3 Statistical models in speech and language processing

This chapter focuses on basic statistical models (Gaussian mixture models (GMM), hidden Markov models (HMM), n-gram models and latent topic models), which are widely used in speech and language processing. These are well-known generative models, and these probabilistic models can generate speech and language features based on their likelihood functions. We also provide parameter-learning schemes based on maximum likelihood (ML) estimation which is derived according to the expectation and maximization (EM) algorithm (Dempster *et al*. 1976). Basically, the following chapters extend these statistical models from ML schemes to Bayesian schemes. These models are fundamental for speech and language processing. We specifically build an automatic speech recognition (ASR) system based on these models and extend them to deal with different problems in speaker clustering, speech verification, speech separation and other natural language processing systems.

In this chapter, Section 3.1 first introduces the probabilistic approach to ASR, which aims to find the most likely word sequence W corresponding to the input speech feature vectors $\mathbf{O}$. Bayes decision theory provides a theoretical solution to build up a speech recognition system based on the posterior distribution of the word sequence $p(W|\mathbf{O})$ given speech feature vectors $\mathbf{O}$. Then the Bayes theorem decomposes the problem based on $p(W|\mathbf{O})$ into two problems based on two generative models of speech features $p(\mathbf{O}|W)$ (acoustic model) and language features $p(W)$ (language model), respectively. Therefore, the Bayes theorem changes the original problem to these two independent generative model problems.

Next, Section 3.2 introduces the HMM with the corresponding likelihood function as a generative model of speech features. The section first describes the discrete HMM, which has a multinomial distribution as a state observation distribution, and Section 3.2.4 introduces the GMM as a state observation distribution of the continuous density HMM for acoustic modeling. The GMM by itself is also used as a powerful statistical model for other speech processing approaches in the later chapters. Section 3.3 provides the basic algorithms of forward–backward and Viterbi algorithms. In Section 3.4, ML estimation of HMM parameters is derived according to the EM algorithm to deal with latent variables included in the HMM efficiently. Thus, we provide the conventional ML treatment of basic statistical models for acoustic models based on the HMM.

From Section 3.6, we go on to describe statistical language models as a generative model of language features. As a standard language model, we introduce n-gram models. Similarly to the HMM parameters, the n-gram parameters are also calculated

https://doi.org/10.1017/CBO9781107295360.004 Published online by Cambridge University Press

by using the ML estimation. However, ML solutions to n-gram parameters are easily overestimated due to the intrinsic sparse data problems in natural languages. Therefore, the section also describes conventional (rather heuristic) smoothing techniques. Some of the smoothing techniques introduced here are revisited in later chapters to be interpreted as the Bayesian approach, where a Bayesian principle provides these smoothing techniques to regularize these models. In addition, it is well known that the number of n-gram parameters is exponentially increased with large n, which makes it impossible to model a whole document structure within the n-gram model.

Section 3.7 provides another generative model of language features, called the latent topic model, which deals with a statistical treatment of a document model. The section also discusses a way of combining such document models and n-gram models.

Finally, following the discussions of statistical acoustic and language models, Section 3.8 provides an example of applying the Bayesian approach to ASR, as a case study. The section provides an exact Bayesian manner of formulating the standard statistical model (HMM) in ASR, and introduces the posterior distributions of the variables used in acoustic models. The section points out the problem arising mainly due to the posterior distributions, which can be solved in the later chapters.

3.1 Bayes decision for speech recognition

This section describes a statistical speech recognition framework as an example of speech and language processing based on the Bayes decision theory. Before the Bayes decision discussion, we first introduce general speech recognition briefly.

Automatic speech recognition aims to extract helpful text information from speech signals, where both speech and text are represented by *sequential patterns*. These patterns correspond to the time-series signals which are observed in sequences of random variables. Speech recognition is processed in a temporal domain. There are some other technical data, e.g., music signal, video signal, text document, seismic signal, gene sequence, EEG signal, ECG signal, financial data, which are also collected in a time domain. The sequential pattern property is unique and different from data analysis in a spatial domain for general image processing and spectral signal processing. In general, speech and text data are driven under some specialized stochastic process and probabilistic model, e.g., HMMs (Rabiner & Juang 1986) are used to represent speech signals and n-gram models are used to characterize word sequences. In front-end processing, speech signals are first chunked into different time frames t with frame length 25 ms and then transformed to a sequence of speech feature vectors by using mel-frequency cepstral coefficients (MFCCs) (Davis & Mermelstein 1980) or perceptual linear prediction (PLP) coefficients (Hermansky 1990). Each feature vector $\mathbf{o}_t$ is regarded as a random vector consisting of entries with *continuous* value. However, the nth word w_n in a word sequence is a discrete value or label among all words in a dictionary $\mathcal{V}$ with vocabulary size $|\mathcal{V}|$. Thus, speech recognition, which involves acoustic and language models, handles the modeling of *continuous* data as well as *discrete* data.

https://doi.org/10.1017/CBO9781107295360.004 Published online by Cambridge University Press

Now we provide a mathematical notation for a speech recognition problem, which recognizes a speech utterance and outputs the corresponding word sequence in the utterance. Let $\mathbf{o}_t \in \mathbb{R}^D$ be a D dimensional feature vector at frame t, and $\mathbf{O} = \{\mathbf{o}_t | t = 1, \cdots, T\}$ be a speech feature sequence for T frames of one utterance. The number of dimensions (D) is usually 39, which consists of 12 dimensional MFCCs + log power, with delta and delta delta coefficients (Furui 1981). On the other hand, the corresponding word sequence is represented by $W = w_1^N = \{w_n | n = 1, \cdots, N\}$. Here, $w_n \in \mathcal{V}$ is the nth word in this word sequence with N words. The continuous-valued speech feature sequence $\mathbf{O}$ and the discrete-valued word sequence W are sequential patterns in an automatic speech recognition system. Based on the mathematical notations of the speech feature and word sequences, the Bayes decision theory is introduced to find a decision rule or *mapping function* $d(\cdot)$ which maps an input feature sequence $\mathbf{O}$ into an output word sequence W by

$$W = d(\mathbf{O}). \tag{3.1}$$

A popular decision rule is designed to find the most likely word sequence $\hat{W}$ corresponding to input feature sequence $\mathbf{O}$ based on the maximum a-posteriori (MAP) decision rule,

$$\hat{W} = d_{\text{MAP}}(\mathbf{O}) \triangleq \arg\max_W \; p(W|\mathbf{O}), \tag{3.2}$$

where $p(W|\mathbf{O})$ is the posterior distribution of W given $\mathbf{O}$. The posterior distribution is often rewritten as

$$\begin{aligned} \hat{W} = d_{\text{MAP}}(\mathbf{O}) &= \arg\max_W \; \frac{p(\mathbf{O}|W)p(W)}{p(\mathbf{O})} \\ &= \arg\max_W \; \underbrace{p(\mathbf{O}|W)}_{\text{acoustic model}} \times \underbrace{p(W)}_{\text{language model}}. \end{aligned} \tag{3.3}$$

