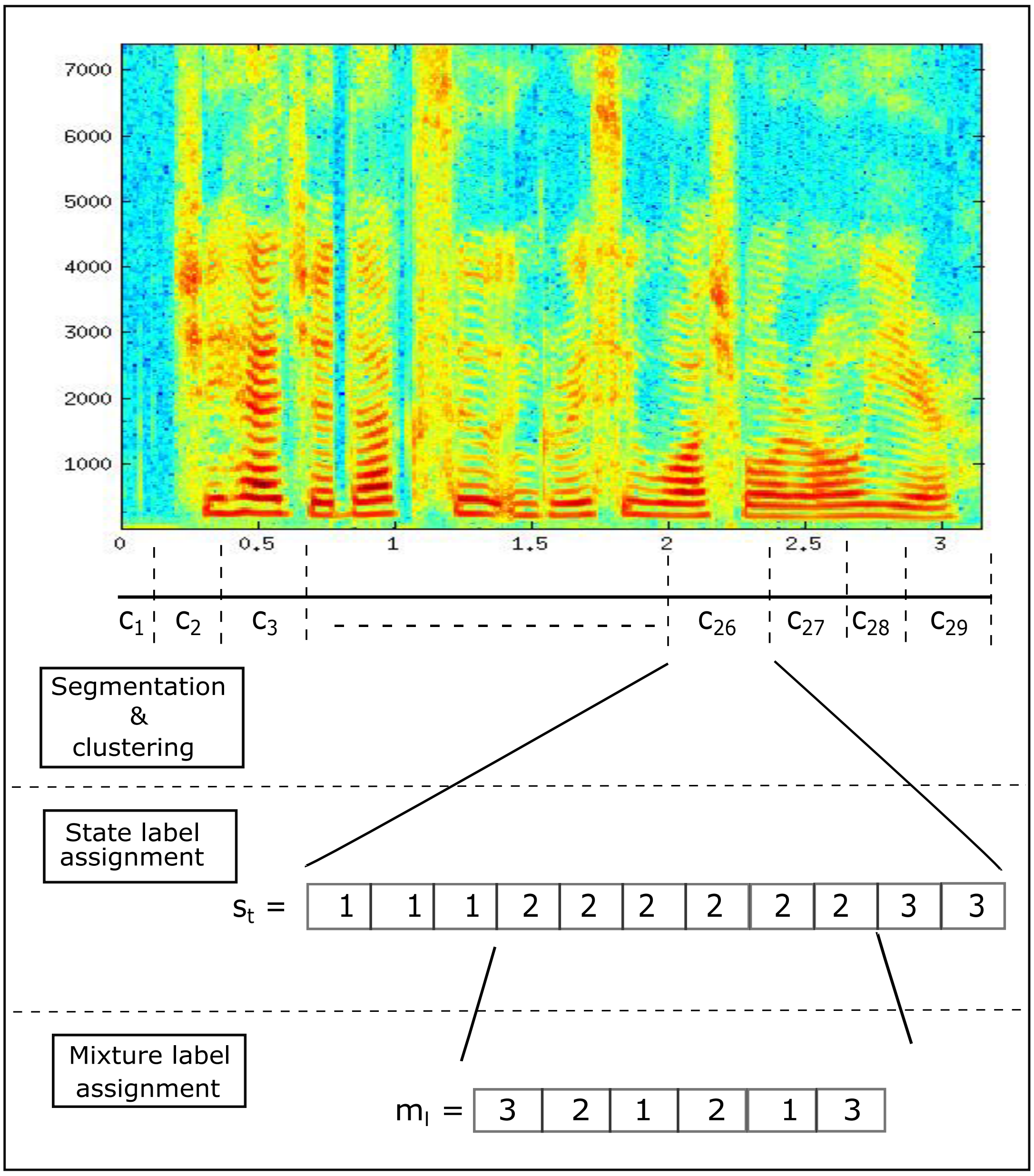


Figure 4.1: Depiction of subtasks: segmentation, clustering and learning for a TIMIT utterance. c_i represents the cluster label of i-th sequence, s_t is the state assigned to the t-th sample within a sequence and m_l is the mixture label of the l-th sample for a given state.

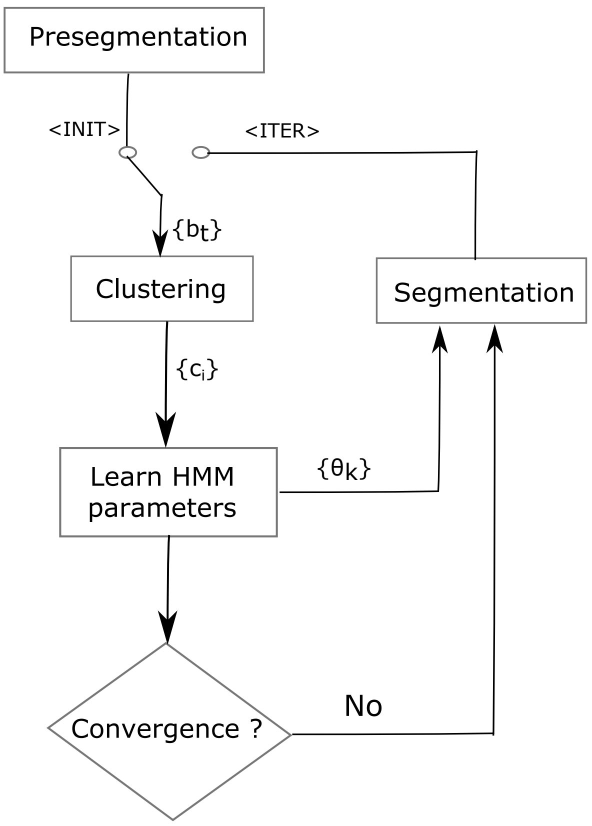


Figure 4.2: Presegmentation algorithm:<INIT> & <ITER> represent initialization and iterations respectively, based on which the clustering block receives input from either the Presegmentation block or the segmentation block.

4.1.1 Segmentation

The first task is to segment the continuous stream of speech samples into sequences of speech samples so that each sequence can be seen as originating from a phoneme. In the figure 4.1, a segmented utterance is shown. However, in practice, the boundary of speech sequences are not known. Until this boundary is known, it is not possible to differentiate between phonemes within a speech stream.

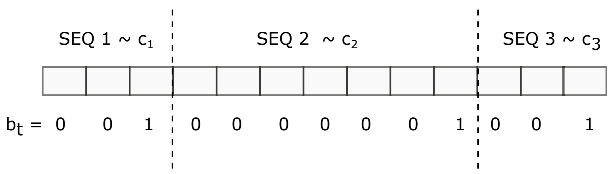


Figure 4.3: Segmentation of continuous speech samples.

As can be seen from figure 4.2, at the first iteration of the algorithm, a presegmentation step is carried out to initialise a boundary variable b_t that determines how the continuous speech stream is divided into sequences of phonemes (this is explained in section 4.3.1). After the first iteration, a segmentation algorithm is used to determine the boundaries. Figure 4.3 shows the boundary variable (b_t) that determines the segments. The sample for which b_t is 1 is the last sample in this particular segment. In the figure 4.1, 29 sequences were determined. This implies that the value of b_t was 1 for 29 of the samples in the utterance.

4.1.2 Clustering

The second task in the algorithm is the task of clustering as can be seen from figure 4.2. The output of this step is a set of cluster labels. For each sequence i, a cluster label c_i is determined which maps the sequence to an HMM. Thus, c_i takes value as:

$$c_i \in \{1, 2, \cdots, k\}$$

where k denotes the number of HMMs known upto the current iteration of the algorithm. Figure 4.3 shows that there is a cluster label for each of the sequences. Multiple sequences may get mapped to one HMM. Also, a sequence can not get mapped to more than one HMM.

The structure of the HMM can be seen from the figure 4.4. The HMM used to model the phonemes has 3 states. It is also a left-right HMM. This means that while it is possible to move from a lower numbered state to a higher numbered state, the reverse is not possible. For example, the system can move from state-1 to state-2 but once in state-2, it can not move back to state-1. The implications of the left-right assumption along with other assumptions can be summarised as:

1. $\forall i > j, a_{ij} = 0$, where a_{ij} denotes the probability of transitioning from state i to state j.
2. First sample in each sequence gets assigned to state-1.
3. Last sample in each sequence gets assigned to state-3.
4. A jump from state-1 to state-3 is not allowed i.e., to reach state-3 of HMM, the system must go through state-2 as well. This implies that the minimum sequence length to be considered is three.

The second condition implies that the set of initial state probabilities becomes:

$$\Pi = \{\pi_1, \pi_2, \pi_3\} = \{1, 0, 0\}$$

4.1.3 Learning

The third and the final task in the algorithm is learning of the parameters of each of the HMMs as shown in figure 4.2.

Once all the sequences corresponding to an HMM are obtained from the clustering step, the next step is to assign to each sample in each sequence a hidden state from which the sample may have originated. A state variable s_t is used for this purpose. The *state variable* s_t denotes the state of the HMM k from which the t-th sample may have originated. Since the HMM has 3 states, s_t takes value as:

$$s_t \in \{1, 2, 3\}$$

Figure 4.4 shows the state assignment for the samples of sequence 26 of the TIMIT utterance shown in figure 4.1 which has a cluster label k.

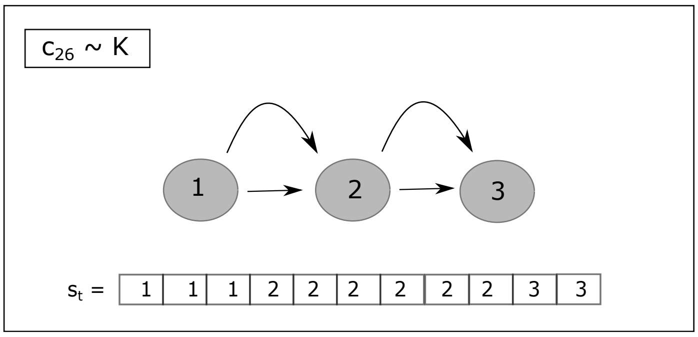


Figure 4.4: Hidden state assignment for the sequence number 26 of TIMIT utterance shown in figure 4.1.

Once the hidden state assignment has been done for all sequences in an HMM, the transition probabilities $\{a_{ij}\}$ of the HMM can be learnt.

The observation density corresponding to each state can be considered to be a mixture density such as a Gaussian mixture model (GMM) density. Thus, all samples belonging to a particular hidden state (as determined by the value of s_t) are assumed to originate from a GMM representing the observation density of that particular state of an HMM. Each such sample is then marked for the GMM mixture component that it belongs to. This is accomplished by using a variable m_l. The variable m_l takes value as:

$$m_l \in \{1, 2, \cdots, N_m\}$$

where N_m denotes the number of mixture components of the GMM. An example of mixture label assignment is shown in the figure 4.1 for all samples belonging to hidden state-2 of sequence number 26 of the TIMIT utterance shown in figure 4.1 with $N_m = 3$. Figure 4.5 magnifies this view to show all the samples.

The state-2 has six samples assigned to it. These get assigned the mixture components $\{3, 2, 1, 2, 1, 3\}$ respectively. All the samples assigned to a particular mixture component are then gathered to further tune and learn the parameters of the corresponding GMM mixture.

The complete set of HMM parameters can now be given by:

$$\theta = \big\{\{a_{ij}\}, \{\epsilon\}, \{\mu_m\}, \{\lambda_m\}\big\}$$

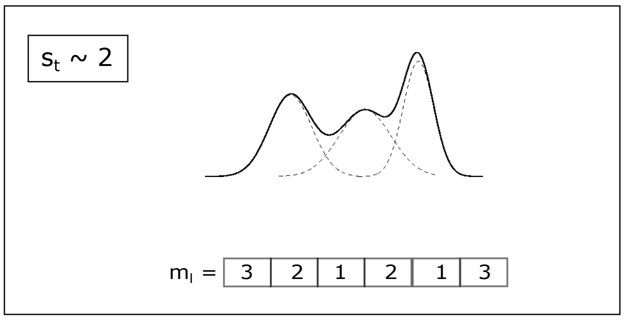


Figure 4.5: Assignment of mixture labels to each sample for all samples belonging to state-2 of figure 4.1.

where $\{a_{ij}\}$ represent the set of transition probabilities for an HMM, $\{\epsilon\}$ is the set of mixture probabilities for GMMs across all states in a given HMM with $\{\mu_m\}$ and $\{\lambda_m\}$ as the corresponding mixture means and precisions.

The algorithm can then proceed to the next iteration or terminate if the convergence has been reached as shown in figure 4.2.

4.2 Generative Process

The system model uses a generative process that aims to generate the observed utterances. This section deals with this generative process and marks a distinction between the system variables that are random in nature and those that are deterministic in nature.

Figure 4.6 shows the graphical model corresponding to the generative process. A shaded circle represents the observed random variable i.e., the observations $(\mathbf{X}_t)$. The shaded rectangles denote deterministic quantities. These deterministic quantities are the hyper parameters such as α_0, β_0, etc. and the boundary variable b_t. The unshaded circles represent latent random variables such as cluster labels c_i. The dashed arrows denote deterministic relations. For example, the segment duration d_i is completely determined once b_t is known. Boundary variables are treated as deterministic variables. Section 4.3.1 discusses the process of obtaining the values of b_t. Therefore, the generative process starts with the inference of the cluster labels which are the latent random variables. The generative process can now be stated as:

- Obtain cluster labels c for each sequence. This is done using Gibbs sampling and will be explained in section 4.3.2.
- Given a cluster label, assign values to hidden state variables s_t for each sample in the sequence. This is achieved by using block Gibbs sampling as will be discussed in section 4.3.3.
- Given the state s_t, select a mixture component of the GMM corresponding to s_t.

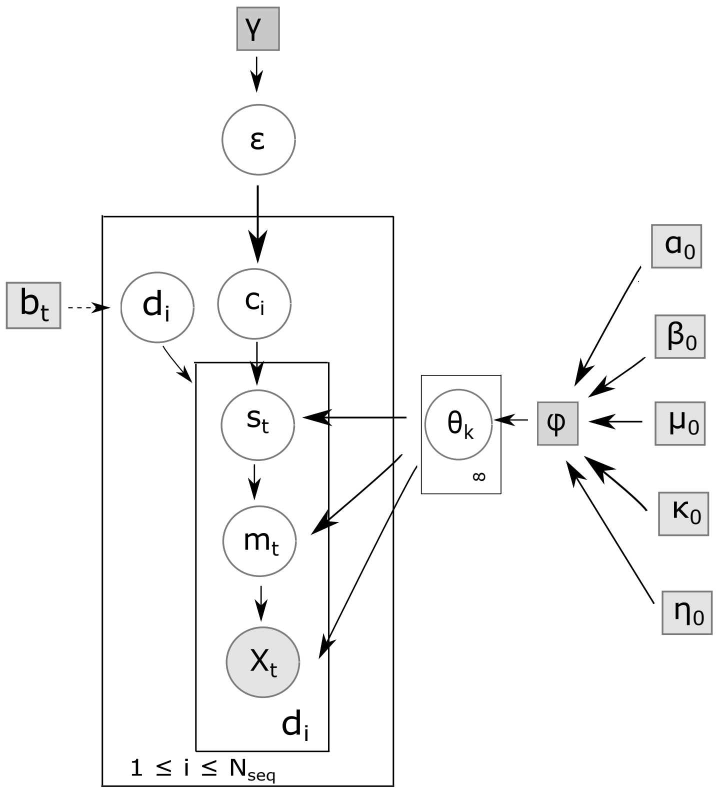


Figure 4.6: Graphical Model of system

This is accomplished using Gibbs sampling and is explained in section 4.3.4.2.

- Use the chosen mixture component to generate the sample X_t.

4.3 Modelling choices and sampling equations

This section deals with the particular modelling choices made and describes various techniques used to obtain the variables listed in the previous sections.