The probabilistic product rule decomposes the posterior distribution into likelihood function $p(\mathbf{O}|W)$ and prior probability $p(W)$ based on acoustic model and language model, respectively. This is a well-known process, called the noisy channel model, that can deal with a speech recognition problem based on acoustic and language models. The same scheme of decomposition of this decision rule is widely used for other speech and language processing including machine translation (Brown, Cocke, Pietra *et al.* 1990, Brants, Popat, Xu *et al*. 2007), spell correction (Brill & Moore 2000), and voice conversion (Saito, Watanabe, Nakamura *et al*. 2012).

However, it is more general to follow a Bayesian perspective for pattern recognition and fulfil an optimal Bayes decision to estimate the decision rule $\hat{d}(\mathbf{O})$ of an input sentence $\mathbf{O}$ by minimizing the expected loss function or Bayes risk, which is defined by Lee & Huo (2000) as a functional of the decision rule:

$$\begin{aligned} r[d] &\triangleq \mathbb{E}_{(W,\mathbf{O})}[\ell(W, d(\mathbf{O}))] \\ &= \sum_W \int \ell(W, d(\mathbf{O})) p(W, \mathbf{O}) d\mathbf{O} \end{aligned}$$

https://doi.org/10.1017/CBO9781107295360.004 Published online by Cambridge University Press

$$= \int p(\mathbf{O}) \left(\sum_{W} \ell(W, d(\mathbf{O})) p(W|\mathbf{O}) \right) d\mathbf{O}$$
$$= \sum_{W} p(W) \int \ell(W, d(\mathbf{O})) p(\mathbf{O}|W) d\mathbf{O}, \tag{3.4}$$

where $\mathbb{E}_{(W,\mathbf{O})}[\cdot]$ denotes the expectation function over joint distribution $p(W, \mathbf{O})$. For the later derivations, Eq. (3.4) provides the two equivalent equations in the third and fourth lines by using the product rule. The loss function satisfies this property:

$$0 \leq \ell(W, d(\mathbf{O}) = W) \leq \ell(W, d(\mathbf{O}) \neq W), \tag{3.5}$$

meaning that the loss due to misclassification $d(\mathbf{O}) \neq W$ is larger than or equal to the loss without misclassification $d(\mathbf{O}) = W$. Therefore, the optimal decision rule $\hat{d}(\mathbf{O})$ can be obtained by minimizing the Bayes risk, which corresponds to minimizing the expected loss function.

Bayes risk is expanded into two expressions, which are shown in the third and fourth equations in the right-hand-side of Eq. (3.4). Following the third equation in Eq. (3.4), we find that Bayes decision rule is equivalent to dealing with a minimization problem:

$$\min_{d \in \mathcal{D}} r[d] = \min_{d \in \mathcal{D}} \int p(\mathbf{O}) \left(\sum_{W} \ell(W, d(\mathbf{O})) p(W|\mathbf{O}) \right) d\mathbf{O}. \tag{3.6}$$

Here $\mathcal{D}$ denotes a set of all possible decision functions. This optimization can be solved by minimizing the expression in the brackets in the above equation, since the decision rule function does not depend on $\mathbf{O}$ in general. This minimization is satisfied by considering the following optimal decision rule given any $\mathbf{O}$:

$$\hat{d}(\mathbf{O}) = \arg \min_{d(\mathbf{O}) \in \mathcal{D}} \sum_{W} \ell(W, d(\mathbf{O})) p(W|\mathbf{O})$$
$$= \arg \min_{d(\mathbf{O}) \in \mathcal{D}} \mathbb{E}_{(W)}[\ell(W, d(\mathbf{O}))|\mathbf{O}]. \tag{3.7}$$

That is, finding the optimal decision rule $\hat{d}(\mathbf{O})$ in terms of minimizing the expected loss (in Eq. (3.4)) is equivalent to finding the optimal decision rule function in terms of the expected loss function given $\mathbf{O}$.

In Goel & Byrne (2000) and Chien, Huang, Shinoda *et al*. (2006), a minimum Bayes risk (MBR) classification was proposed to fulfil an optimal Bayes decision in Eq. (3.7) for automatic speech recognition by using a predefined loss function. In Goel & Byrne (2000), the word error rate (WER) loss function $\ell_{\text{WER}}(W, \mathbf{O})$ was calculated using Levenshtein distance between word sequence hypotheses. This function was used to build an MBR decision rule:

$$d_{\text{MBR}}(\cdot) = \arg \min_{d(\cdot)} \sum_{W} p(W)$$
$$\times \int \ell_{\text{WER}}(W, d(\mathbf{O})) p(\mathbf{O}|W) d\mathbf{O}, \tag{3.8}$$

which is derived according to the fourth equation in the right-hand-side of Eq. (3.4).

https://doi.org/10.1017/CBO9781107295360.004 Published online by Cambridge University Press

More popularly, a meaningful loss function for speech recognition is simply specified by a so-called zero-one loss function which treats misclassification of each observation sample $\mathbf{O}$ equally, namely by using

$$\ell_{01}(W, d(\mathbf{O})) = \begin{cases} 0 & \text{if correctly classified or } d(\mathbf{O}) = W \\ 1 & \text{if wrongly classified or } d(\mathbf{O}) \neq W. \end{cases} \tag{3.9}$$

Substituting Eq. (3.9) into the fourth line in Eq. (3.4) leads to zero-one Bayes risk:

$$\begin{aligned} r_{01}[d] &= \sum_W p(W) \int_{d(\mathbf{O}) \neq W} \ell_{01}(W, d(\mathbf{O})) p(\mathbf{O}|W) d\mathbf{O} \\ &\quad + \sum_W p(W) \int_{d(\mathbf{O}) = W} \ell_{01}(W, d(\mathbf{O})) p(\mathbf{O}|W) d\mathbf{O} \\ &= \sum_W p(W) \int_{d(\mathbf{O}) \neq W} p(\mathbf{O}|W) d\mathbf{O}. \end{aligned} \tag{3.10}$$

In the third line of Eq. (3.10), the expectation operation using zero-one loss function $r_{01}(d(\cdot))$ is calculated over all observations which are wrongly classified, i.e., $d(\mathbf{O}) \neq W$. This loss function corresponds to unconditional error probability, which is reasonable to act as a measure of goodness of the decision rule for speech recognition.

In addition, Eq. (3.10) is further rewritten as

$$\begin{aligned} r_{01}[d] &= \sum_W p(W) \left(1 - \int_{d(\mathbf{O}) = W} p(\mathbf{O}|W) d\mathbf{O} \right) \\ &= 1 - \sum_W \int_{d(\mathbf{O}) = W} p(W) p(\mathbf{O}|W) d\mathbf{O} \end{aligned} \tag{3.11}$$

by using the following properties:

$$\sum_W p(W) = 1, \tag{3.12}$$

$$\int p(\mathbf{O}|W) d\mathbf{O} = 1. \tag{3.13}$$

The resulting decision rule $d_{01}(\cdot)$ follows the minimum classification error criterion which leads to the MAP decision rule as addressed in Eq. (3.2), i.e.,

$$\hat{W} = d_{01}(\mathbf{O}) = d_{\text{MAP}}(\mathbf{O}) = \arg\max_W \; p(W|\mathbf{O}). \tag{3.14}$$

The most likely word sequence $\hat{W}$ is found so as to achieve the highest posterior probability for correctly classified observation sequence $d(\mathbf{O}) = W$.

Using an MAP decision rule, the probability measure $p(\mathbf{O}|W)$ calculates how likely the acoustic observation sequence $\mathbf{O}$ is, based on the word sequence hypothesis W. We also name $p(\mathbf{O}|W)$ as the acoustic likelihood function. There are many kinds of acoustic models which are assumed in calculation of the statistical model $p_\Theta(\mathbf{O}|W)$ based on a set of acoustic parameters Θ. In this chapter, the hidden Markov model (HMM) is considered for acoustic modeling and the HMM parameters Θ are plugged into the

https://doi.org/10.1017/CBO9781107295360.004 Published online by Cambridge University Press

probability measure estimator $\hat{p}_\Theta(\mathbf{O}|W)$. On the other hand, the probability measure $p(W)$ is defined as the prior probability of word sequence W. This measure calculates the joint probability for a sequence of words based on a set of multinomial parameters or n-gram parameters Θ. The plug-in language model $\hat{p}_\Theta(W)$ is determined as a language model estimator. Here, acoustic parameters and linguistic parameters are both included in model parameters Θ. For an automatic speech recognition (ASR) system, we will estimate the acoustic parameters and the linguistic parameters from a set of training utterances $\mathbf{O}$ and their word transcriptions W according to the Maximum Likelihood (ML) estimation. We assume that these plug-in models $\{\hat{p}_\Theta(\mathbf{O}|W), \hat{p}_\Theta(W)\}$ given ML parameters Θ are true. The prediction of new test utterance $\mathbf{O}$ based on the estimated MAP decision rule $\hat{d}_{\text{MAP}}(\mathbf{O})$ is performed by

$$\begin{aligned}\hat{d}_{\text{MAP}}(\mathbf{O}) &= \arg\max_W \ \hat{p}(W|\mathbf{O}) \\ &= \arg\max_W \ \hat{p}_\Theta(\mathbf{O}|W)\hat{p}_\Theta(W). \end{aligned} \tag{3.15}$$

However, the point estimates of the ML-based acoustic model and language model Θ from the given observation space Ω_o using the collected training data may not generalize well for the unknown test data outside the training space Ω_o. The distributions $p(\mathbf{O}|W)$ and $p(W)$ may not be correctly assumed or may be over-trained or under-trained. From a Bayesian perspective, these issues could be tackled by treating acoustic parameters and linguistic parameters Θ as random variables. Consideration of these uncertainties is helpful for recognition of new test data. For this consideration, the expected loss function in Eq. (3.4) is calculated by additionally marginalizing over continuous parameters Θ:

$$r_{\text{BPC}}(d(\cdot)) = \mathbb{E}_{(W,\mathbf{O},\Theta)}[\ell(W, \mathbf{O}, \Theta)]. \tag{3.16}$$

The Bayesian predictive classification (BPC) rule,

$$d_{\text{BPC}}(\cdot) = \arg\min_{d(\cdot)\in\Omega_d} r_{\text{BPC}}(d(\cdot)), \tag{3.17}$$

(Jiang, Hirose & Huo 1999, Huo & Lee 2000, Lee & Huo 2000, Chien & Liao 2001) was proposed to establish a robust decision rule for unknown test speech. The zero-one loss function $\ell_{01}(\cdot)$ was applied in previous studies on the BPC rule. Details of BPC-based speech recognition will be addressed in Section 6.3.

In addition, Bayes decision theory was developed to estimate the *discriminative* acoustic model for speech recognition (Juang & Katagiri 1992). The idea is to minimize the expected loss function based on the logistic sigmoid function,

$$\ell(d_k(\mathbf{O}, \Theta)) = \frac{1}{1 + \exp^{-\alpha d_k(\mathbf{O},\Theta)}}, \tag{3.18}$$

which is a function of misclassification measure defined as

$$\begin{aligned} d_k(\mathbf{O}, \Theta) =& -g_k(\mathbf{O}; \Theta) \\ &+ \log\left(\frac{1}{K-1}\sum_{j,j\neq k}\exp(g_j(\mathbf{O}; \Theta))\right). \end{aligned} \tag{3.19}$$

https://doi.org/10.1017/CBO9781107295360.004 Published online by Cambridge University Press

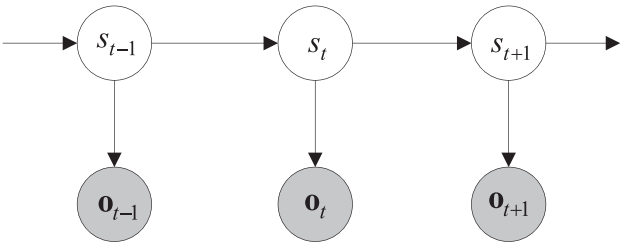

Figure 3.1 Graphical model of HMM without model parameters.

In Eq. (3.18), α is a tuning parameter and the misclassification measure of phone class C_k from training data $\mathbf{O} = \{\mathbf{o}_t\}$ is calculated by measuring the distance between the discriminant functions $g_k(\mathbf{O}, \Theta)$ of the target phone $\mathbf{o}_t \in C_k$ and its competing phones $\mathbf{o}_t \notin C_k$. The discriminant function of competing phones is averaged over all phones except the target phone. The discriminative acoustic model was trained according to the minimum classification error (MCE) criterion which is closely related to the minimum Bayes risk for optimal Bayes decision (Juang & Katagiri 1992).