4.3.1 Boundary Variable b_t

The boundary variable b_t determines the boundary of segments in a continuous stream of speech samples. It is a deterministic variable and takes value according to the following rule:

$$b_t = \begin{cases} 1, & \text{if sample is the last in a sequence} \\ 0, & \text{otherwise} \end{cases}$$

A segment between times j and k is thus defined as a set of consecutive samples for which $b_{j-1} = 1$ and $b_k = 1$. For $j = 1$ i.e., for the very first sample, $b_1 = 0$. This can be verified from figure 4.3. Boundary variable is first initialised using a presegmentation algorithm. Afterwards, Viterbi algorithm is used to determine its values during regular iterations.

4.3.1.1 Initialization: Presegmentation

At the first iteration of the algorithm, the method suggested in [AEEM01] is used to determine the values of b_t. The method tries to detect the times at which there is maximum change in the signal energy. This is done for each feature vector dimension separately and then the results from each dimension are combined to produce the most significant points of change which are referred to as 'jumps'. Jumps are detected by thresholding the following function called as jump function:

$$J_d^a[t] = \left| \sum_{m=t-a}^{t-1} \frac{x_d[m]}{a} - \sum_{m=t+1}^{t+a} \frac{x_d[m]}{a} \right|$$

The jump function evaluates the difference of means of $'a'$ samples that precede and succeed the current sample. d represents the dimension of the feature vector. For a given dimension d, a set of candidate peaks $\{t'\}$ are selected from the jump function if there exists integers l and m such that:

$$\begin{aligned} J_d^a[t'] > J_d^a[t'-1] > \cdots > J_d^a[l] \leq J_d^a[l-1] \\ J_d^a[t'] \geq J_d^a[t'+1] \geq \cdots \geq J_d^a[m] < J_d^a[m+1] \end{aligned}$$

The integer l is the number of samples preceding the current sample in the jump function, upto which there is a consecutive decrease in the amplitudes of these preceding samples. Thus, $J_d^a[l] \leq J_d^a[l-1]$ i.e., the amplitude of l-th sample is not bigger as compared to the $(l-1)$-th sample. Similarly, the integer m represents the number of samples succeeding the current sample such that the amplitude of the successive samples decrease. This implies $J_d^a[m] < J_d^a[m+1]$ which means that the amplitude of $(m+1)$-th is bigger than that of the m-th sample. After obtaining the values of l and m for each sample in the jump function, the height of the peak is found as:

$$h = min(h_1, h_2)$$

where $h_1 = (J_d^a[t'] - J_d^a[l])$ and $h_2 = (J_d^a[t'] - J_d^a[m])$.

A threshold value b is then used to retain peaks with $h > b$. A $\{D \times T\}$ matrix $\mathbf{S}$ is now defined with values $S(d,t) = 1$ if a valid peak is present and $S(d,t) = 0$ otherwise. Using $\mathbf{S}$, it is possible to combine the peaks across all dimensions of the feature vector in order to infer the boundary variable values. A function $f[j]$ is defined as:

$$f[j] = \sum_{k=p}^{q} \sum_{d=1}^{D} S(k,d)|j-k|, \quad \forall j \in [p,q] \tag{4.1}$$

p and q denote a sliding window of length $(c+1)$ within which the function $f[j]$ is evaluated. A set of functions $f[j]$ is computed for all points in interval $F = [a, T-a]$. Within the interval $[p, q]$, the equation 4.1 tries to determine the distance of each point from a prospective boundary. The point within $[p, q]$ for which this distance is minimum is most likely to be the boundary. This minimum distance point j' is found such that $f[j'] = min(f[j])$. A set of all such boundary points is obtained over the entire sample interval $F = [a, T-a]$. The points which are selected at least once are then marked as candidate boundary points. A threshold can be set to retain only those candidate points as boundary points which have been selected a certain number of times throughout the interval $[a, T-a]$. This procedure of evaluation of $f[j]$ can be explained using figure 4.7.

For the figure, it is assumed that $a = 2, c = 2$ and the length of the sequence $T = 8$. For $t = a = 2$, values of p and q are obtained as:

$$\begin{aligned} & p = t \\ \implies & p = 2 \\ & q = t + c \\ \implies & q = 4 \end{aligned}$$

Using the equation 4.1, the value of $f[j]$ is evaluated at $j = 2, 3, 4$ since j needs to be evaluated for all values in the interval $[p, q]$. j' can now be found using $f[j'] = min(f[j])$ for $t = a$. An accumulator vector 'acc' keeps the count of each of the peaks found using j' by incrementing the value $acc[j']$ each time j' is selected as the minimum distance point. The bottom part of the figure 4.7 depicts the values of p and q for $t = T - a - c$. Hence $f[j]$ gets computed for $j = 4, 5, 6$. A procedure FINDMIN(S,p,q) can now be defined to find j' for a given (S, t, c).

Algorithm 2 Find minimum distance: Presegmentation

```
1: procedure FINDMIN($S, t, t+c$)
2:     set $p = t$, $q = t + c$
3:     for $j \to p$ to q do
4:         determine $f[j]$ using eq. 4.1
5:     end for
6:     find $j$' s.t. $f[j'] = min(f[j])$
7:     return $j'$
8: end procedure
```

At the end, peaks can be detected in the accumulator function. For each peak, the boundary variable $b_t = 1$. For other points, $b_t = 0$. The complete algorithm for presegmentation can now be formulated.

An empirical measure of the quality of segmentation can be obtained in a supervised way using recall:

$$recall = \frac{\text{number of correctly detected segmentation points}}{\text{number of true segmentation points}}$$

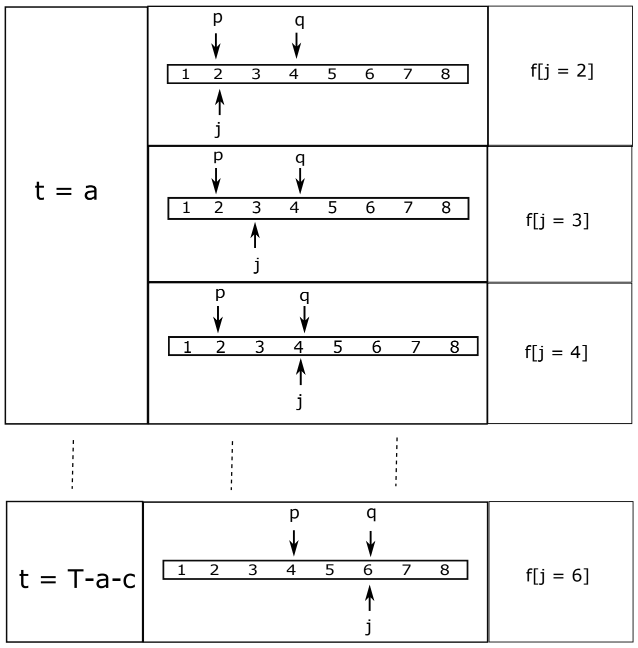


Figure 4.7: Presegmentation procedure: $T = 8, a = 2, c = 2$

Algorithm 3 Presegmentation algorithm

```
 1: Initialize $acc = [\mathbf{0}]_{1\times T}$, $\mathbf{S} = [\mathbf{0}]_{D\times T}$
 2: for $d \to 1$ to $D$ do
 3:     for $t \to a$ to $T - a$ do
 4:         determine $J_d^a[t]$
 5:     end for
 6:     detect peak using $J_d^a[t]$ and threshold $b$
 7:     set $S(t,d) = 1$ if a peak is detected
 8: end for
 9: for $t \to a$ to $T - a - c$ do
10:     $j' =$ FINDMIN(S,t,t+c)
11:     $acc[j'] = acc[j'] + 1$
12: end for
13: find subset $G \subset \{1, 2, .., T\}$ which contains peaks in acc and subset $\bar{G} \subset \{1, 2, .., T\}$
    with no peaks
14: set $b_t = 1$ for $t \in$ G and set $b_t = 0$ for $t \in \bar{G}$
```

A boundary is said to be correctly detected if it lies at the same point as the original boundary with a tolerance of ± 2 samples.

A measure of the amount of segmentation can be obtained by evaluating the percentage of over segmentation:

$$Oversegmentation(O_s) = \frac{\text{Number of segmentation points obtained}}{\text{Total number of true segmentation points}} \times 100$$

The value of O_s is more than 100% if number of detected segments are more than the true count of the segments.

4.3.1.2 Iterations: Segmentation

After the first iteration, few HMMs are obtained (as explained in later sections). Hence, it is possible to use this information to apply Viterbi algorithm to determine the values of b_t. This is achieved by cascading the learnt HMMs. An example is shown in figure 4.8 where it is assumed that after the first iteration of the algorithm, two HMMs are learnt. The cascaded HMM in figure 4.8 shows two HMMs, one in green, the other in blue with their transitions marked in the corresponding ink. There are also transitions marked with black ink and depict *inter HMM transitions.* Hence a transition is possible from HMM1 to HMM2 and vice-versa. Also, there is a possibility of system entering the same HMM. These transitions are shown as transitions from state-3 to state-1 which otherwise are not possible within an HMM because of the left-right assumption. The state space of the cascaded system of figure 4.8 can be defined as:

$$\Omega_{cas} = \{1, 2, 3, 4, 5, 6\}$$

Here, the states $\{4, 5, 6\}$ are mapped to states $\{1, 2, 3\}$ of HMM-2. A cascaded transition matrix is then made:

$$\mathbf{A}_{cas} = \begin{bmatrix} a_{11} & a_{12} & 0 & 0 & 0 & 0 \\ 0 & a_{22} & a_{23} & 0 & 0 & 0 \\ \epsilon_1 \cdot (1 - z_1) & 0 & z_1 & \epsilon_2 \cdot (1 - z_1) & 0 & 0 \\ 0 & 0 & 0 & a_{44} & a_{45} & 0 \\ 0 & 0 & 0 & 0 & a_{55} & a_{56} \\ \epsilon_1 \cdot (1 - z_2) & 0 & 0 & \epsilon_2 \cdot (1 - z_2) & 0 & z_2 \end{bmatrix}$$

with,

- $z_1 = \frac{\text{No. of transitions from state-3 to state-3 in HMM-1}}{\text{No. of transitions from state-3 in HMM-1 + 1}}$
- $z_2 = \frac{\text{No. of transitions from state-3 to state-3 in HMM-2}}{\text{No. of transitions from state-3 in HMM-2 + 1}}$
- ϵ_1 - probability of being in HMM-1.
- ϵ_2 - probability of being in HMM-2.
- $\sum_{i=1}^{K} \epsilon_i = 1$

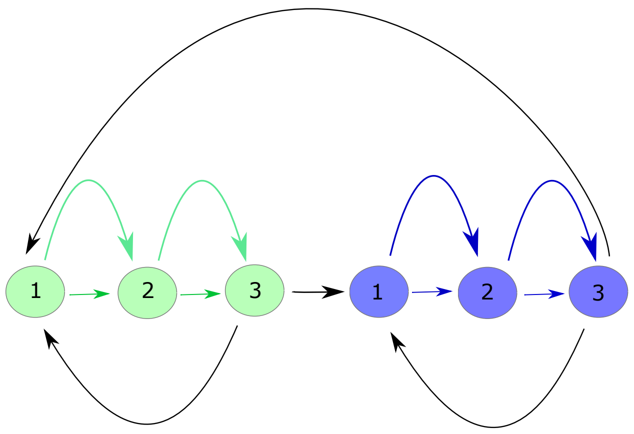


Figure 4.8: HMM cascading for determination of b_t.The green and blue arrows represent the inter-state transitions for HMM-1 and HMM-2 respectively.The black arrows represent inter-HMM transitions.

The additional 1 transition counted in the denominator for computing variables z_1 and z_2 accounts for one transition out of the respective HMM.

An initial state probability is also assigned for the cascaded system. For the example shown in figure 4.8, the cascaded initial probability vector is:

$$\begin{aligned}\underline{\Pi}_{cas} &= \{\pi_1, \pi_2, \pi_3, \pi_4, \pi_5, \pi_6\} \\ &= \{\epsilon_1, 0, 0, \epsilon_2, 0, 0\}\end{aligned}$$

Using the cascaded transition matrix and the cascaded initial probabilities ,Viterbi algorithm as described in 2.3.2.2 is now applied to the speech samples to obtain the most likely sequence of states. Hence, each sample gets assigned to a state from the cascaded state space Ω_{cas}. Now the boundaries can be detected as follows:

$$b_t = \begin{cases} 1, & \text{if } mod(s_t, N) = 0 \text{ and } mod(s_{t+1}, N) = 1 \\ 0, & \text{otherwise} \end{cases}$$

s_t here refers to the state at time t for the cascaded HMM system. *mod* denotes the modulus operation. N is the number of states per HMM which is 3 in our case. The boundary variable for $t = T$ is 1, where T is the total number of samples. The segmentation algorithm can be stated as:

Algorithm 4 Segmentation algorithm

```
 1: define cascaded state space $\Omega_{cas}$
 2: construct $\mathbf{A}_{cas}, \underline{\Pi}_{cas}$
 3: obtain state assignments $\{s_t\}$ for all samples using Viterbi algorithm
 4: for $t \rightarrow 1$ to $T$ do
 5:     if $mod(s_t, N) = 0$ and $mod(s_t + 1, N) = 1$ then
 6:         set $b_t = 1$
 7:     else
 8:         set $b_t = 0$
 9:     end if
10: end for
```

4.3.2 Cluster Label c_i

Once the segments have been obtained, the next task is to determine the cluster label c of the each of these segments. This is done using Dirichlet process and Gibbs sampling. Each unique cluster label corresponds to a unique HMM. Let c_{-i} represent the set of all cluster labels except that of sequence $\mathbf{X}_i$ whose cluster label c_i needs to be obtained. To use Gibbs sampling, the conditional distributions of a variable given all other variables in the system must be known. The conditional posterior probability of c_i being equal to k, given all the parameters in the system can be obtained using Baye's rule:

$$P(c_i = k|c_{-i}, \mathbf{X}_i, \theta_k, \cdots) \propto P(c_i = k|c_{-i}) \cdot P(\mathbf{X}_i|\theta_k) \tag{4.2}$$

where:

- θ_k represents the parameters of the k-th HMM.
- $\mathbf{X}_i$ represents the sequence whose label needs to be obtained.
- The dots in the conditional posterior term represent all the observed data, system variables and hyperparameters for the model other than the current observation sequence($\mathbf{X}_i$), θ_k and c_{-i}.

The expression in equation 4.2 constitutes a product of conditional prior (first term in equation 4.2) and a likelihood (second term in equation 4.2). A discussion on each of these terms follows next.

4.3.2.1 Conditional prior term

$P(c_i = k|c_{-i})$ is the conditional prior which is the probability of the cluster label of sequence $\mathbf{X}_i$ to be equal to value k given the cluster labels of all other sequences. The expression for this term varies depending upon whether we know the total number of HMMs the system contains i.e., whether the total number of phonemes for a language is known. Thus, the derivation for the conditional prior term is divided into two cases: Known HMM count & Unknown HMM count.