3.2 Hidden Markov model

The previous section describes the Bayes decision theory and introduces acoustic model $p(\mathbf{O}|W)$ and language models $p(W)$. This section describes hidden Markov models (HMMs) (Rabiner & Juang 1986) (Figure 3.1) as a standard statistical acoustic model in detail. Before describing the HMM in detail, we first explain what HMM represents in speech recognition.

3.2.1 Lexical unit for HMM

The acoustic model $p(\mathbf{O}|W)$ means that we provide a likelihood function of the observations $\mathbf{O}$ given the word sequence W. However, since the number of all possible word sequences is an exponential order, we cannot prepare a likelihood function for each word sequence. Instead, we first introduce a lexical sequence $L = \{l_m \in \mathcal{L} | m = 1, \cdots, M\}$ that is composed of phonemes (e.g., /a/,/k/), context-dependent phonemes (e.g., /a/-/k/-/i/,/a/-/k/-/a/), or words (e.g., numbers, commands) as l_m. A discussion about the context-dependent phoneme (allophone) unit can be found in Section 6.5. We usually use a phoneme unit defined by linguistics, and the automatic discovery of the phoneme unit from speech data by using Bayesian nonparametrics can be found in Section 8.4. The definition of the lexical unit depends on application, but the standard acoustic model for LVCSR uses (context-dependent) phonemes as a lexical unit, and hereinafter in this chapter, we use phonemes to define a lexical unit. Then, $\mathcal{L}$ is a set of all distinct phonemes.

https://doi.org/10.1017/CBO9781107295360.004 Published online by Cambridge University Press

By using the lexical sequence L, we can revisit the MAP decision rule in Eq. (3.15) for ASR as follows:

$$\begin{aligned} d_{\mathrm{MAP}}(\mathbf{O}) &= \arg\max_{W} p(W|\mathbf{O}) \\ &= \arg\max_{W} p(\mathbf{O}|W)p(W) \\ &= \arg\max_{W} \sum_{L} p(\mathbf{O}, L|W)p(W). \end{aligned} \tag{3.20}$$

By using the product rule, and assuming that the likelihood function only depends on the lexical sequence L, it is rewritten as:

$$\begin{aligned} d_{\mathrm{MAP}}(\mathbf{O}) &= \arg\max_{W} \sum_{L} p(\mathbf{O}|L, W)p(L|W)p(W) \\ &\approx \arg\max_{W} \sum_{L} p(\mathbf{O}|L)p(L|W)p(W), \end{aligned} \tag{3.21}$$

where $p(L|W)$ is called a lexical model. Usually, the lexical model is not a probabilistic model, but is obtained deterministically by using a lexical dictionary, which provides a phoneme sequence (or multiple phoneme sequences) given a word. We further assume that the alignment of $\mathbf{O}$ for phoneme l_m is already given. This means that $\mathbf{O}$ is segmented to $\mathbf{O}_m = \{\mathbf{o}_{t_{m-1}+1}, \cdots, \mathbf{o}_{t_m}\}$ where $t_0 = 0, t_M = T$, and $\{t_m\}_{m=1}^{M-1}$ is given.[1]

By assuming that $\mathbf{O}_m$ is independent and identically distributed (iid) for phoneme l_m, the acoustic model is factorized by m as:

$$p(\mathbf{O}|L) = \prod_{m=1}^{M} p(\mathbf{O}_m|l_m). \tag{3.22}$$

This $p(\mathbf{O}_m|l_m)$ is an actual likelihood function that we deal with for ASR, and is represented by an HMM. Therefore, the acoustic model is composed of $|\mathcal{L}|$ HMMs where $|\mathcal{L}|$ denotes the number of distinct phonemes. The following section explains the HMM for one phoneme, and omits phoneme l_m in the explanation. In addition, since the alignment is already given, we omit the segmentation information m based on t_m, and use $\mathbf{O} = \{\mathbf{o}_t \in \mathbb{R}^D | t = 1, \cdots, T\}$ instead of $\mathbf{O}_m$ to be modeled by an HMM.

3.2.2 Likelihood function of HMM

This section describes a likelihood function of the HMM for a phoneme, where the HMM is a popular formalism for representation of sequential patterns. The likelihood function defined here is used to estimate HMM parameters.

In a set of D dimensional continuous-valued speech feature vectors $\mathbf{O}$, each observation $\mathbf{o}_t$ is represented under a Markov state s_t as illustrated in Figure 3.1. We assume

[1] In the actual search ($\arg\max_W$) process in Eq. (3.21), this alignment information is not fixed, and is searched at the same time. Many techniques have been developed to efficiently search this huge space considering the lexicon and word sequences (e.g., Ney, Haeb-Umbach, Tran *et al.* (1992) and weighted finite state transducer (WFST) based techniques (Mohri, Pereira & Riley 2002, Hori & Nakamura 2013)).

https://doi.org/10.1017/CBO9781107295360.004 Published online by Cambridge University Press

that each speech frame $\mathbf{o}_t$ is independent conditionally on its state label s_t. Starting from the state at the first time frame with an initial state probability π_{s_1}, which is a model parameter of HMM,

$$p(s_1) \triangleq \pi_{s_1}, \tag{3.23}$$

where each state at time t depends on the state at its previous time $t-1$. The state transition probability

$$p(s_t|s_{t-1}) \triangleq a_{s_{t-1}s_t} \tag{3.24}$$

is also introduced as HMM parameters to drive the change of HMM states or equivalently characterize the transition of acoustic events under a phone unit. The parameters of initial state probabilities and state transition probabilities should satisfy the following constraints:

$$\sum_{j=1}^{J} \pi_j = 1, \qquad \sum_{j=1}^{J} a_{ij} = 1, \qquad \forall i \tag{3.25}$$

respectively. A sequence of hidden states $S = \{s_t \in \{1, \cdots, J\} | t = 1, \cdots, T\}$ corresponding to a sequence of observations $\mathbf{O}$ is marginalized in calculation of likelihood function. HMMs basically involve a doubly stochastic process. One is for the observed speech frame sequence $\mathbf{O}$ and the other is for the corresponding HMM state sequence S, which is not observed and is regarded as a latent variable. The model which includes a latent variable is called the *latent model*.