<u>Known number of HMMs</u>:
Initially, a Dirichlet distribution with a hyperparameter set $\underline{\gamma}$ is assumed on the initial probability of occurrence of cluster label classes.

$$\begin{aligned} P(c_i = k) &= \epsilon_k \\ \underline{\epsilon} &\sim Dir(\underline{\gamma}) \\ P(\epsilon_1, \epsilon_2, ..., \epsilon_K | \underline{\gamma}) &= Dir(\epsilon_1, \epsilon_2, ..., \epsilon_K; \underline{\gamma}) \end{aligned} \tag{4.3}$$

Here, ϵ_k denotes the initial probabilities of occurrence of the label class k. A symmetric Dirichlet distribution can be utilised to simplify the derivation by defining $\underline{\gamma} = \{\gamma/K, \gamma/K, \cdots \gamma/K\}$. Hence the left hand side term in equation 4.3 can be written as:

$$\begin{aligned} P(\epsilon_1, \epsilon_2, ..., \epsilon_K | \underline{\gamma}) &= Dir(\epsilon_1, \epsilon_2, ..., \epsilon_K; \gamma/K, \gamma/K, \cdots \gamma/K) \\ \implies P(\epsilon_1, \epsilon_2, ..., \epsilon_K | \underline{\gamma}) &= \frac{\Gamma(\gamma)}{\Gamma(\gamma/K)^K} \prod_{j=1}^{K} \epsilon_j^{\gamma/K-1} \end{aligned} \tag{4.4}$$

The probability of occurrence of cluster labels for a sequence is:

$$P(c_1, c_2, ..., c_{N_{seq}} | \epsilon_1, \epsilon_2, ..., \epsilon_K) = \prod_{j=1}^{K} \epsilon^{N_j} \tag{4.5}$$

where N_j denotes the number of sequences assigned to each label class. Now, it is possible to directly model the dependence of c_i on γ by integrating out the $\underline{\epsilon}$. Using eq. 4.4 and 4.5:

$$\begin{aligned} P(c_1, c_2, ..., c_{N_{seq}} | \underline{\gamma}) &= \int ... \int P(c_1, c_2, ..., c_{N_{seq}} | \underline{\epsilon}) P(\epsilon_1, \epsilon_2, ..., \epsilon_K | \underline{\gamma}) d\epsilon_1 ... d\epsilon_K \\ &= \frac{\Gamma(\gamma)}{\Gamma(\gamma/K)^K} \int \cdots \int \prod_{j=1}^{K} \epsilon_j^{N_j + \gamma/K - 1} d\epsilon_1 d\epsilon_2 \cdots d\epsilon_K \\ &= \frac{\Gamma(\gamma)}{\Gamma(T + \gamma)^K} \prod_{j=1}^{K} \frac{\Gamma(N_j + \gamma/K)}{\Gamma(\gamma/K)} \end{aligned}$$

Having obtained the joint distribution of cluster labels given the Dirichlet distribution parameter, the conditional distribution of a cluster label given all other sequence cluster labels and the Dirichlet parameter can be evaluated as:

$$P(c_i = k | c_1, c_2, \cdots, c_{i-1}, c_{i+1}, \cdots, c_{N_{seq}}; \gamma) = \frac{N_{-k} + \gamma/K}{N_{seq} - 1 + \gamma} \tag{4.6}$$

$$\implies P(c_i = k | c_{-i}) = \frac{N_{-k} + \gamma/K}{N_{seq} - 1 + \gamma} \tag{4.7}$$

where, N_{-k} is the number of sequences with label k excluding the current sequence.
<u>Unknown number of HMMs</u>:
In case the total number of HMMs are unknown, infinite number of HMMs may be

obtained since there is no assumption on the upper limit of the number of phonemes a language could accommodate. Corresponding to this case, equation 4.6 is modified by taking a limit $K \to \infty$. For the HMMs having $N_{-k} \neq 0$, i.e., for the HMMs that have been discovered until a particular iteration of the algorithm, equation 4.6 becomes:

$$\begin{aligned} P(c_i = k | c_1, c_2, \cdots, c_{i-1}, c_{i+1}, \cdots, c_{N_{seq}}; \gamma) &= \lim_{k \to \infty} \frac{N_{-k} + \gamma/K}{N_{seq} - 1 + \gamma} \\ &= \frac{N_{-k}}{N_{seq} - 1 + \gamma} \end{aligned} \tag{4.8}$$

For each sequence however, there is a possibility of the sequence originating from a new HMM (k_{new}) since the total number of HMMs are unknown. Therefore, for each sequence, in addition to evaluating the probability of the sequence to belong to existing HMMs, the probability of the sequence to belong to a new HMM is also calculated. This process is referred to as *Dirichlet process*. The expression in equation 4.8 for the new HMM becomes:

$$P(c_i = k_{new} | c_1, c_2, \cdots, c_{i-1}, c_{i+1}, \cdots, c_{N_{seq}}; \gamma) = \frac{\gamma}{N_{seq} - 1 + \gamma} \tag{4.9}$$

4.3.2.2 Likelihood term

The term $P(\mathbf{X}_i|\theta_k)$ in equation 4.2 is the likelihood term for determining the likelihood of the sequence X_i originating from HMM k having attributes θ_k. The discussion for this likelihood term is also divided into two cases; when number of HMMs are known and when the number of HMMs are unknown.
Known number of HMMs:
For the known HMMs, the term is obtained by using the forward variable α of the Forward-Backward algorithm discussed in section 2.3.1. Likelihood of a sequence of length T originating from HMM k can then be found as:

$$P(\mathbf{X}_i|\theta_k) = \sum_{j=1}^{N} \alpha_T(j) \tag{4.10}$$

where N is the number of states per HMM, which has been chosen to be 3 for our case.
Unknown number of HMMs:
If the total number of HMMs are not known in advance, in addition to the HMMs discovered up till the current iteration of the algorithm, the likelihood of the given sequence to belong to a new HMM also needs to be considered. This is found as:

$$P(\mathbf{X}_i|c_i = k_{new}) = \int P(\mathbf{X}_i|c_i = k_{new}, \phi) P(\phi) d\phi$$

The integral in the above equation is handled by using suggestions given in [Nea12]. A new HMM with parameters θ_{new} is sampled from a base distribution (ϕ) which is also depicted in the generative model of figure 4.6. The base distribution is determined by the set of all system hyperparameters and each new HMM can be considered to be an instance of this base distribution. Thus, an HMM from the base distribution ϕ is sampled and then the likelihood is computed as shown in equation 4.10.

4.3.2.3 Conditional Posterior term

Finally, the expression for the term $P(c_i = k|c_{-i}, \mathbf{X}_i, \theta_k, \cdots)$ in equation 4.2 depends upon whether the total number of HMMs are known in advance. Considering these two cases:
<u>Known number of HMMs</u>:

$$P(c_i = k|c_{-i}, \mathbf{X}_i, \theta_k, \cdots) \propto \frac{N_{-k} + \gamma/K}{N_{seq} - 1 + \gamma} \cdot P(\mathbf{X}_i|\theta_k) \tag{4.11}$$

<u>Unknown number of HMMs</u>:

$$P(c_i = k|c_{-i}, \mathbf{X}_i, \theta_k, \cdots) \propto \frac{N_{-k}}{N_{seq} - 1 + \gamma} \cdot P(\mathbf{X}_i|\theta_k) \tag{4.12}$$

For the new HMM:

$$P(c_i = k_{new}|c_{-i}, \mathbf{X}_i, \theta_{knew}, \cdots) \propto \frac{\gamma}{N_{seq} - 1 + \gamma} \int P(\mathbf{X}_i|c_i = k_{new}, \phi)P(\phi)d\phi \tag{4.13}$$

Algorithm 5 Cluster label sampling algorithm

```
1: if number of HMMs are known then
2:     for $k \to 1$ to $K$ do
3:         evaluate $P(c_i = k|c_{-i}, \mathbf{X}_i, \theta_k, \cdots)$ using eq. 4.11
4:     end for
5:     sample $c_i$
6: else
7:     Sample new HMM ($\theta_{knew}$) from $\phi$
8:     evaluate $P(c_i = k_{new}|c_{-i}, \mathbf{X}_i, \theta_{knew}, \cdots)$ using eq. 4.13
9:     for $k \to 1$ to $K$ do
10:        evaluate $P(c_i = k|c_{-i}, \mathbf{X}_i, \theta_k, \cdots)$ using eq.4.12
11:    end for
12:    sample $c_i$
13: end if
```

4.3.3 Hidden state s_t

The hidden state s_t is sampled using a blocked Gibbs sampler. A blocked Gibbs sampler samples from a joint distribution of variables conditioned on all other variables in the system. The state variables are block sampled given all latent variables, hyperparameters and observations in the system.

The backward-sampling technique discussed in section 2.3.2.3 carries out blocked Gibbs sampling effectively; it samples all the state variables for a given HMM and sequence in one go rather than modelling probabilities of the form $P(s_t|s_{-t})$ i.e., the conditional probability of state variable at time t given the state variables at all other times.

Here the state of the last sample is fixed to be state-3. This comes from our model assumptions where it has been assumed that the last sample in each sequence has to be state-3. The conditional probability of state variable s_t given all observations, system parameter and hyperparameters is given as a categorical distribution with N state probabilities:

$$P(s_t = j|\cdots) \propto Cat(s_t|v_t(1), \cdots, v_t(N)) \tag{4.14}$$

v_t is defined from section 2.3.2.3 as:

$$v_t(j) = \alpha_t(j) \cdot a_{jl}$$

$\alpha_t(j)$ is the forward variable at observation t and state j as explained in section 2.3.1 and a_{jl} denotes transition probability from state j to state l with $s_{t+1} = l$. An illustration for assignment of state to each sample for every sequence in an HMM is shown in figure 4.9. The figure shows all the sequences assigned to a specific HMM. Each sample within each sequence is assigned a state using the procedure explained above. The algorithm

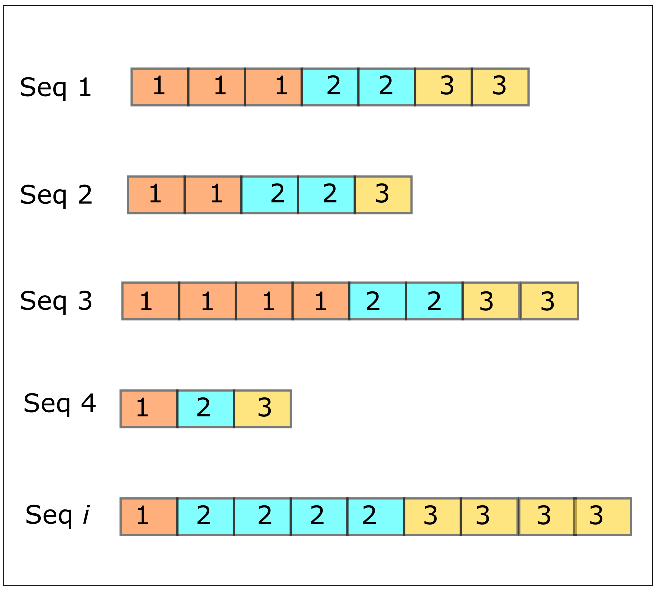


Figure 4.9: Hidden state assignment for every sample in each sequence corresponding to one HMM.

for state assignment can be applied to all sequences with same cluster label.

4.3.4 HMM parameters θ

Once the state assignments have been obtained for samples corresponding to an HMM, the parameters of that HMM can be learnt. These parameters comprise:

Algorithm 6 State variables sampling algorithm for one sequence of length T

```
for $j \rightarrow 1$ to $N$ and $t \rightarrow 1$ to $T$ do
    evaluate forward variable $\alpha_t(j)$
end for
set $s_T = N$
for $t \rightarrow T-1$ to 1 do
    $l = s_{t+1}$
    for $j \rightarrow 1$ to $N$ do
        $v_t(j) = \alpha_t(j)a_{jl}$
    end for
    sample $s_t$ using 4.14
end for
```

- Transition probabilities a_{ij}.
- GMM attributes for each state:
  - mixture weights (w)
  - mixture means (μ)
  - mixture precisions (λ)

4.3.4.1 HMM parameters: Transition Probabilities a_{ij}

For the HMM k, the set of transition probabilities $\{a_{ij}\}$ needs to be learnt. This can be done by imposing a symmetric Dirichlet distribution prior on probability of transitioning to all states given a particular state i.e.,

$$a_i \sim Dir(a_i; \eta_0) \tag{4.15}$$

Given a sequence with its state assignments, a vector $\underline{r}$ can be defined. This vector contains the set of transitions for a given sequence. Figure 4.10 shows one such sequence

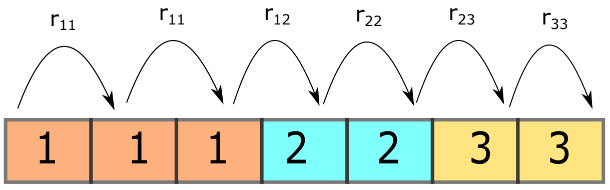


Figure 4.10: A sequence with corresponding state assignments and transitions **r**.

with given state assignments and corresponding transitions. In the figure, r_{ij} represents a transition from state i to j. The vector $\underline{r}$ can be constructed for this sequence as:

$$\underline{r} = \{r_{11}, r_{11}, r_{12}, r_{22}, r_{23}, r_{33}\}$$

Now, the transitions from a given state can be extracted for each state from the above vector as:

$$\underline{r_1} = \{r_{11}, r_{11}, r_{12}\}$$
$$\underline{r_2} = \{r_{22}, r_{23}\}$$

Note that since no updates need to be made for state-3 (as the system will always remain in state-3 once it reaches state-3), the vector r_3 need not be considered and hence $a_{33} = 1$ for all HMMs. Now, the likelihood of $\underline{r_i}$ being observed given the set of probabilities of transitioning from state i i.e., a_i is a multinomial distribution:

$$P(\underline{r_i}|\underline{a_i}) = Mult(\underline{r_i}; \underline{a_i}) \tag{4.16}$$

$$\implies P(\underline{r_i}|\underline{a_i}) \propto \prod_{j=1}^{N_r i} a_{ij} \tag{4.17}$$

where N_{ri} is the number of transitions observed from state i. For figure 4.10, $N_{r1} = 3$ and $N_{r2} = 2$. Now, equation 4.15 and 4.16 can be used to obtain the conditional posterior distribution of $\underline{a_i}$ as:

$$P(\underline{a_i}|\underline{r_i}, \eta) \sim P(\underline{r_i}|a_i) \cdot P(a_i; \eta)$$

The above conditional posterior is also a Dirichlet distribution since the Dirichlet distribution is a conjugate prior for the multinomial distribution which is the likelihood. Thus, the conditional posterior on a_i can be written as:

$$P(\underline{a_i}|\underline{r_i}, \eta) \propto Dir(\underline{a_i}; \underline{\eta'}) \tag{4.18}$$
with,
$$\eta'_j = \eta_{0j} + n_{ij}$$

n_{ij} denotes the number of transitions from state i to state j. For figure 4.10, for determining a_1, $n_{11} = 2$ and $n_{12} = 1$.

The same procedure can be applied for state-2 as well. In case multiple sequences get mapped to an HMM (which will be the usual case), the vector $\underline{r}$ will contain transitions from all the sequences. The procedure for learning a_{ij} for a given HMM and state assignments s_t for all sequences within this HMM can be stated as:

Algorithm 7 Algorithm for learning transition probabilities

```
1: procedure LEARNTRANSITION($s_t, \eta$)
2:     for $i \to 1$ to 2 do
3:         construct $\underline{r_i}$
4:         evaluate $P(\underline{a_i}|\underline{r}_i, \eta)$ using 4.18
5:         update $\eta$'
6:         sample $\underline{a}_i = \{a_{ii}, a_{i(i+1)}\}$
7:     end for
8:     return $\{a_{ij}\}$
9: end procedure
```

4.3.4.2 HMM parameters: Mixture attributes m_l, μ, λ

As mentioned previously, each state within an HMM is represented using a Gaussian Mixture Model. After getting the state assignments, all the samples corresponding to a state can be used to learn the parameters of the GMM corresponding to that state. For example, in the figure 4.9, all samples that have been assigned state-1 can be used to learn the parameters of the GMM that represents state-1. If L samples have the same state assignment $s_t = s$, they can be represented as $\mathbf{O}_s = \{o_1, o_2, ..., o_L\}$.

The first task with regards to learning GMM parameters is to assign a mixture label (m_l) to each sample in $\mathbf{O}_s$. This is achieved by using Gibbs sampling. A conditional posterior on the mixture labels m_l is defined given the parameters of the HMM (θ_k for the k-th HMM) from the last iteration, the state label $s_t = s$, the current sample o_l and all the other observations and hyperparameters in the system (represented by dots) as:

$$\begin{aligned} P(m_l = m|\theta_k, s_t, o_l, \cdots) &\propto P(m_l = m|\theta_k, s_t)P(o_l|\theta_k, s_t, m_l = m) \\ &= w^m_{k,s_t} P(o_l|\mu^m_{k,s_t}, \lambda^m_{k,s_t}) \end{aligned} \tag{4.19}$$

The above expression is evaluated for all $m \in [1, 2, ..., N_m]$ with the total number of mixture components per GMM equal to N_m. A sample is drawn from the above normalised distribution that is derived using Baye's rule and consists of a conditional prior and the likelihood of the sample o_l being generated from the m-th mixture component having mean μ^m_{k,s_t} and precision λ^m_{k,s_t}. Each of these terms need to be updated for the next iteration and will be dealt with separately.

Learning mixture weights w^m_{k,s_t}:

The conditional prior term $P(m_l = m|\theta_k, s)$ signifies the probability of the mixture label of l-th sample in $\mathbf{O}_s$ to be equal to m given the HMM parameters for k-th HMM and state label $s_t = s$ of the sample. It is represented by $w^m_{k,s}$. In other words, $w^m_{k,s}$ represents the mixture weights for m-th mixture corresponding to state s of HMM k. Let $\underline{w_{k,s}}$ represent the vector of weights for such a GMM. A symmetric Dirchlet probability distribution is first imposed on $\underline{w_{k,s}}$ with hyperparameter ρ as:

$$\underline{w_{k,s}} \sim Dir(\underline{\rho})$$

Let $\underline{m_{k,s}}$ denote the mixture label assignments of the samples from 4.19. The posterior of $\underline{w_{k,s}}$ is obtained as:

$$P(\underline{w_{k,s}}|\underline{m_{k,s}}, \rho, \cdots) \propto P(\underline{w_{k,s}}; \rho) \cdot P(\underline{m_{k,s}}|\underline{w_{k,s}}) \tag{4.20}$$

$P(\underline{m_{k,s_t}}|\underline{w_{k,s_t}})$ is a multinomial distribution. Thus, equation 4.20 now becomes:

$$\begin{aligned} P(\underline{w_{k,s}}|\cdots) &\propto Dir(\underline{w_{k,s}}; \rho) \cdot Mult(\underline{m_{k,s}}; \underline{w_{k,s}}) \\ &\propto Dir(\underline{m_{k,s}}; \underline{\rho'}) \end{aligned} \tag{4.21}$$

The last step occurs since Dirichlet prior is a conjugate prior for the multinomial distribution. Hence, the posterior on the weights is also a Dirichlet distribution with updated parameter ρ' given as:

$$\rho'_m = \rho_m + \sum_{m_t \in \underline{m}_{k,s_t}} \delta(m_l, m)$$

where δ signifies the kronecker delta function and ρ'_m represents the Dirichlet hyperparameter for m-th mixture. Using the updated Dirichlet hyperparameters, an updated set of weights can be obtained for the next iteration.

Learning mixture mean (μ) and precision (λ):

The likelihood term $P(o_l|\mu^m_{k,s_t}, \lambda^m_{k,s_t})$ in the equation 4.19 relies on the parameters of the relevant mixture of the GMM i.e., the mean $\mu^m_{k,s}$ and precision $\lambda^m_{k,s}$. These parameters can be learnt for the next iteration once the mixture labels have been obtained using eq. 4.19.
Assuming that the dimensions d of the feature vector are independent of each other, a normal-gamma distribution is used as a prior on the joint distribution of the mean and precision:

$$\begin{aligned} & P(\mu^{m,d}_{k,s}, \lambda^{m,d}_{k,s}; \mu_0, \kappa_0, \alpha_0, \beta_0) \sim NG(\mu^{m,d}_{k,s}, \lambda^{m,d}_{k,s}|\mu_0, (\kappa_0\lambda^{m,d}_{k,s})^{-1}, \alpha_0, \beta_0) \\ \implies & P(\mu^{m,d}_{k,s}, \lambda^{m,d}_{k,s}; \mu_0, \kappa_0, \alpha_0, \beta_0) \sim \mathcal{N}(\mu^{m,d}_{k,s}|\mu_0, (\kappa_0\lambda^{m,d}_{k,s})^{-1}) Ga(\lambda^{m,d}_{k,s}|\alpha_0, \beta_0) \end{aligned}$$

where, $\mu_0, \kappa_0, \alpha_0, \beta_0$ are the hyperparameters of the normal-gamma distribution and are assumed to be known. For each HMM k, state s_t, GMM mixture m and dimension d, the posterior joint distribution of mean and precision can be updated. Denoting samples mapped to m-th mixture component by $\mathbf{o} = \{o_1, o_2, ..., o_n\}$ and dropping subscripts and superscripts to simplify notation, the updates can be formulated by obtaining the conditional posterior distribution of μ given λ and samples $\mathbf{o}$:

$$P(\mu, \lambda|\mathbf{o}) = P(\mathbf{o}|\mu, \lambda) \cdot P(\mu, \lambda; \mu_0, \kappa_0, \alpha_0, \beta_0)$$

The likelihood term in the above equation is a Gaussian distribution since samples have been assumed to be Gaussian distributed. This means that the conditional posterior term in the above equation will also have a Normal-Gamma distribution with updated parameters since Normal-Gamma distribution is a conjugate prior for Gaussian distribution.

$$\begin{aligned} & P(\mu, \lambda|\mathbf{o}) = NG(\mu, \lambda; \mu', \kappa', \alpha', \beta') \\ & \text{with,} \\ & \mu' = \kappa_0\mu_0 + n\bar{\mathbf{o}} \\ & \kappa' = \kappa_0 + n \\ & \alpha' = \alpha_0 + \frac{n}{2} \\ & \beta' = \beta_0 + \frac{1}{2}\sum_{i=1}^{n}(\underline{o}_i - \bar{\mathbf{o}})^2 + \frac{\kappa_0 n(\bar{\mathbf{o}} - \mu_0)^2}{2(\kappa_0 + n)} \end{aligned}$$

$\bar{\mathbf{o}}$ denotes the empirical mean of the corresponding dimension of all samples assigned to the mixture.

Once the joint conditional posterior of μ and λ is available, the conditional posteriors for individual random variables μ and λ can be obtained as:

$$P(\mu|\lambda, \mathbf{o}) = \mathcal{N}(\mu; \frac{\kappa_0\mu_0 + n\bar{\mathbf{o}}}{\kappa_0 + n}, (\kappa_0 + n)\lambda^{-1}) \tag{4.22}$$

$$P(\lambda|\mathbf{o}) = Ga(\lambda; \alpha_0 + \frac{n}{2}, \beta_0 + \frac{1}{2}\sum_{i=1}^{n}(o_i - \bar{\mathbf{o}})^2 + \frac{\kappa_0 n(\bar{\mathbf{o}} - \mu_0)^2}{2(\kappa_0 + n)}) \tag{4.23}$$

The sampling for μ and λ is done by first sampling λ from the gamma distribution which has updated hyperparameters α' and β' i.e.,

$$\lambda' \sim Ga(\lambda; \alpha', \beta')$$

Then using this sampled precision, the mean is sampled:

$$\mu \sim \mathcal{N}(\mu; \mu', (\kappa'\lambda')^{-1})$$

A procedure called LEARNGMM($\underline{w_{k,s}}, \underline{\mu_{k,s}}, \underline{\lambda_{k,s}}$) can now be defined that takes the weight vector, the means and the precisions from the previous iterations and outputs updated weights, means and precisions for a given GMM:

Algorithm 8 Algorithm for learning GMM attributes

```
 1: procedure LearnGMM($\underline{w_{k,s}}, \underline{\mu_{k,s}}, \underline{\lambda_{k,s}}$)
 2:     for $l \rightarrow 1$ to $L$ do
 3:         get $m_l$ using eq. 4.19
 4:     end for
 5:     for $m \rightarrow 1$ to $N_m$ do
 6:         get all samples($\mathbf{o}$) with $m_l = m$
 7:         update $w^m_{k,s}$ using eq. 4.21
 8:         update $\mu^m_{k,s}$ using eq. 4.22
 9:         update $\lambda^m_{k,s}$ using eq. 4.23
10:     end for
11:     return ($\underline{w_{k,s}}, \underline{\mu_{k,s}}, \underline{\lambda_{k,s}}$)
12: end procedure
```

Simulation Set up

This chapter discusses the experimental set up and describes the factors on which different simulations are carried out. First, the TIMIT database that has been used for carrying out the experiments is described. Then, the various factors on which different experiments are created are presented. This is followed by a discussion of the evaluation metric which is cluster purity. Lastly, details of implementations are presented.

5.1 TIMIT Database

TIMIT database has been designed to conduct research at a phonetic level of speech. It consists of phonetically transcribed speech covering 8 dialects of American English spoken by both males and females. There are a total of 6300 sentences divided into training, development and test sets. The number training sentences are 4620 and only these sentences have been used for conducting experiments in this thesis. The utterances are stored as 16 KHz speech waveforms files. The time aligned phonetic transcriptions of the utterances is provided. Therefore, the true phonetic label of each sample sequence is known.

5.2 Mel Frequency Cepstral Coefficients (MFCCs)

MFCCs are being used as input feature vectors to the algorithm. The utterances from TIMIT corpus are converted into Mel Frequency Cepstral Coefficients. MFCCs tend to emulate human auditory perception well and have been the state of the art in studies related to speech processing [KS04]. Thus, they are used as feature vectors for this study as well. The procedure to obtain MFCCs is described in [DM80]. The frame duration considered is 25 ms which corresponds to 400 samples in the TIMIT corpus. A frame overlap of 10 ms has been considered to counteract the effect of discontinuity from one frame to next. A total of 13 Mel coefficients were retained along with 13 delta and 13 delta-delta coefficients. This means that there are 39 dimensions in the feature vector.

5.3 Experimental setup

There are several distinguishing factors on which the experiments have been designed. This section describes these factors and how a variation in these factors lead to different experiments.

- Boundary variable b_t: In certain experiments, the true values of boundary variables are known for the entire speech recording. This implies that the segmentation step of the algorithm is not required. Thus, only the clustering and learning steps need to performed.
- Number of HMMs: If it is assumed that the number of phonemes for a given language are known, then the total number of HMMs are also known. Algorithmically speaking, this implies that the Dirichlet process is no longer required i.e., each sequence only needs to be mapped to one of the known HMM and there is no need of sampling a new HMM at any point.
- Learning algorithm: After the segmentation and clustering steps, the parameters of each HMM needs to be learnt. The learning step can be performed in a number of ways: using Gibbs sampling as discussed in chapter-4 or using Baum-Welch algorithm which was presented in chapter-2. Another variation that is tried is a mix of both of these approaches where the first half of the total number of iterations of the algorithm use Gibbs sampling and the last half uses Baum-Welch algorithm for learning HMM parameters. This is done since Baum-Welch algorithm corresponds to a local optimization approach i.e., it is an EM method that gives a good local solution. Once a parameter set has been determined for an HMM using Gibbs sampling, Baum-Welch algorithm further tunes these parameters and tries to determine a good local solution. The presegmentation, segmentation and clustering approaches remain same as before regardless of the learning approach.
- Supervised/Unsupervised training: In the supervised approach, the data-base is divided into sets of training and validation sequences. It is ensured that each phoneme label has a representation in both the training and validation data. At each iteration of training the supervised algorithm, since the sequence labels are already known and the boundary variable is also known, only the learning step of the algorithm is performed to learn the parameters of each HMM corresponding to phoneme. The HMM parameters hence learnt are used to cluster the validation data sequences. The training data is also clustered to compare the clustering quality with the validation data. In the unsupervised algorithm, the cluster labels are unknown, hence all steps of the algorithm are performed.
- Number of mixtures per GMM: As already mentioned, each HMM is composed of 3 states where each state is modelled by a Gaussian mixture model. The number of components in GMM can be varied to change the model complexity. The number of components considered for experiments are: 1, 8, 32. Also, in some experiments, the number of components are gradually increased by doubling the number of mixture components every few iterations: for the first 50 iterations,

the number of components is 1 i.e., just a Gaussian distribution. Then, after 50 iterations, the number of components are doubled i.e., from iterations 51 to 100, the number of components are 2. This gradual splitting continues until the number of components become 32. Denoting the number of old mixtures by N_m and new by N'_m, the splitting process can be described as a step by step process:

(i) Get N'_m number of initial means from the means of the old mixtures by moving one standard deviation to the left and right of the mean of the old component. Distribute the mixture weights equally between the new components. This is shown for a single component in figure 5.1.

(ii) The N'_m initial means obtained in the first step are then fed as input to k-means clustering algorithm ([HW79]) which gives a set of N'_m clusters with updated means. A covariance matrix for each of these N'_m clusters can also be obtained once the mixture identity of each sample is determined by k-means algorithm.

(iii) The means and covariances obtained in the second step are then fed along with all the samples assigned to this state of a given HMM to an EM algorithm. The mixture weights are obtained from step (i). The EM algorithm, which is a soft clustering algorithm, further tunes up the means, covariances and mixture weights for each new mixture component of a given state of an HMM.

Parameters for the new components are hence obtained.

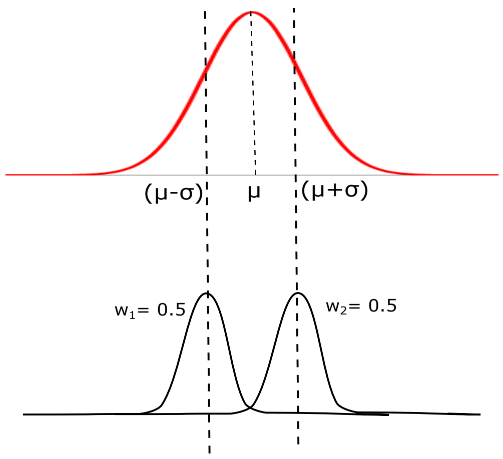


Figure 5.1: Splitting of Gaussian distributions: Step (i) of the splitting process.

- HMM attributes initialization approach: There are different ways the attributes of the HMMs can be initialised. These attributes are: means(μ), precisions(λ), weights(w) of the GMM components of each state of each HMM and the transition probabilities $\{a_{ij}\}$ within each HMM. Initialization approaches are discussed next.

5.4 HMM attributes initialization

In the scenario where data is pre-segmented and the number of HMMs are known, we have:
Given: Number of HMMs (K), number of mixture components per GMM (N_m), the global mean (μ_G) of complete dataset, global covariance (σ_G) of complete dataset.
Task: To initialise $a_{ij} \forall \{i, j\} \in \Omega : \{1, 2, 3\}$ and $w^m_{k,s}, \mu^m_{k,s}, \lambda^m_{k,s}$ for each of the K HMMs, S states and N_m mixture components.
Two methods of initialization can be adopted:

(i) Random start
Sample precision $\lambda^m_{k,s}$ from a gamma distribution with shape parameter α_0 and inverse scale parameter $\beta_0 = \frac{1}{\sigma_G}$using equation:

$$\lambda^m_{k,s} \sim Ga(\alpha_0, \beta_0) \tag{5.1}$$

Using the precision sampled above, sample $\mu^m_{k,s}$ from a normal distribution;

$$\mu^m_{k,s} \sim \mathcal{N}(\mu_G, (\kappa_0 \lambda^m_{k,s})^{-1}) \tag{5.2}$$

where κ_0 is a fixed parameter.