Since S is not observed, the likelihood function of $\mathbf{O}$ is given by the summation of the joint distribution of $\mathbf{O}$ and S over all possible state sequences conditioned on a set of HMM parameters Θ:

$$p(\mathbf{O}|\Theta) = \sum_{S \in \mathcal{S}} p(\mathbf{O}, S|\Theta). \tag{3.26}$$

$\mathcal{S}$ denotes a set of all possible state sequences, which is often omitted when it is trivial. Note that the summation over all possible state sequences requires the exponential order of computations, which is not feasible in a practical use. The joint distribution $p(\mathbf{O}, S|\Theta)$ is also called *complete data likelihood*, where a set of observed data and latent variables ($\{\mathbf{O}, S\}$) is called *complete data*. The joint distribution (complete data likelihood) is a useful representation to provide approximate solutions for the latent model (Dempster *et al*. 1976, Huang, Ariki & Jack 1990). Therefore, we focus on the joint likelihood function of the HMM.

By using the product rule, the joint distribution in Eq. (3.26) is decomposed into the likelihood function $p(\mathbf{O}|S, \Theta)$ given the state sequence S, and the state sequence probability $P(S|\Theta)$ as follows:

$$p(\mathbf{O}, S|\Theta) = p(\mathbf{O}|S, \Theta)p(S|\Theta). \tag{3.27}$$

https://doi.org/10.1017/CBO9781107295360.004 Published online by Cambridge University Press

Since $\mathbf{o}_t$ is independent and identically distributed (iid) given the state sequence $S = \{s_1, \cdots, S_T\}$, the likelihood function $p(\mathbf{O}|S, \Theta)$ is represented as follows:

$$p(\mathbf{O}|S, \Theta) = \prod_{t=1}^{T} p(\mathbf{o}_t|\Theta_{s_t}). \tag{3.28}$$

Here, Θ_j is a set of state-dependent HMM parameters, where j is the index of the HMM state. Similarly, the state sequence probability $P(S|\Theta)$ given the state sequence $S = \{s_1, \cdots, S_T\}$ can be represented by the initial probability in Eq. (3.23) and transition probabilities in Eq. (3.24), as follows:

$$p(S|\Theta) = \pi_{s_1} \prod_{t=2}^{T} a_{s_{t-1}s_t}. \tag{3.29}$$

Thus, by substituting Eqs. (3.28), (3.29), and (3.27) into Eq. (3.26), we can obtain the following equation:

$$p(\mathbf{O}|\Theta) = \sum_{S=\{s_t\}\in\mathcal{S}} \left(\pi_{s_1} p(\mathbf{o}_1|\Theta_{s_1}) \prod_{t=2}^{T} a_{s_{t-1}s_t} p(\mathbf{o}_t|\Theta_{s_t}) \right). \tag{3.30}$$

This is the basic equation of the likelihood function of the HMM with the parameter $\Theta = \{\{\pi_j\}_{j=1}^J, \{\{a_{ij}\}_{j=1}^J\}_{i=1}^J, \{\Theta_j\}_{j=1}^J\}$. $p(\mathbf{o}_t|\Theta_j)$ is called the *emission probability*, and can be any distribution of $\mathbf{o}_t$.

If we use discrete values as observations for $\mathbf{o}_t$, the emission probability of the HMM is represented by a multinomial distribution. This HMM is called a *discrete HMM* (DHMM). In the DHMM, the HMM parameters are formed as $\Theta = \{\{\pi_j\}_{j=1}^J, \{\{a_{ij}\}_{j=1}^J\}_{i=1}^J, \{\{b_{jk}\}_{k=1}^K\}_{j=1}^J\}$ where the emission probabilities of the discrete observation values k given state j are represented as the multinomial distribution parameter b_{jk}. Assuming that the vector space of observations $\mathbf{O} = \{\mathbf{o}_t\}$ in state j is partitioned into K subspaces by clustering or vector quantization, we express the state observation probability b_{jk} using the probability $p(\mathbf{o}_t \in \mathcal{C}_k|\Theta_j)$, which is constrained by the property

$$\sum_{k=1}^{K} p(\mathbf{o}_t \in \mathcal{C}_k|\Theta_j) = \sum_{k=1}^{K} b_{jk} = 1. \tag{3.31}$$

Here $\mathcal{C}_k$ denotes a set of discrete feature vectors that belongs to a partitioned subspace k. This partition is undertaken by using Vector Quantization (VQ) techniques.

Historically, DHMM was first used for speech modeling with the VQ techniques (Matsui & Furui 1994, Jelinek 1997). However, since the speech feature vector is a continuous value, the Gaussian distribution or the mixture of Gaussian distributions is often used for an acoustic model. That is, an acoustic model, which dealt with discrete observation values obtained from VQ codes, was shifted to so-called *continuous density HMM* (CDHMM), where the HMM with Gaussian distribution based emission probabilities could model continuous observation values more appropriately. CDHMM is described in the next section.

https://doi.org/10.1017/CBO9781107295360.004 Published online by Cambridge University Press

3.2.3 Continuous density HMM

For the HMM framework addressed in Section 3.2, the state observation probability $p(\mathbf{o}_t|\Theta_j)$ is calculated according to the type of observation data. When using discrete observation symbols, each observation vector $\mathbf{o}_t \in C_k$ is simply represented by a codebook partition k which is determined through vector quantization over the vector space of all training samples $\mathbf{O}$. Nonetheless, the representation of a high dimensional feature vector $\mathbf{o}_t \in \mathbb{R}^D$ based on a set of discrete codebook labels is not adequate to fully reflect the randomness of the observation vector. More generally, we calculate the probability density function (pdf) of the observation vector $\mathbf{o}_t$ given an HMM state $s_t = j$ to determine the state observation probability $p(\mathbf{o}_t|\Theta_j)$. It is popular to represent the randomness of continuous-valued $\mathbf{o}_t$ using the multivariate Gaussian distribution $\mathcal{N}(\mathbf{o}_t|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$ defined in Appendix C.6 as:

$$\begin{aligned} p(\mathbf{o}_t|\Theta_j) &= \mathcal{N}(\mathbf{o}_t|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) \\ &\triangleq \frac{1}{(2\pi)^{D/2}|\boldsymbol{\Sigma}_j|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{o}_t - \boldsymbol{\mu}_j)^\intercal \boldsymbol{\Sigma}_j^{-1}(\mathbf{o}_t - \boldsymbol{\mu}_j)\right), \end{aligned} \tag{3.32}$$

with state-dependent mean vector $\boldsymbol{\mu}_j \in \mathbb{R}^D$ and covariance matrix $\boldsymbol{\Sigma}_j \in \mathbb{R}^{D\times D}$.