The GMM weights are initialised by assigning equal weights to all mixture components i.e.,

$$w^m_{k,s} = \frac{1}{N_m} \tag{5.3}$$

The transition probabilities are obtained by sampling $a_i \forall i \in \{1, 2\}$ from Dirichlet distribution with hyperparameter $\underline{\delta} = \{\delta_1, \delta_2\}$:

$$a_i \sim Dir(\underline{\delta}) \tag{5.4}$$

The value of δ_1 and δ_2 is kept the same (typically 1) so that no transition is favoured more than the other. For state-1, a_{11} and a_{12} get sampled using 5.4. Similarly, for state-2, a_{22} and a_{23} get sampled for $i = 2$ in equation 5.4. This gives the set of probabilities $\{a_{11}, a_{12}, a_{22}, a_{23}\}$. Note that the transitions probabilities need not be initialised for $i = 3$ since once in state-3, the system remains in state-3.

(ii) Flat start:
Sample a cluster label c_i for each sequence such that the probability of occurrence of each label is $\frac{1}{K}$. This is equivalent to sampling from a categorical distribution with K parameters $\{1/K\}$ i.e.,

$$c_i \sim Cat(\{1/K\}_{1\times K}) \tag{5.5}$$

The sequences with the same cluster label belong to the same HMM. Samples from each of these sequences are then assigned a state. For a sequence of length T, state assignment rules are described in 5.1. In table 5.1, $floor(x)$ = integer part of x. For example, $floor(2.6) = 2$. A few examples of state assignment are shown in figure 5.2 for different values of T.

Algorithm 9 Algorithm for random initialisation of HMM attributes

```
 1: for $k \to 1$ to $K$ do
 2:     for $s \to 1$ to 3 do
 3:         for $m \to 1$ to $N_m$ do
 4:             sample $\lambda^m_{k,s}$ using eq. 5.1
 5:             sample $\mu^m_{k,s}$ using eq. 5.2
 6:             obtain $w^m_{k,s}$ using eq. 5.3
 7:         end for
 8:         sample $a_s$ using 5.4
 9:     end for
10: end for
```

State index	Sample index
1	1 to $floor(T/3)$
2	$(floor(T/3) + 1)$ to $(2 \times floor(T/3))$
3	$(2 \times floor(T/3) + 1)$ to T

Table 5.1: Rule for state assignment for flat start initialisation

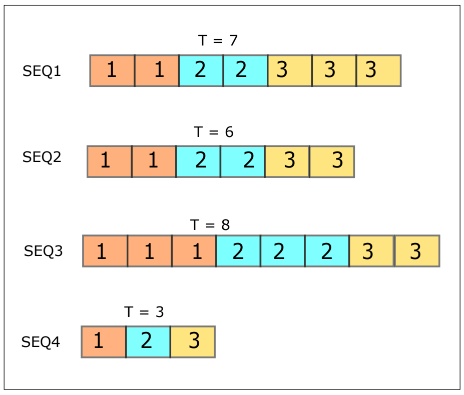


Figure 5.2: State assignment in flat start initialisation

Once the state assignments for all sequences within an HMM are obtained, transition probabilities are initialised as:

$$a_{ij} = \frac{\text{Number of transitions from state } i \text{ to state } j}{\text{Number of transitions from state } i} \tag{5.6}$$

Again, transitions need to be initialised for the set $\{a_{11}, a_{12}, a_{22}, a_{23}\}$ since $a_{ij} = 0, \forall i > j$ and $a_{33} = 1$ due to the left-right HMM assumption.

The attributes of a GMM for a state can be initialised by using samples assigned to that particular state. This is done by first sampling precision for each of the N_m

mixtures:

$$\lambda^m_{k,s} \sim Ga(\alpha_0, \beta_0) \tag{5.7}$$

where, β_0 is the inverse of the variance of the samples assigned to the s-th state of k-th HMM i.e.,

$$\beta_0 = \frac{1}{\sigma_s}$$

Once precision has been sampled, mean for every mixture can be sampled from a normal distribution using the newly sampled precision and the mean of all the samples assigned to s-th state:

$$\mu^m_{k,s} \sim \mathcal{N}(\mu_s, (\kappa_0 \lambda^m_{k,s})^{-1}) \tag{5.8}$$

The process to sample GMM mixture weights begins by first assigning a mixture label to each sample (for a given state). This is done using a categorical distribution with all probabilities set to $\{1/N_m\}$ i.e.,

$$m_l \sim Cat(\{1/N_m\}_{1\times N_m}) \tag{5.9}$$

Once the mixture labels have been obtained, the number of samples that got assigned to each mixture label is counted (n_m). Now, the mixture weights are sampled from a Dirichlet distribution with the Dirichlet hyperparameter of a mixture incremented by the count of corresponding mixture:

$$\begin{aligned} &w_{k,s} \sim Dir(\underline{\eta'}) \\ &\underline{\eta'} = \{\eta'_1, \eta'_2, ..., \eta'_{N_m}\} \\ &\eta'_m = \eta_0 + n_m \end{aligned} \tag{5.10}$$

5.5 Sampling base distribution (ϕ)

If the number of HMMs are unknown, Dirichlet process is used to sample a new possible HMM for each sequence. This new HMM is sampled from a base distribution with parameters (ϕ). The base distribution has a hyperparameter set:

$$\phi = \{\alpha_0, \beta_0, \kappa_0, \mu_0, \eta_0\}$$

where:

- α_0, β_0 determine the precision of the GMM mixture components λ:

$$\lambda \sim Ga(\alpha_0, \beta_0)$$

- μ_0, κ_0 determine the means of the GMM mixture components μ:

$$\mu \sim \mathcal{N}(\mu_0, (\kappa_0 \lambda)^{-1})$$

- η_0 determine the transition probabilities a_{ij}:

$$a_{ij} \sim Dir(\eta_0)$$

A new HMM is sampled from the base distribution exactly in the same way as the random initialisation discussed in section 5.4. The typical values of various hyperparameters used are summarised in table 5.2.

Hyperparameter	Value
α_0	1
β_0	inverse of the global variance of complete dataset
κ_0	1
μ_0	global mean of complete dataset
η_0	1

Table 5.2: Typical values of hyperparameters for sampling base distribution

5.6 Evaluation measure: Cluster Purity

The metric that has been used to evaluate and compare performance of algorithm under various conditions is cluster purity. Cluster purity aims at judging the quality of the identified clusters by evaluating the percentage of the majority classes found in each of the cluster. It is evaluated using the following equation:

$$\text{Cluster Purity (CP)} = \frac{1}{N} \sum_{K} \max_{j} |\{\omega_k | L(\omega_k) = j\}|$$

where,
w_k is the set of all samples that have been assigned to k-th cluster.
$L(w_k)$ is the set of true class labels of the samples within the k-th cluster.
N is the total number of samples in the entire sample space.
K is the total number of clusters.

The above equation for cluster purity determines the class for each cluster that has maximum representation within that cluster by finding the number of samples that belong to this majority class. This is done for every discovered cluster. These maximum counts from each of the identified clusters are then added together and normalised by the total number of samples in the entire sample space. This gives a measure of how pure the identified clusters are on average.

Figure 5.3 explains cluster purity. It shows three clusters learnt from a data that originates from three different true classes. These classes are depicted using different shapes: a square, a star and a pentagon. As can be seen from the figure, each cluster gets some samples from each of the true classes. The numbers of each of these true classed is depicted below the corresponding cluster. For cluster 1, the 'star' class has

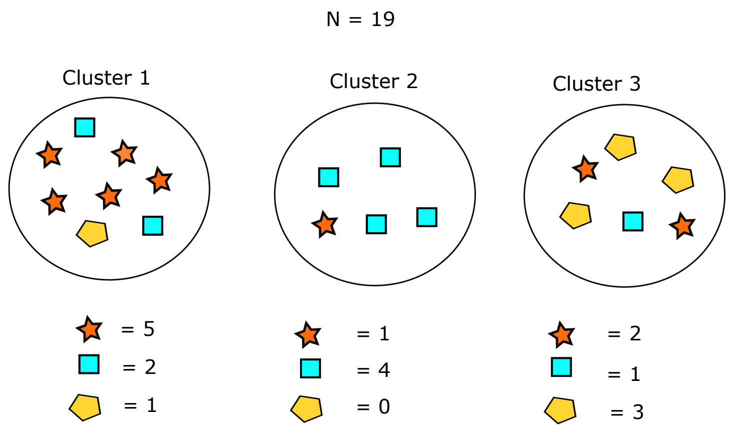
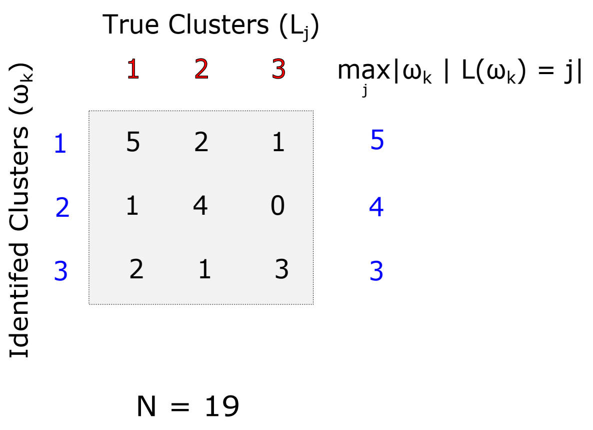


Figure 5.3: Cluster Purity Depiction

maximum representation with 5 samples as compared to other classes which have 2 and 1 samples assigned to this cluster respectively. For cluster 1, the term $max|w_k|L(w_k)| = 5$. Maximum counts can be found similarly for cluster 2 and cluster 3 as 4 and 3 respectively. Hence, the cluster purity for this example can be obtained as:

$$\begin{aligned} CP &= \frac{1}{19}(5+4+3) \\ &= 0.6315 \\ \implies CP &= 63.15\% \end{aligned}$$

In practice, it is easier to work with a confusion matrix which is a $\{K \times H\}$ matrix with K number of discovered clusters and H number of true clusters. A confusion matrix can be built for the example depicted in figure 5.3.



True Clusters (L_j)

Identifed Clusters (ω_k)	1	2	3	$\max_j \vert\omega_k \mid L(\omega_k) = j\vert$
1	5	2	1	5
2	1	4	0	4
3	2	1	3	3

N = 19

5.6.1 Cluster Purity on TIMIT database

TIMIT database contains 61 phonemes that represent the total number of phonemes in the English language. But results are often reported on a reduced set of 48 phonemes which are extracted by deleting a class of phonemes and folding some of the phonemes together into a single class. This is done following the process specified in [LH89]. Thus, for the experiments in this thesis, this reduced set of 48 phonemes are given as the input to the algorithm. The output of the algorithm is obtained by folding these 48 phonemes into 39 phonemes as has been suggested in [LH89].

The process of folding for evaluation (48 to 39) can be explained with the help of a minimal example in which there are 4 true classes and 4 discovered clusters. A total of 413 sequences are present in a fictional database. The cluster purity without folding is 40.67% as shown in the first confusion matrix in the figure 5.4. Now, lets say that the classes 2 and 4 need to be folded together as a part of the phoneme reduction process. The entries in columns 2 and 4 corresponding to true cluster labels 2 and 4 are added together to produce an updated confusion matrix shown in the right hand side of figure 5.4. This leads to an increased cluster purity of 52.78%.

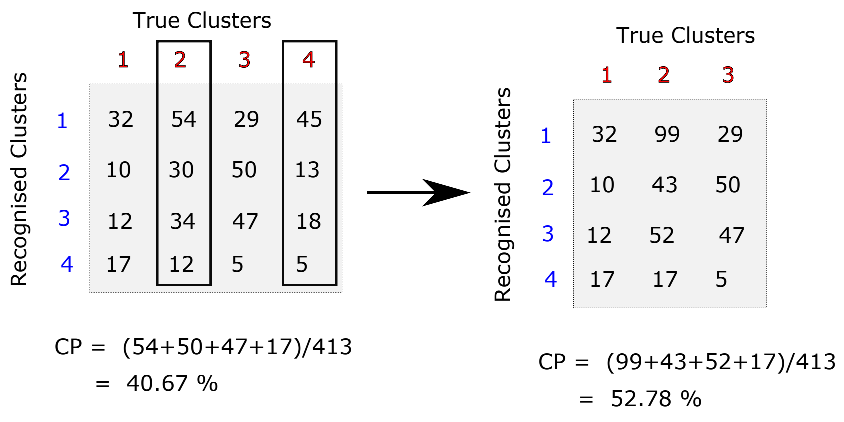


Figure 5.4: Phoneme folding for unsupervised case

Note that only the true class labels are folded together i.e., only the entries along the columns in the confusion matrix are added and not the rows. This is because the ground truth is not known for the discovered clusters in an unsupervised setting. This means that the true identity of a discovered cluster is not known and the folding stops after adding the entries of those columns of the confusion matrix which are to be folded into one class. In a supervised setting however, the identity of all the clusters are known. This is because each of the cluster is trained using the sequences corresponding to a particular phoneme only. Thus, there is a one-to-one correspondence between the actual and discovered labels of each sequence. This implies a two-step folding for the supervised case as illustrated in figure 5.5.

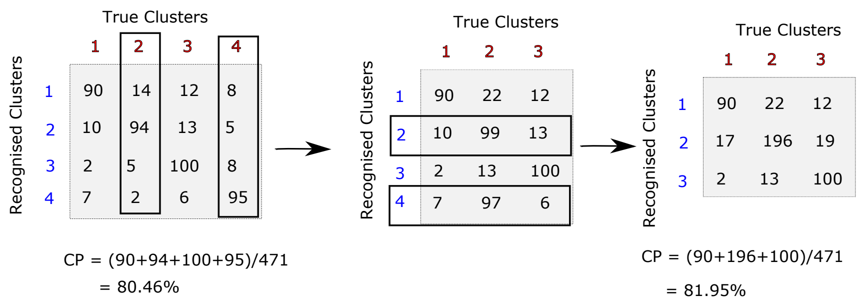


Figure 5.5: Two-step phoneme folding for supervised case

Again, a fictitious database having 4 true classes is considered. Initially, the cluster purity is 80.46% without folding. If sequences from classes 2 and 4 are to be folded together, then the two step folding process can be explained as:

1. In the first step, the entries in the columns 2 and 4 of the confusion matrix are added. This gives a 4×3 confusion matrix.
2. In the second step, the entries in rows 2 and 4 of the confusion matrix are added together. This gives a 3×3 confusion matrix with a cluster purity of 81.95%.

These steps are of course, inter-changeable.

5.7 Evaluating segmentation performance

When the speech utterances are unsegmented, segmentation algorithms are deployed to infer the boundary variables as explained in chapter-4. A way to evaluate the performance of these segmentation/ presegmentation algorithms is required since the effectiveness of segmentation has a significant impact on the performance of the algorithm as a whole.

If wrong segments are detected, then samples that should correspond to a particular phoneme become a part of other phoneme which leads to erroneous results. A measure of correctness of the discovered segment boundaries can be obtained by evaluating the ratio of the number of correctly detected boundaries to the total number of boundaries in the data. A boundary is said to be correctly detected if it lies within a range of 2 samples from the correct boundary. This measure is known as *recall.*

$$recall = \frac{\text{number of correctly detected segmentation points}}{\text{number of true segmentation points}}$$

A high recall is desired since it means that a high number of true boundaries have been discovered.

Another interesting metric to evaluate segmentation performance is the *over-segmentation* percentage. It is the ratio of the number of boundaries detected using the segmentation algorithm to the total number of boundaries in the data. It is given as:

$$Oversegmentation(O_s) = \frac{\text{Number of segmentation points obtained}}{\text{Total number of true segmentation points}} \times 100$$

If the value of O_s is higher than 100%, it means that the utterance is over-segmented i.e., more number of boundaries have been discovered as compared to the true number of boundaries. This implies that the sequence length of each phoneme would be shorter on average in comparison to the true average sequence length. In case the value of O_s is less than 100%, it means that less number of boundaries have been discovered as compared to the true number of boundaries which leads to a longer sequence length.

5.8 Implementation details

This section discusses the details with regard to implementation of simulations for various experiments.

5.8.1 Cluster label assignment in unsegmented data

When the data is pre segmented, the labels are assigned to each sequence using Dirichlet process represented by equations discussed in chapter-4:

$$P(c_i = k | c_{-i}, \mathbf{X}_i, \theta_k, \cdots) \propto \frac{N_{-k}}{N_{seq} - 1 + \gamma} \cdot P(\mathbf{X}_i | \theta_k) \tag{5.11}$$

For the new HMM:

$$P(c_i = k_{new} | c_{-i}, \mathbf{X}_i, \theta_{knew}, \cdots) \propto \frac{\gamma}{N_{seq} - 1 + \gamma} \int P(\mathbf{X}_i | c_i = k_{new}, \phi) P(\phi) d\phi \tag{5.12}$$

Hence, for each sequence, the cluster label for a sequence is sampled given the cluster labels of all other sequences. This approach can not be directly applied to the case with unsegmented data. This is because the samples that make up a sequence in the last iteration may no longer be the part of the same sequence. This implies that there is no cluster label available for the sequences from the last iteration since the definition of a sequence has changed. A way to understand this problem is by looking at the organisation of data in the TIMIT database. There are a total of 4620 training sentences in the TIMIT database. Each sentence corresponds to multiple phonemes where each phoneme is represented by a sequence of samples. The issue is represented for one such sentence in figure 5.6.

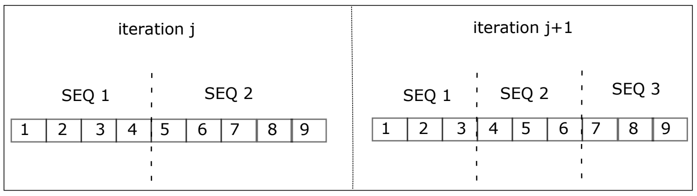


Figure 5.6: Implementation issue with unsegmented data

The figure shows the segmentation of the same sentence having 9 samples in total, in the j-th and $(j+1)$-th iteration. As can be seen from the figure, in the j-th iteration, 2 sequences are discovered with length 4 and 5 respectively. However, in the $(j+1)$-th iteration, 3 sequences each with length 3 are discovered due to a change of the boundary variables in this iteration. Hence, the labels for the sequences in the j-th iteration can no more be used to sample cluster labels for sequences in the same sentence in the $(j+1)$-th iteration.

The solution to this issue is made using the block sampling of the cluster labels for a sentence given the cluster labels of sequences in the other sentences. The equations 5.11 and 5.12 are modified as:

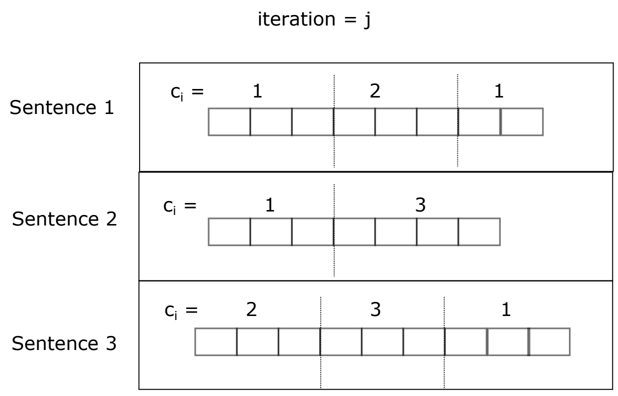
$$P(c_i^R = k | c_{-i}^{-R}, \mathbf{X}_i, \theta_k, \cdots) \propto \frac{N_{-k}^{-R}}{N_{seq}^{-R}} \cdot P(\mathbf{X}_i|\theta_k) \tag{5.13}$$

For new HMM:

$$P(c_i = k_{new} | c_{-i}^{-R}, \mathbf{X}_i, \theta_{knew}, \cdots) \propto \frac{\gamma}{N_{seq}^{-R}} \int P(\mathbf{X}_i | c_i = k_{new}, \phi) P(\phi) d\phi \tag{5.14}$$

Here, c_i^R denotes the cluster label of the i-th sequence in the R-th sentence while c_{-i}^{-R} represent the cluster labels of the sequences that are not in the R-th sentence.
N_{seq}^{-R} represents the total number of sequences excluding the sequences in the R-th sentence.

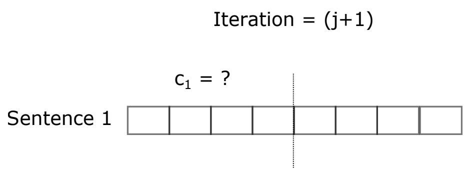
Thus, for each sequence in a given sentence, the proportion of the cluster labels in all the other sentences are used as the prior term in the equation 5.13. An example is shown for a fictional database having 3 sentences in the figure . The boundaries and the corresponding cluster labels for each of the discovered sequences in iteration j is shown in the figure. Also shown is a table with a count of each cluster label throughout the database and the corresponding probabilities of occurrence of each cluster label. For example, out of the 8 sequences in the database, 4 are assigned the cluster label 1. Thus the probability of appearance of cluster 1 is 4/8.



Label	1	2	3
counts	4	2	2
Probability	4 /8	2/8	2/8

Figure 5.7: Unsegmented data assignments at iteration j.

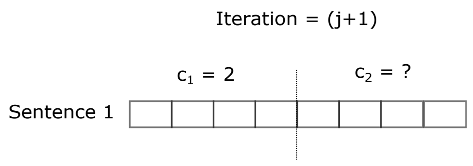
Now, in the iteration $(j + 1)$, the cluster label for each sequence in each sentence needs to be determined. First, labels for sequences in sentence 1 are determined. For carrying out this task, all the labels learnt in the iteration j for sentence 1 are discarded and the corresponding count is reduced from the cluster. The total count is also reduced accordingly. This is illustrated in figure 5.8.



Label	1	2	3
counts	2	1	2
Probability	2 /5	1/5	2/5

Figure 5.8: Unsegmented data assignments for sentence 1 at iteration $(j + 1)$.

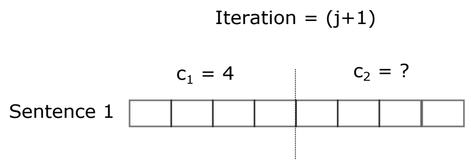
In the figure, it can be seen that in the $(j + 1)$-th iteration, 2 sequences are discovered for sentence-1. At first, the cluster label for the first sequence in the sentence needs to be found. This is done by deducting the counts of the j-th iteration for each cluster label that was present for this sentence in the j-th iteration. An updated count and probability statistic is also shown in figure 5.8. These probabilities are equal to the term $\frac{N_{-k}^{-R}}{N_{seq}^{-R}}$ in equation 5.13. The term $P(c_1^R = k|c_{-1}^{-R}, \mathbf{X}_i, \theta_k, \cdots)$ can now be evaluated for each of the three HMMs in the system shown in the figure. Also, a new HMM can be sampled for this sequence using equation 5.14. Hence, a cluster label can be found for this sequence. If an existing cluster label is sampled for sequence 1, the relevant counts are updated and label for the next sequence in the sentence is found using these updated values. This is shown in figure 5.9. Here sequence 1 gets a label 2 which was an already existing cluster class. The updated probabilities for sampling cluster label of the second sequence (c_2) in sentence-1 is also shown. Now, lets suppose a new cluster



Label	1	2	3
counts	2	2	2
Probability	2 /6	2/6	2/6

Figure 5.9: Unsegmented data assignments for sentence 1 at iteration $(j + 1)$. A cluster label from existing cluster classes is sampled for sequence 1.

label gets sampled for this sequence i.e., cluster label 4. the situation looks as shown in figure 5.10. The updated probabilities for sampling cluster label of the second sequence (c_2) in sentence-1 can also be observed.



Label	1	2	3	4
counts	2	1	2	1
Probability	2 /6	1/6	2/6	1/6

Figure 5.10: Unsegmented data assignments for sentence 1 at iteration $(j+1)$. A cluster label from a new cluster class is sampled for sequence 1.

Hence, a cluster label for sequence-2 can be found depending upon the label of sequence-1. Proceeding the same way for each sentence, cluster labels for each sequence can be found.

5.8.2 Cluster Purity: True label mapping for unsegmented data

Cluster purity evaluation requires building a confusion matrix as explained in the previous section. This confusion matrix can only be made when the true labels of the sequences are known. In the case of unsegmented data, these labels can not be known directly since the number and length of sequences change. In order to get the true labels of these sequences, the rule of maximum overlap is used. The samples of the newly segmented sequences are time aligned with their truly segmented counterparts. The samples in the truly segmented sequences which have maximum overlap with the samples in the newly segmented sequence determine the label of this newly segmented sequence. The process can be explained with the help of figure 5.11.

In the figure, a sentence with 11 samples is shown. The upper part of the figure has true segmentation of the sequence which corresponds to 3 sequences along with their respective true labels. The lower part of the figure shows the newly determined segmentation on the same sentence. Here, 2 sequences are discovered. The label c_1 for sequence 1 is determined to be 2 since 3 of the samples of this sequence overlapped in time with the sequence 1 of truly segmented sequence while only one of the samples of this sequence aligned with sequence 2 of the truly segmented sentence. Hence this sequence gets the label of the sequence 1 of the truly segmented sentence, which is 2. Similarly, the sequence 2 gets a label 5.

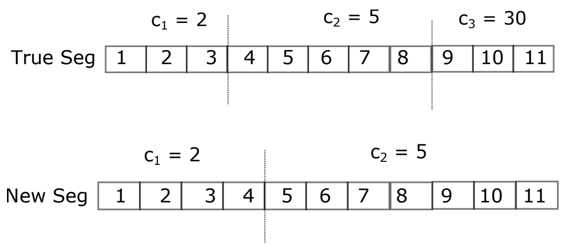


Figure 5.11: Maximum overlap assignment for determining the true labels of newly segmented sequences for evaluation of cluster purity.

5.8.3 Rescaling self transition probabilities for segmentation algorithm

The segmentation algorithm uses a cascaded system where all the HMMs in the system are cascaded together so that boundary variables could be determined as discussed in chapter-4. Reiterating the cascaded transition matrix for a two HMM system from chapter-4:

$$\mathbf{A}_{cas} = \begin{bmatrix} a_{11} & a_{12} & 0 & 0 & 0 & 0 \\ 0 & a_{22} & a_{23} & 0 & 0 & 0 \\ \epsilon_1 \cdot (1 - z_1) & 0 & z_1 & \epsilon_2 \cdot (1 - z_1) & 0 & 0 \\ 0 & 0 & 0 & a_{44} & a_{45} & 0 \\ 0 & 0 & 0 & 0 & a_{55} & a_{56} \\ \epsilon_1 \cdot (1 - z_2) & 0 & 0 & \epsilon_2 \cdot (1 - z_2) & 0 & z_2 \end{bmatrix}$$

The self transition probability of state-3 of HMM k, denoted by z_k, is given as:

$$z_k = \frac{\text{No. of transitions from state-3 to state-3 in HMM } k}{\text{No. of transitions from state-3 in HMM } k + 1}$$

From the expression of z_k, it can be seen that the value of z_k is generally high. This implies that the probability of system leaving the current HMM is very low. The boundaries between different sequences are marked by the event that the system moves from one HMM to the next. If the probability of the system leaving the current HMM is low, these transition events will happen less often. This implies that the sequences will be longer. To avoid this behaviour, z_k can be manually rescaled so that the transitions from one HMM to other HMMs become more probable. Thus, z_k can be set to $\frac{z_k}{F}$ with F as the rescaling factor with property $F > 1$. The cascaded transition matrix for a two HMM system then looks like:

$$\mathbf{A}_{cas} = \begin{bmatrix} a_{11} & a_{12} & 0 & 0 & 0 & 0 \\ 0 & a_{22} & a_{23} & 0 & 0 & 0 \\ \epsilon_1 \cdot (1 - \frac{z_1}{F}) & 0 & \frac{z_1}{F} & \epsilon_2 \cdot (1 - \frac{z_1}{F}) & 0 & 0 \\ 0 & 0 & 0 & a_{44} & a_{45} & 0 \\ 0 & 0 & 0 & 0 & a_{55} & a_{56} \\ \epsilon_1 \cdot (1 - \frac{z_2}{F}) & 0 & 0 & \epsilon_2 \cdot (1 - \frac{z_2}{F}) & 0 & \frac{z_2}{F} \end{bmatrix}$$

5.8.4 Variance flooring

When Baum-Welch algorithm is used for tuning the parameters of the HMM, singularity problem arises for certain GMM components corresponding to an HMM state. A particular mixture component of a GMM may get assigned a single sample or no sample at all. This makes the variance of this Gaussian component zero which gives rise to singularities. The situation becomes worse as the number of mixtures are increased since many components may not get assigned any sample.

To circumvent the problem, the suggestion in [MKLB98] has been followed. A fixed variance floor of 0.01 is provided for every component of GMM for every state and HMM.

Same variance floor is used when using k-means clustering and EM algorithm for splitting the Gaussians as explained in section 5.3. If the covariance of a cluster is less than 0.01 (for each dimension) then it is set to 0.01. Similarly, if the EM algorithm in the last step of splitting determines the covariance of a dimension of a mixture component to be less than 0.01, then the variance of that dimension of the mixture component is set to be 0.01.

5.8.5 Convergence evaluation

Cluster purity, which has been chosen to be the evaluation metric also determines the convergence. Instead of using an objective function to determine convergence, a subjective measure is chosen. The algorithm is said to have converged when there is not much change in the cluster purity for several iterations. This can be seen from the figure 5.12 where the evolution of cluster purity is shown with iterations. The case shown corresponds to a supervised case. In unsupervised case, convergence is achieved in more number of iterations as compared to the supervised case. The last iteration of the algorithm is used to determine the cluster purity corresponding to the scenario.

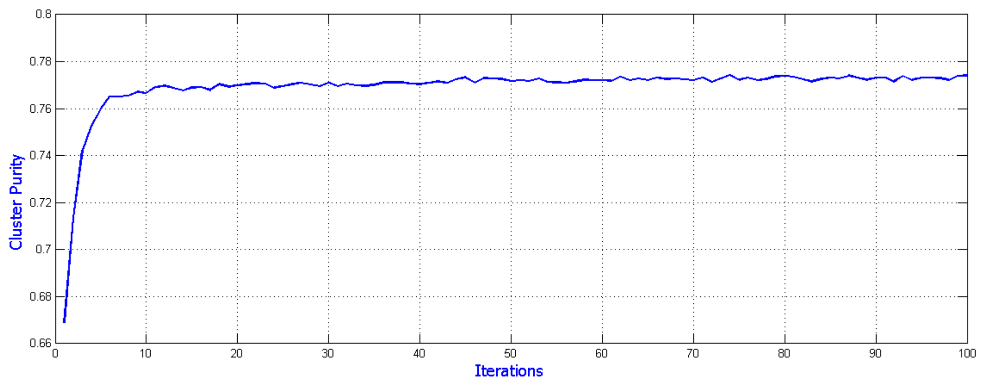


Figure 5.12: Cluster purity convergence.

Results

In this chapter, different simulation scenarios along with the corresponding results are presented. The simulations are divided into 3 major categories:

1. Known number of HMMs with presegmented data.
2. Unknown number of HMMs with presegmented data.
3. Unknown number of HMMs with unsegmented data.

6.1 Known number of HMMs with presegmented data

In these set of experiments, the variation of cluster purity will be shown with respect to following criteria:

1. Varying number of GMM mixture components per state.
2. Different initialisation strategies: Flat start and random start.
3. Algorithm used for learning HMM parameters.

6.1.1 Varying number of GMM components (N_m)

The cluster purity for different number of GMM components is shown in the table 6.1:

N_m	Cluster purity (%)
1	53.61
8	52.74
32	52.12
1 to 32	55.94

Table 6.1: Unsupervised case: Cluster purity values for varying number of GMM mixture components per state for all HMMs.