However, a single Gaussian distribution is insufficient to represent the state-dependent observation space for an HMM state j because there are large amounts of training data collected from varying speakers with different genders, ages, accents, speaking rates, channel distortions and background noises, which are used to train the parameters of individual HMM states. Accordingly, a Gaussian mixture model (GMM) is adopted to represent the state-dependent observation space. This is based on a set of Gaussian distributions which reflect the variations of speech feature vectors within an HMM state due to various acoustic conditions. The state observation probability density function of a feature vector $\mathbf{o}_t$ at time t and in state j is expressed by GMM with K mixture components:

$$\begin{aligned} p(\mathbf{o}_t|s_t = j, \Theta_j) &= \sum_{k=1}^{K} p(\mathbf{o}_t, v_t = k|s_t = j, \Theta_j) \\ &= \sum_{k=1}^{K} p(v_t = k|s_t = j, \Theta_j) p(\mathbf{o}_t|s_t = j, v_t = k, \Theta_j) \\ &= \sum_{k=1}^{K} \omega_{jk} \mathcal{N}(\mathbf{o}_t|\boldsymbol{\mu}_{jk}, \boldsymbol{\Sigma}_{jk}). \end{aligned} \tag{3.33}$$

In Eq. (3.33), each mixture component of state j is expressed by a Gaussian distribution:

$$\begin{aligned} p(\mathbf{o}_t|s_t = j, v_t = k, \Theta_j) &= \mathcal{N}(\mathbf{o}_t|\boldsymbol{\mu}_{jk}, \boldsymbol{\Sigma}_{jk}) \\ &\triangleq \frac{1}{(2\pi)^{D/2}|\boldsymbol{\Sigma}_{jk}|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{o}_t - \boldsymbol{\mu}_{jk})^\intercal \boldsymbol{\Sigma}_{jk}^{-1}(\mathbf{o}_t - \boldsymbol{\mu}_{jk})\right), \end{aligned} \tag{3.34}$$

https://doi.org/10.1017/CBO9781107295360.004 Published online by Cambridge University Press

and the prior probability of a mixture component $v_t = k$ is defined as a mixture weight,

$$p(v_t = k|s_t = j, \Theta_j) \triangleq \omega_{jk}. \tag{3.35}$$

The state-dependent GMM parameters $\{\omega_{jk}, \boldsymbol{\mu}_{jk}, \boldsymbol{\Sigma}_{jk}\}_{k=1}^{K}$ consist of mixture weights ω_{jk}, mean vectors $\boldsymbol{\mu}_{jk}$, and covariance matrices $\boldsymbol{\Sigma}_{jk}$ for K Gaussian mixture components. The resulting realization of HMM using GMM as state observation probability is also called the *continuous density HMM* (CDHMM). The CDHMM parameters are formed as

$$\Theta \triangleq \{\{\pi_j\}_{j=1}^{J}, \{\{a_{ij}\}_{j=1}^{J}\}_{i=1}^{J}, \{\{\omega_{jk}, \boldsymbol{\mu}_{jk}, \boldsymbol{\Sigma}_{jk}\}_{k=1}^{K}\}_{j=1}^{J}\}, \tag{3.36}$$

with an additional constraint on the prior probability of mixture components or the mixture weights.

Typically, HMM covariance matrices are assumed to be diagonal in practical implementation, i.e., $\boldsymbol{\Sigma}_{jk} = \text{diag}(\Sigma_{jk1}, \cdots, \Sigma_{jkd}, \cdots, \Sigma_{jkD})$. Note that this book uses Σ to represent a diagonal component of covariance, or variance when $D = 1$. Then, the diagonal covariance version of Eq. (3.34) is factorized to the univariate Gaussian distribution (Appendix C.5) by the dimension index d as follows:

$$\begin{aligned} p(\mathbf{o}_t|s_t = j, v_t = k, \Theta_j) &= \prod_{d=1}^{D} \mathcal{N}(o_{td}|\mu_{jkd}, \Sigma_{jkd}) \\ &\triangleq \prod_{d=1}^{D} \frac{1}{(2\pi)^{1/2}(\Sigma_{jkd})^{1/2}} \exp\left(-\frac{1}{2\Sigma_{jkd}}(o_{td} - \mu_{jkd})^2\right). \end{aligned} \tag{3.37}$$

This diagonal covariance representation can represent the distribution with all scalar variables, which makes the calculation very simple compared with the vector and matrix representation in the multivariate Gaussian distribution in Eq. (3.34). In addition, the number of parameters for the covariance matrix is reduced from $D * (D + 1)/2$ to D, which can reduce the computational and memory costs. The full covariance Gaussian in Eq. (3.34) also requires computation of the inverse and the determinant of the covariance matrix ($\boldsymbol{\Sigma}_{jk}^{-1}$ and $|\boldsymbol{\Sigma}_{jk}|$), which makes the computation numerically unstable in addition to incurring the matrix computation cost. However, the speech feature vectors have correlations over dimensions, and the diagonal covariance assumption is not adequate. There are more sophisticated models such as the subspace mean and variance methods, aiming to bridge the gap between full and diagonal covariance modeling, which have been proposed (Gales 1999, Axelrod, Gopinath & Olsen 2002). In this book, we keep the vector and matrix representation for the multivariate Gaussian distribution for generality.

Based on the above likelihood discussion of the CDHMM for single observation vector $\mathbf{o}_t$ given HMM state $s_t = j$ at frame t, we consider the marginal likelihood for a whole observation sequence $\mathbf{O}$. Under CDHMM, we have an additional latent variable $v_t = k$ from the mixture component of GMM for each speech frame $\mathbf{o}_t$ at time t, compared with Eq. (3.30). Thus, the sequence of mixture components

$$V \triangleq \{v_t \in \{1, \cdots, K\}|t = 1, \cdots, T\}, \tag{3.38}$$

https://doi.org/10.1017/CBO9781107295360.004 Published online by Cambridge University Press

corresponding to the sequence of observed speech frames $\mathbf{O} = \{\mathbf{o}_t | t = 1, \cdots, T\}$, is introduced in estimation of HMM parameters in addition to $S = \{s_t \in \{1, \cdots, J\} | t = 1, \cdots, T\}$. Considering two sets of latent variables S and V, we consider the likelihood function of the CDHMM by following the formulation in Section 3.2.2. First, the marginal likelihood function of the CDHMM is represented as the joint distribution $p(\mathbf{O}, S, V|\Theta)$:

$$p(\mathbf{O}|\Theta) = \sum_{S,V} p(\mathbf{O}, S, V|\Theta), \tag{3.39}$$

where all possible state sequences S and mixture component sequences V are considered in calculation of the marginal distribution. The joint distribution $p(\mathbf{O}, S, V|\Theta)$ is also called the complete data likelihood, as we discussed in Eq. (3.26) with additional latent variable V. This equation is further decomposed by using the product rule as follows:

$$p(\mathbf{O}, S, V|\Theta) = p(\mathbf{O}|S, V, \Theta)p(V|S, \Theta)p(S|\Theta). \tag{3.40}$$

We provide an actual distribution for each probabilistic function.

Since the distributions $p(S|\Theta)$ are given in Eq. (3.29), we focus on $p(V|S, \Theta)$ and $p(\mathbf{O}|S, V, \Theta)$. Since v_t only depends on s_t, given the state sequence $S = \{s_1, \cdots, S_T\}$, $p(V|S, \Theta)$, it is represented by the factorized form with frame t as follows:

$$p(V|S, \Theta) = \prod_{t=1}^{T} p(v_t|s_t, \Theta) = \prod_{t=1}^{T} \omega_{s_t v_t}. \tag{3.41}$$

Similarly, from Eq. (3.34), $p(\mathbf{O}|S, V, \Theta)$ is also represented by the factorized form with frame t given S and V as:

$$\begin{aligned} p(\mathbf{O}|S, V, \Theta) &= \prod_{t=1}^{T} p(\mathbf{o}_t|s_t, v_t, \Theta) \\ &= \prod_{t=1}^{T} \mathcal{N}(\mathbf{o}_t|\boldsymbol{\mu}_{s_t v_t}, \boldsymbol{\Sigma}_{s_t v_t}). \end{aligned} \tag{3.42}$$

Thus, by substituting Eqs. (3.29), (3.41), and (3.42) into Eq. (3.40), the complete data likelihood function of the CDHMM is represented as

$$p(\mathbf{O}, S, V|\Theta) = \pi_{s_1} \omega_{s_1 v_1} \mathcal{N}(\mathbf{o}_1|\boldsymbol{\mu}_{s_1 v_1}, \boldsymbol{\Sigma}_{s_1 v_1}) \left(\prod_{t=2}^{T} a_{s_{t-1} s_t} \omega_{s_t v_t} \mathcal{N}(\mathbf{o}_t|\boldsymbol{\mu}_{s_t v_t}, \boldsymbol{\Sigma}_{s_t v_t}) \right). \tag{3.43}$$

Thus, the marginal likelihood of the CDHMM is extended from Eq. (3.30) as

$$p(\mathbf{O}|\Theta) = \sum_{S,V} \pi_{s_1} \omega_{s_1 v_1} \mathcal{N}(\mathbf{o}_1|\boldsymbol{\mu}_{s_1 v_1}, \boldsymbol{\Sigma}_{s_1 v_1}) \left(\prod_{t=2}^{T} a_{s_{t-1} s_t} \omega_{s_t v_t} \mathcal{N}(\mathbf{o}_t|\boldsymbol{\mu}_{s_t v_t}, \boldsymbol{\Sigma}_{s_t v_t}) \right). \tag{3.44}$$

https://doi.org/10.1017/CBO9781107295360.004 Published online by Cambridge University Press

This is the basic equation of the likelihood function of the CDHMM with the parameter $\Theta = \{\{\pi_j\}, \{a_{ij}\}, \{\omega_{jk}\}, \{\boldsymbol{\mu}_{jk}\}, \{\boldsymbol{\Sigma}_{jk}\}\}$. As we discussed in the previous section, the summation over all possible state and mixture component sequences requires an exponential order of computations, which is not tractable in practical use. Section 3.3 describes how to efficiently compute the marginal likelihood by using the forward and backward algorithms, and most probable state sequence based on the Viterbi algorithm. Section 3.4 also describes how to estimate the HMM parameters Θ efficiently based on these algorithms and the expectation and maximization algorithm.

Before moving to these explanations, as a subset model of the CDHMM, we introduce a simple GMM without an HMM, which is also widely used in speech and language processing.

3.2.4 Gaussian mixture model

The GMM is a simplified model of the CDHMM, without considering the state sequence S in the previous section. However, this simple GMM is still very powerful for modeling speech features. For example, the GMM is used in speech and noise models in speech enhancement, and in speaker models in speaker verification and clustering, which are discussed in Section 4.6. The GMM can also be widely used to model other signals than speech, e.g., image processing, and biosignals. Therefore, this section only introduces the marginal likelihood of a GMM to be used in later chapters, similar to that of the CDHMM in Eq. (3.44):

$$\begin{aligned} p(\mathbf{O}|\Theta) &= \sum_{V} p(\mathbf{O}, V|\Theta) = \sum_{V} p(\mathbf{O}|V, \Theta) p(V|\Theta) \\ &= \sum_{V} \prod_{t=1}^{T} p(\mathbf{o}_t|v_t, \Theta) p(v_t|\Theta) \\ &= \prod_{t=1}^{T} \sum_{v_t=1}^{K} \omega_{v_t} \mathcal{N}(\mathbf{o}_t|\boldsymbol{\mu}_{v_t}, \boldsymbol{\Sigma}_{v_t}), \end{aligned} \tag{3.45}$$

with the parameter $\Theta = \{\{\omega_k\}, \{\boldsymbol{\mu}_k\}, \{\boldsymbol{\Sigma}_k\}\}$. Since v_t is independent of $v_{t' \neq t}$ in a GMM, the sequential summation over V is independently applied to each $p(\mathbf{o}_t|v_t, \Theta)p(v_t|\Theta)$. So, we can factorize $p(\mathbf{O}|\Theta)$ into the following tth frame distribution given the kth mixture component :

$$\begin{aligned} p(\mathbf{o}_t|v_t = k, \Theta) &= \mathcal{N}(\mathbf{o}_t|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \\ p(v_t = k|\Theta) &= \omega_k. \end{aligned} \tag{3.46}$$

Thus, unlike the HMM case that needs to consider the HMM state sequence S (computation based on an exponential order for the frame length), the GMM can compute the likelihood of all possible V by using the straightforward computation of Eq. (3.45) with the linear order computation of the frame length. As regards the parameter estimation, the EM algorithm is widely used, and this is discussed in Section 3.4.