As can be seen from the table, varying the number of mixture components does not have any significant impact upon the cluster purity. The cluster purity values for $N_m = 1, 8, 32$ are very close to each other. However, when the number of mixtures is gradually increased using the process explained in chapter-5, the cluster purity obtained is better (55.94%) as compared to the cases where the number of components are fixed throughout the simulation. This means that a gradual increase in the model complexity for the GMMs seems to enhance the cluster purity and leads to better estimation of the GMMs.

The results for the supervised case are presented in table 6.2. Here, the cluster purity was evaluated for both the training as well as validation sequences i.e., after learning the HMM parameters using the labelled training sequences, the training as well as validation sequences are assigned cluster labels and the corresponding cluster purity is computed.

N_m	CP validation(%)	CP training(%)
1	65.09	65.61
8	74.09	76.23
32	77.12	83.83
1 to 32	77.34	83.55

Table 6.2: Supervised case: Cluster purity values for varying number of GMM mixture components per state for all HMMs.

As can be seen from the table 6.2, the cluster purity for training sequences is better than for the validation sequences. This is expected since the parameters are trained on the training sequences and represent these sequences well. However, the number of components makes a difference in this case. The cluster purity for $N_m = 8$ is better than cluster purity for $N_m = 1$. Cluster purity improves further when number of mixtures become 32. A gradual increase in the number of mixtures from 1 to 32 also shows a better performance as compared to cases with 1 and 8 components respectively. Thus, an increased model complexity for the GMMs leads to better estimation of the GMMs regardless of whether the complexity is increased gradually or stays same throughout the simulation.

6.1.2 Initialisation strategies

The cluster purity values (%) obtained using the flat start and random start initialisations are present for the unsupervised case in table 6.3 for $N_m = 8, 32$.

Initialisation	$N_m = 8$	$N_m = 32$
Flat	52.74	52.12
Random	50.08	47.97

Table 6.3: Cluster purity values for different initialisation approaches

As can be seen from the table, the flat start initialization shows an improved cluster purity regardless of the number of mixtures. For example, when $N_m = 8$, flat start initialization gives a cluster purity of 52.74% as compared to 50.08% when random initialisation is used. The difference becomes more remarkable with $N_m = 32$. There is a cluster purity improvement of more than 5% when flat start is used instead of random initialisation.

The better cluster purity results with flat start can be expected. Flat start takes into account the left-right structure of the HMM. First one-third of the samples of every sequence of an HMM are used to initialise attributes of GMM corresponding to state-1. Similarly, the second third of the sequence samples are used to initialise the attributes of state-2 GMM and the last third initialises the attributes of state-3 GMM. This emulates the fact that one can only go from a lower state to a higher state which is our model assumption. Thus, a flat start can be expected to give a good initialisation of the GMM attributes for each state and hence, a higher cluster purity performance.

6.1.3 Learning algorithm

As described in the last chapter, a variety of approaches could be used to learn the HMM parameters once segmentation and clustering steps have been performed. The different approaches used are: Gibbs sampling, Baum-Welch algorithm and a combination of Gibbs sampling and Baum-Welch algorithm wherein the half of the total number of iterations are performed using Gibbs sampling and the other half is done using Baum-Welch algorithm. In the experiments, out of a total of 300 iterations, first 150 iterations were performed using Gibbs sampling and the next 150 iterations used Baum-Welch algorithm. Table 6.4 shows the result of using these variations for an unsupervised case with $N_m = 8$. Flat start was used here since flat start gave better results as explained in the previous section.

Algorithm	Cluster purity (%)
Gibbs	52.74
Baum-Welch	53.39
Gibbs-BW	54.82

Table 6.4: Unsupervised case: Cluster purity values for different learning algorithms with $N_m = 8$.

The table shows that the performance of Gibbs sampling and Baum Welch algorithm is almost the same. The difference in cluster purities in these cases is less than 1%. On the other hand, the mixed approach employing both Gibbs and Baum-Welch algorithm gives better results as compared to the standalone approaches. A comparison of Gibbs and Baum-Welch approaches is also shown for the case where number of mixtures are increased from 1 to 32.

Algorithm	Cluster purity (%)
Gibbs	55.94
Baum-Welch	54.82
Gibbs-BW	55.32

Table 6.5: Unsupervised case: Cluster purity values for different learning algorithms with $N_m = 1$ to 32.

Here, the performance of Gibbs sampling and Baum-Welch algorithm is similar. The scenario with the mixed approach was run by first increasing the number of mixture components gradually while using Gibbs sampling approach. After 50 iterations had been run with $N_m = 32$ with Gibbs sampling, Baum-Welch algorithm was used to further tune the parameters with $N_m = 32$. Another observation is that when number of components increase from 1 to 32, better cluster purity is obtained as compared to the case with $N_m = 8$. This reiterates the fact that a gradual increase in the model complexity gives better cluster purity performance.

A comparison of the learning approaches can similarly be presented for the supervised case as shown in table 6.6 for $N_m = 8$.

Algorithm	Cluster purity training (%)	Cluster purity validation (%)
Gibbs	76.23	74.09
Baum-Welch	76.53	74.5

Table 6.6: Supervised case: Cluster purity values for different learning algorithms with $N_m = 8$.

As can be seen from the table, the performance of the learning algorithms is similar in supervised case as well for both the training as well as validation sequences. Further experiments were carried out by increasing the number of mixture components from 1 to 32 and the results obtained are summarised in table 6.7.

Algorithm	Cluster purity training (%)	Cluster purity validation (%)
Gibbs	83.55	77.34
Baum-Welch	84.88	77.64

Table 6.7: Supervised case: Cluster purity values for different learning algorithms with $N_m = 1$ to 32.

The performance of different learning algorithms is similar in supervised case also as can be seen from the values of cluster purity for the validation data which is 77.34% for Gibbs sampling and 77.64% for Baum-Welch algorithm.

6.2 Unknown number of HMMs with presegmented data

When the number of HMMs are unknown, Dirichlet process is used to sample HMM label for each sequence. Two experiments were carried out corresponding to this scenario: with $N_m = 8$ and N_m =1 to 32. The cluster purities, along with the number of HMMs discovered in each case are summarised in table 6.8.

N_m	K	Cluster purity (%)
8	173	60.91
1 to 32	110	64.04

Table 6.8: Cluster purity values for different number of GMM components for unknown number of components with presegmented data along with the number of HMMs recovered.

As can be seen from the table 6.8, lesser number of HMMs are discovered with $N_m = 1$ to 32 as compared to $N_m = 8$. This behaviour is expected since a gradual increase of the number of mixture components tends to give better results as explained in the previous sections. The true number of HMMs (i.e., phonemes) is 48 and hence the number 110 is closer to the true number as opposed to 173 HMMs which were discovered when the number of GMM mixture components were kept fixed at 8.

Figure 6.1 shows the confusion matrix for $N_m = 8$. The confusion matrix has been normalised per row so that the distribution of sequences across all the phoneme classes can be seen for each cluster. It can be observed that there is a correlation between the discovered cluster labels and the true labels each of which correspond to an English phoneme. For instance, clusters 22, 23 and 25 correspond to true phoneme number 7 which represents the phoneme /d/. These entries have been marked in the figure with a red circle. The figure also shows that one cluster label may be mapped to multiple phonemes. For example, cluster 100 gets mapped to phoneme label 9, 22 and 29. This can be seen from the figure where the lines are intersecting.

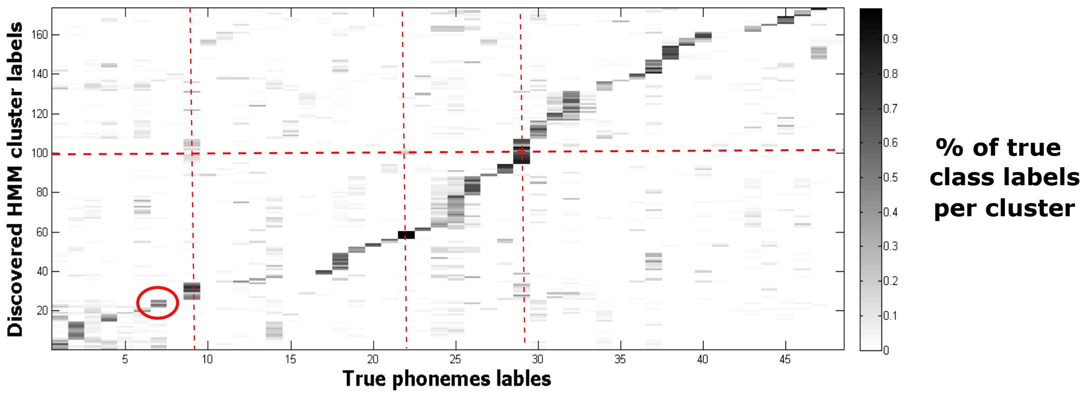


Figure 6.1: Confusion matrix for unknown number of HMMs with presegmented data and $N_m = 8$

The confusion matrix for $N_m = 1$ to 32 can be seen in figure 6.2 with similar patterns as were observed in the case of $N_m = 8$.

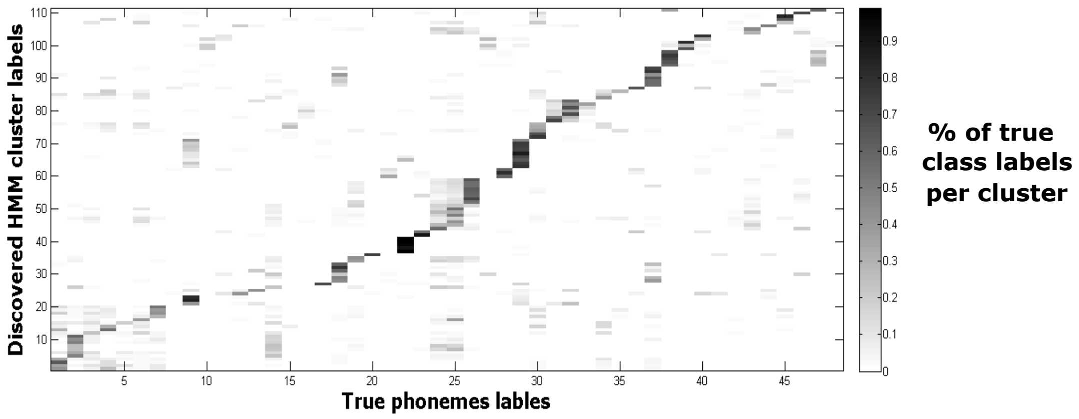


Figure 6.2: Confusion matrix for unknown number of HMMs with presegmented data and N_m =1 to 32.

6.3 Unknown number of HMMs with unsegmented data

In case of unsegmented data, in addition to cluster purity, the recall and percentage of over segmentation is also calculated.

Running the presegmentation step as explained in chapter-4, a recall of 41.38% was observed with an over-segmentation of 121.8%. The number of sequences obtained were 197594 (The original number being 159582).

Clustering and learning was done for 20 iterations using boundaries obtained with the presegmentation step. During these iterations, segmentation step was not performed so that the algorithm can train well using boundaries obtained after the presegmentation step. The cluster purity obtained after this step was 41.6% and a total of 160 HMMs were obtained. The algorithm was then run using all the 3 steps: segmentation, clustering and learning as described in chapter-4 and chapter-5. After the first segmentation step was performed, there was a drastic decrease in the number of sequences which became 97179. In the subsequent iterations, this number kept on decreasing and became 22210. Most of the sequences got mapped to the phoneme /hh/ (8135). The number of phonemes discovered were 72 and the cluster purity became 45.61%. The recall corresponding to this case was 6.51% with an over-segmentation value of 14.12% indicating that the data was under-segmented as can also be seen from the number of sequences obtained.

The self transition probabilities of state-3 (z_k) was high for all HMMs (> 0.8). Assuming this high probability was the reason for the increase in the length of sequences, a rescaling of z_k was done as explained in chapter-6. The rescaling factor F was set to 10. But this also did not stop the sequences from getting longer. The decrease in the number of sequences was drastic (97000) which is the same result that was obtained when z_k was kept intact.

Hence the segmentation performance could not be improved even with the use of rescaling.

Summary and Conclusion

This thesis presented different approaches to handle the tasks of segmentation, clustering and learning for an acoustic speech recognition system for the recognition of phonemes.

7.1 Summary

Hidden Markov models were used to model each phoneme. Chapter-2 discussed Hidden Markov models in detail along with the traditional expectation maximization approach called Baum-Welch algorithm that is used to learn the attributes of an HMM.

Markov chain Monte Carlo methods include a set of sampling methods that can be applied to learn the parameters of a model given the observations. Chapter-3 introduced some MCMC methods such as Metropolis-Hastings sampling and Gibbs sampling. Gibbs sampling was applied to learn the parameters of a bivariate Gaussian as a proof of concept which was also presented in chapter-3. Gibbs sampling involves sampling from conditional distributions of one random variable given all the other random variables in a multivariate setting. Bayesian inference can be applied to determine these conditional posterior distributions. This tends to give better estimates as compared to a maximum likelihood approach. To recover analytical expressions for the conditional posteriors, conjugate priors were used. These aspects were discussed in chapter-3.

Chapter-4 presented a description of the system modelling approach and included a detailed explanation of the way different algorithms were applied to the model. The task of segmentation was tackled by using Viterbi algorithm which was applied by cascading all the HMMs together to give a state space that contained all the states of all the HMMs. Every sample of an utterance was then assigned the state from which it may have originated. *Recall* and *over-segmentation* metrics were used to evaluate the effectiveness of this method. The task of clustering was dealt with by using Gibbs sampling. The HMM parameters were also learnt using Gibbs sampling.

The experimental set ups for various simulation scenarios were described in chapter-5. All the simulation results were presented in chapter-6.

7.2 Conclusion

The simulations were divided into 3 major categories: known number of HMMs with presegmented data, unknown number of HMMs with presegmented data and unknown number of HMMs with unsegmented data.

For the case with known number of HMMs, the HMM attributes could be initialised using one of the two approaches: Flat start and Random start. *Cluster purity*, which was chosen as the evaluation metric, had a higher value when HMM attributes were initialised using flat start.

A comparison between Gibbs sampling and Baum-Welch algorithm for learning of HMM attributes was also made. No difference in the performance was observed regardless of the learning algorithm used i.e., cluster purity values were similar with the use of Gibbs sampling and Baum-Welch algorithm. However, when a mixed learning approach, which involved using Gibbs sampling for the first half of the total number of iterations and Baum-Welch for the next half of the total number of iterations was used, higher cluster purity values were obtained. Similar conclusion was drawn for supervised case. Also, the cluster purity values were higher in case of supervised training as opposed to the unsupervised case, which was expected.

A comparison between the performance of the algorithm was also made for varying number of mixture components of the Gaussian mixture models that represented the observation densities of the HMM states. Varying the number of mixtures seemed to have no effect on the cluster purity in the unsupervised case when the number of mixtures were kept same in every iteration of the algorithm. However, when the number of components were gradually increased from 1 to 32 by splitting as discussed in chapter-5, better values for cluster purity were observed. In the supervised case, however, the cluster purity values became higher when more mixture components were used even if the number of components remained same for every iteration of the algorithm. For the case with unknown number of HMMs, gradual splitting of mixture components gave better performance as compared to cases where the number of components remained same throughout the duration of the simulation.

When the utterances were unsegmented, the segmentation step was carried out using the Viterbi algorithm. It was observed that a low recall for the boundaries were obtained. Also the utterances were under-segmented with the number of sequences much lower than the true number of sequences. This implies that the sequences got longer with iterations.

7.3 Future work

Future work for this thesis may involve using other presegmentation algorithms such as those described in [SL96], [AN]. Gibbs sampling based segmentation approaches like the one discussed in [LG12] could be used to improve performance in the case where data is unsegmented.

Also, instead of using Gaussian mixture models for modelling state observation probabilities of the HMMs, a deep neural network could be employed as discussed in [HDY$^+$12].

Appendix

1 Normal gamma distribution as a conjugate prior for Gaussian distribution

The Normal gamma distribution has been assumed as the joint distribution for mean and precision parameters of each GMM mixture component that models the observation densities of the states. It can be shown that given the data which is normally distributed, a normal gamma prior on the mean and precision is a conjugate prior for this Gaussian likelihood.

An observation x can be assumed to be a sample from a normal distribution with unknown mean μ and precision λ i.e.,

$$x \sim \mathcal{N}(\mu, \lambda^{-1})$$

A prior joint distribution can be assumed on the mean and precision of the above Gaussian distribution as:

$$(\mu, \lambda) \sim \mathrm{NG}(\mu_0, \kappa_0, \alpha_0, \beta_0)$$

This implies,

$$P(\mu, \lambda) \propto \lambda^{\alpha_0 - \frac{1}{2}} \exp[-\beta_0 \lambda] \exp[-\frac{\kappa_0 \lambda (\mu - \mu_0)^2}{2}] \tag{1}$$

For a set of n independent and identically distributed random variables (i.i.d) observations $\mathbf{X} = \{x_1, ..., x_n\}$, Baye's theorem could be applied to retrieve an expression for the posterior density of μ and λ given this set of observations i.e.,

$$P(\lambda, \mu | \mathbf{X}) \propto P(\mathbf{X} | \lambda, \mu) P(\lambda, \mu)$$

The term $P(\mathbf{X}|\lambda, \mu)$ represents the likelihood of the observation set given the mean and precision. Since the observations are i.i.d, the likelihood can be written as:

$$\begin{aligned} P(\mathbf{X}|\lambda, \mu) &= P(x_1, x_2, \cdots, x_n | \lambda, \mu) \\ &= \prod_{i=1}^{n} P(x_i | \lambda, \mu) \end{aligned}$$

Expanding the above term:

$$\begin{aligned}P(\mathbf{X}|\lambda,\mu) &\propto \prod_{i=1}^{n} \lambda^{1/2} \exp[\frac{-\lambda}{2}(x_i-\mu)^2] \\ &\propto \lambda^{n/2} \exp[\frac{-\lambda}{2}\sum_{i=1}^{n}(x_i-\mu)^2]\end{aligned}$$

Adding and subtracting the empirical mean, $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$, of the samples in the exponent term,

$$P(\mathbf{X}|\lambda,\mu) \propto \lambda^{n/2} \exp[\frac{-\lambda}{2}\sum_{i=1}^{n}(x_i-\bar{x}+\bar{x}-\mu)^2]$$

Representing the sample variance by $s = \frac{1}{n}\sum_{i=1}^{n}(x_i-\bar{x})^2$, the above expression can be written as:

$$\begin{aligned}P(\mathbf{X}|\lambda,\mu) &\propto \lambda^{n/2} \exp[\frac{-\lambda}{2}\sum_{i=1}^{n}\left((x_i-\bar{x})^2+(\bar{x}-\mu)^2\right)] \\ &\propto \lambda^{n/2} \exp[\frac{-\lambda}{2}\left(ns+n(\bar{x}-\mu)^2\right)] \qquad (2)\end{aligned}$$

Using equations 1 and 2, the posterior distribution of μ and λ given observation set $\mathbf{X}$ can now be obtained as:

$$\begin{aligned}P(\lambda,\mu|\mathbf{X}) &\propto P(\mathbf{X}|\lambda,\mu)P(\lambda,\mu) \\ &\propto \lambda^{n/2} \exp[\frac{-\lambda}{2}\left(ns+n(\bar{x}-\mu)^2\right)]\times \\ &\qquad \lambda^{\alpha_0-\frac{1}{2}} \exp[-\beta_0\lambda] \exp[-\frac{\kappa_0\lambda(\mu-\mu_0)^2}{2}] \\ \implies P(\lambda,\mu|\mathbf{X}) &\propto \lambda^{\frac{n}{2}+\alpha_0-\frac{1}{2}} \exp[-\lambda\left(\frac{1}{2}ns+\beta_0\right)]\times \\ &\qquad \exp\left[-\frac{\lambda}{2}\left(\kappa_0(\mu-\mu_0)^2+n(\bar{x}-\mu)^2\right)\right] \qquad (3)\end{aligned}$$

The term $\exp\left[-\frac{\lambda}{2}\left(\kappa_0(\mu-\mu_0)^2+n(\bar{x}\mu)^2\right)\right]$ can be simplified by completing the square as:

$$\begin{aligned}
\kappa_0(\mu-\mu_0)^2 + n(\bar{x}-\mu)^2 &= \kappa_0\mu^2 - 2\kappa_0\mu\mu_0 + \kappa_0\mu_0^2 + n\mu^2 - 2n\bar{x}\mu + n\bar{x}^2 \\
&= (\kappa_0+n)\mu^2 - 2(\kappa_0\mu_0 + n\bar{x})\mu + \kappa_0\mu_0^2 + n\bar{x}^2 \\
&= (\kappa_0+n)(\mu^2 - 2\frac{\kappa_0\mu_0+n\bar{x}}{\kappa_0+n}\mu) + \kappa_0\mu_0^2 + n\bar{x}^2 \\
&= (\kappa_0+n)\left(\mu - \frac{\kappa_0\mu_0+n\bar{x}}{\kappa_0+n}\right)^2 + \kappa_0\mu_0^2 + n\bar{x}^2 - \\
&\qquad \frac{(\kappa_0\mu_0+n\bar{x})^2}{\kappa_0+n} \\
&= (\kappa_0+n)\left(\mu - \frac{\kappa_0\mu_0+n\bar{x}}{\kappa_0+n}\right)^2 + \frac{\kappa_0 n(\bar{x}-\mu_0)^2}{\kappa_0+n}
\end{aligned}$$

Using this in equation 3, we get:

$$\begin{aligned}
P(\lambda,\mu|\mathbf{X}) &\propto \lambda^{\frac{n}{2}+\alpha_0-\frac{1}{2}} \exp\left[-\lambda\left(\frac{1}{2}ns+\beta_0\right)\right] \times \\
&\qquad \exp\left[-\frac{\lambda}{2}\left((\kappa_0+n)\left(\mu - \frac{\kappa_0\mu_0+n\bar{x}}{\kappa_0+n}\right)^2 + \frac{\kappa_0 n(\bar{x}-\mu_0)^2}{\kappa_0+n}\right)\right] \\
&\propto \lambda^{\frac{n}{2}+\alpha_0-\frac{1}{2}} \exp\left[-\lambda\left(\frac{1}{2}ns+\beta_0+\frac{\kappa_0 n(x-\mu_0)^2}{2(\kappa_0+n)}\right)\right] \times \\
&\qquad \exp\left[-\frac{\lambda}{2}(\kappa_0+n)\left(\mu - \frac{\kappa_0\mu_0+n\bar{x}}{\kappa_0+n}\right)^2\right]
\end{aligned}$$

Comparing the above expression with expression in equation 1, it can be seen that the posterior is also a Normal-gamma distribution:

$$P(\lambda,\mu|\mathbf{X}) = \mathrm{NG}\left(\frac{\kappa_0\mu_0+n\bar{x}}{\kappa_0+n}, \kappa_0+n, \alpha_0+\frac{n}{2}, \beta_0+\frac{1}{2}\left(ns+\frac{\kappa_0 n(\bar{x}-\mu_0)^2}{\kappa_0+n}\right)\right)$$

2 Dirichlet distribution as the conjugate prior for multinomial distribution

Multinomial distribution has been used to model the likelihood for transitions as well as mixture labels of the samples assigned to a particular GMM representing a state. It can be shown that Dirichlet distribution serves as a conjugate prior for the multinomial likelihood.

Consider the random variables $p = \{p_1, p_2, \cdots, p_K\}$ distributed according to the Dirichlet distribution with hyperparameters denoted by α i.e.,

$$p \sim Dir(\alpha_1, \alpha_2, \cdots, \alpha_K)$$

i.e.,

$$P(p_1, p_2, \cdots, p_K; \alpha_1, \alpha_2, \cdots, \alpha_K) \propto \prod_{i=1}^{K} p_i^{\alpha_i - 1} \tag{4}$$

Also, let a set of random variables $\mathbf{X} = \{x_1, x_2, \cdots, x_K\}$ be distributed according to a multinomial distribution with hyperparameters $p = \{p_1, p_2, \cdots, p_K\}$:

$$P(x_1, x_2, \cdots, x_K; p_1, p_2, \cdots, p_K) \propto \prod_{i=1}^{K} p_i^{x_i} \tag{5}$$

If $\mathbf{D}$ is the set of all observations where each observation comes from one of the x_i classes, the posterior can now be evaluated using equation 4 as likelihood and equation 5 as prior distribution:

$$\begin{aligned} P(p_1, p_2, \cdots, p_K | \mathbf{D}) &\propto \prod_{i=1}^{K} p_i^{\alpha_i - 1} \times \prod_{i=1}^{K} p_i^{x_i^j} \\ &\propto \prod_{i=1}^{K} p_i^{\alpha_i - 1 + x_i} \end{aligned}$$

where x_i denotes the total number of samples in the i-th class. The above expression corresponds to a Dirichlet distribution with hyperparameters as:

$$\alpha_i^{'} = \alpha_i + x_i$$

Bibliography

[AEEM01] Guido Aversano, Anna Esposito, Antonietta Esposito, and Maria Marinaro. A new text-independent method for phoneme segmentation. In *Proc the 44th IEEE Midwest Symposium on Circuits and Systems*, volume 2, pages 516–519, 2001.

[AN] Stephanie Antetomaso and MA Norton. Unsupervised phoneme segmentation in continuous speech.

[Bes74] Julian Besag. Spatial interaction and the statistical analysis of lattice systems. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 192–236, 1974.

[Bis06] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[CG92] George Casella and Edward I George. Explaining the gibbs sampler. *The American Statistician*, 46(3):167–174, 1992.

[CL11] Chun-An Chan and Lin-Shan Lee. Unsupervised hidden markov modeling of spoken queries for spoken term detection without speech recognition. In *Twelfth Annual Conference of the International Speech Communication Association*, 2011.

[DM80] Steven B Davis and Paul Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 28(4):357–366, 1980.

[Eim75] Peter D Eimas. Auditory and phonetic coding of the cues for speech: Discrimination of the [rl] distinction by young infants. *Perception & Psychophysics*, 18(5):341–347, 1975.

[Fel08] Willliam Feller. *An introduction to probability theory and its applications*, volume 2. John Wiley & Sons, 2008.

[Fry77] Dennis Fry. *Homo loquens: Man as a talking animal*. CUP Archive, 1977.

[GC$^+$93] John S Garofolo, Linguistic Data Consortium, et al. *TIMIT: acoustic-phonetic continuous speech corpus*. Linguistic Data Consortium, 1993.

[GG84] Stuart Geman and Donald Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (6):721–741, 1984.

[GG06] Alvin Garcia and Herbert Gish. Keyword spotting of arbitrary words using minimal speech resources. In *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*, volume 1, pages I–I. IEEE, 2006.

[GS90] Alan E Gelfand and Adrian FM Smith. Sampling-based approaches to calculating marginal densities. *Journal of the American statistical association*, 85(410):398–409, 1990.

[Has70] W Keith Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.

[HDY$^+$12] Geoffrey Hinton, Li Deng, Dong Yu, George E. Dahl, Abdel rahman Mohamed, Navdeep Jaitlyp, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, and Tara Sainath. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *Signal Processing Magazine, IEEE*, 29(6):82–97, 2012.

[HW79] John A Hartigan and Manchek A Wong. Algorithm as 136: A k-means clustering algorithm. *Applied statistics*, pages 100–108, 1979.

[JR91] Biing Hwang Juang and Laurence R Rabiner. Hidden markov models for speech recognition. *Technometrics*, 33(3):251–272, 1991.

[KS04] Hyoung-Gook Kim and Thomas Sikora. Comparison of mpeg-7 audio spectrum projection features and mfcc applied to speaker recognition, sound classification and audio segmentation. In *Acoustics, Speech, and Signal Processing, 2004. Proceedings.(ICASSP'04). IEEE International Conference on*, volume 5, pages V–925. IEEE, 2004.

[KSH$^+$06] Patricia K. Kuhl, Erica Stevens, Akiko Hayashi, Toshisada Deguchi, Shigeru Kiritani, and Paul Iverson. Infants show a facilitation effect for native language phonetic perception between 6 and 12 months. *Developmental Science*, 9(2):F13–F21, 2006.

[Kuh04] Patricia K Kuhl. Early language acquisition: cracking the speech code. *Nat Rev Neurosci*, 5(11):831–843, nov 2004.

[LCSSK67] Alvin M Liberman, Franklin S Cooper, Donald P Shankweiler, and Michael Studdert-Kennedy. Perception of the speech code. *Psychological review*, 74(6):431, 1967.

[LG12] Chia-ying Lee and James Glass. A Nonparametric Bayesian Approach to Acoustic Model Discovery. *Acl*, (July):40–49, 2012.

[LH89] Kai-Fu Lee and Hsiao-Wuen Hon. Speaker-independent phone recognition using hidden markov models. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 37(11):1641–1648, 1989.

[LSJ88] Chin-Hui Lee, Frank K Soong, and Biing-Hwang Juang. A segment model based approach to speech recognition. In *Acoustics, Speech, and Signal*

Processing, 1988. ICASSP-88., 1988 International Conference on, pages 501–541. IEEE, 1988.

[MGT03] I Dan Melamed, Ryan Green, and Joseph P Turian. Precision and recall of machine translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology: companion volume of the Proceedings of HLT-NAACL 2003–short papers-Volume 2*, pages 61–63. Association for Computational Linguistics, 2003.

[MKLB98] Håkan Melin, Johan W Koolwaaij, Johan Lindberg, and Frédéric Bimbot. A comparative evaluation of variance flooring techniques in hmm-based speaker verification. 1998.

[ML06] Morris L Marx and Richard J Larsen. *Introduction to mathematical statistics and its applications*. Pearson/Prentice Hall, 2006.

[MN03] Hugo Meinedo and Joao Neto. Audio segmentation, classification and clustering in a broadcast news task. In *Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03). 2003 IEEE International Conference on*, volume 2, pages II–5. IEEE, 2003.

[Mob00] Bijan G Mobasseri. Digital modulation classification using constellation shape. *Signal processing*, 80(2):251–277, 2000.

[MRR$^+$53] Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6):1087–1092, 1953.

[MU49] Nicholas Metropolis and Stanislaw Ulam. The monte carlo method. *Journal of the American statistical association*, 44(247):335–341, 1949.

[Nea12] Radford M Neal. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media, 2012.

[Oha74] John J Ohala. Phonetic explanation in phonology. *parasession on natural phonology*, pages 251–74, 1974.

[OT01] Aude Oliva and Antonio Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *International journal of computer vision*, 42(3):145–175, 2001.

[Pet06] Slobodan Petrovic. A comparison between the silhouette index and the davies-bouldin index in labelling ids clusters. In *Proceedings of the 11th Nordic Workshop of Secure IT Systems*, pages 53–64, 2006.

[Rab89] Lawrence R Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[RV96] Raimundo Real and Juan M Vargas. The probabilistic basis of jaccard's index of similarity. *Systematic biology*, pages 380–385, 1996.

[SL96] Youngjoo Suh and Youngjik Lee. Phoneme segmentation of continuous speech using multi-layer perceptron. In *Spoken Language, 1996. ICSLP 96. Proceedings., Fourth International Conference on*, volume 3, pages 1297–1300. IEEE, 1996.

[Ste00] Kenneth N Stevens. *Acoustic phonetics*, volume 30. MIT press, 2000.

[Tow71] John T Townsend. Theoretical analysis of an alphabetic confusion matrix. *Perception & Psychophysics*, 9(1):40–50, 1971.

List of Figures

List of Tables