Figure 3.2 illustrates a univariate GMM distribution $p(o) = \frac{1}{3}\mathcal{N}(o|\mu_1 = 2, \Sigma_1 = 36) + \frac{1}{3}\mathcal{N}(o|\mu_2 = 30, \Sigma_2 = 64) + \frac{1}{3}\mathcal{N}(o|\mu_3 = 45, \Sigma_3 = 16)$ which is plotted with

https://doi.org/10.1017/CBO9781107295360.004 Published online by Cambridge University Press

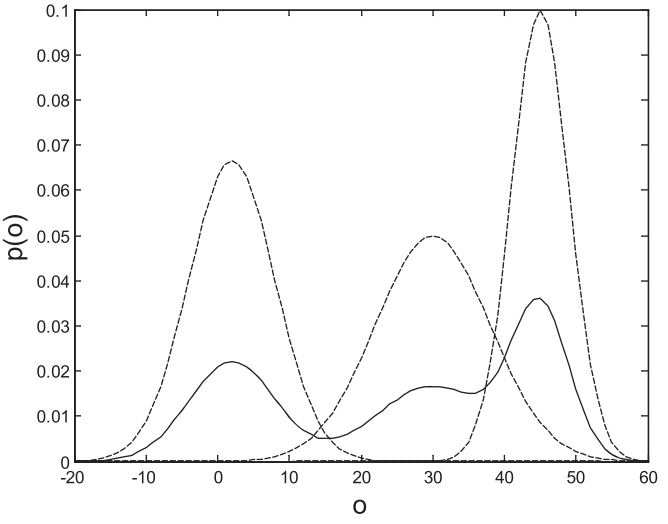

Figure 3.2 Distributions of three individual Gaussians (dashed line) and the corresponding GMM (solid line).

a dashed line and is formed as a mixture of three Gaussian distributions with different means $\{\mu_1, \mu_2, \mu_3\}$ and variances $\{\Sigma_1, \Sigma_2, \Sigma_3\}$, which are shown with a solid line. As shown in the figure, the GMM is a multi-modal distribution with multiple peaks, which can model multiple factors in speech features. Therefore, the GMM can accurately represent speech variations that cannot be represented by a single Gaussian distribution.

3.2.5 Graphical models and generative process of CDHMM

The previous sections explain the marginal likelihood functions of the GMM and CDHMM based on their joint likelihood distributions. As we discussed in the graphical model representation in Section 2.2, once we obtain the joint likelihood distribution of a model, we can obtain the corresponding graphical model and generative process of the model. This section provides the graphical model and generative process of the GMM and CDHMM, respectively, and these are used in the following chapters to deal with these models by using the Bayesian approach.

Graphical models and generative process of GMM

Based on the previous explanations, we can provide a generative process and graphical model of a K-component GMM discussed in Section 3.2.4 as an example. Given feature vectors $\mathbf{O}$, latent variables V, and model parameters Θ, the joint distribution of a complete data set $\{\mathbf{O}, V\}$ is represented from Eq. (3.45) as follows:

$$p(\mathbf{O}, V|\Theta) = \prod_{t=1}^{T} p(\mathbf{o}_t|v_t, \Theta)p(v_t|\Theta). \tag{3.47}$$

https://doi.org/10.1017/CBO9781107295360.004 Published online by Cambridge University Press

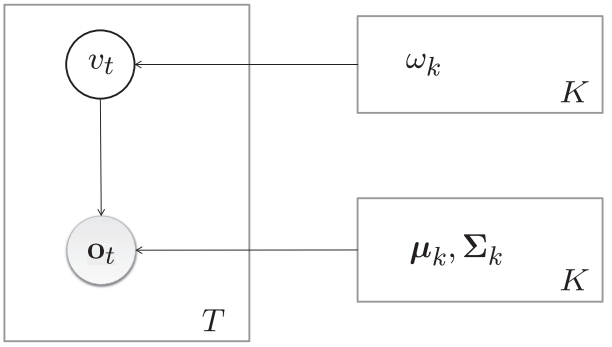

Figure 3.3 Graphical model of Gaussian mixture model.

Thus, the joint distribution of the GMM is parameterized as

$$p(\mathbf{O}, V|\Theta) = \prod_{t=1}^{T} \omega_{v_t} \mathcal{N}(\mathbf{o}_t|\boldsymbol{\mu}_{v_t}, \boldsymbol{\Sigma}_{v_t}). \tag{3.48}$$

Thus, we have the following three kinds of variables:

- **Observation**: $\mathbf{O} = \{\mathbf{o}_1, \cdots, \mathbf{o}_T\}$;
- **Latent**: $V = \{v_1, \cdots, v_T\}$;
- **Non-probabilistic**: $\Theta = \{\omega_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K$.

The dependencies of the above variables are expressed in Eq. (3.48), and we provide the generative process of the GMM in Algorithm 2. Given the mixture weight ω_{v_t}, latent variable v_t is sampled from the multinomial distribution.

In addition, by using the plate for t and k, as discussed in Section 2.2.2, we can simply write the graphical model of the GMM, as shown in Figure 3.3.

Algorithm 2 Generative process of Gaussian mixture model

Require: T, $\{\omega_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K$
1: **for** $t = 1, \cdots, T$ **do**
2: Draw v_t from Mult$(v_t|\{\omega_k\}_{k=1}^K)$
3: Draw $\mathbf{o}_t$ from $\mathcal{N}(\mathbf{o}_t|\boldsymbol{\mu}_{v_t}, \boldsymbol{\Sigma}_{v_t})$
4: **end for**

Graphical models and generative process of CDHMM

Similar to the GMM case, we can also provide a generative process and graphical model of a continuous density HMM discussed in Section 3.2 as another example. Given feature vectors $\mathbf{O}$, latent variables S and V, and model parameters Θ, the joint likelihood function of complete data $\mathbf{O}$, V, and S based on a continuous density HMM is represented as follows:

https://doi.org/10.1017/CBO9781107295360.004 Published online by Cambridge University Press

$$p(\mathbf{O}, S, V|\Theta) = \pi_{s_1}\omega_{s_1v_1}\mathcal{N}(\mathbf{o}_1|\boldsymbol{\mu}_{s_1v_1}, \boldsymbol{\Sigma}_{s_1v_1}) \times \prod_{t=2}^{T} a_{s_{t-1}s_t}\omega_{s_tv_t}\mathcal{N}(\mathbf{o}_t|\boldsymbol{\mu}_{s_tv_t}, \boldsymbol{\Sigma}_{s_tv_t}). \tag{3.49}$$

The CDHMM has the following three kinds of variables:

- **Observation**: $\mathbf{O} = \{\mathbf{o}_1, \cdots, \mathbf{o}_T\}$;
- **Latent**: $S = \{s_1, \cdots, s_T\}$ and $V = \{v_1, \cdots, v_T\}$;
- **Non-probabilistic**: $\Theta = \{\{\pi_j\}_{j=1}^J, \{\{a_{ij}\}_{j=1}^J\}_{i=1}^J, \{\{\omega_{jk}, \boldsymbol{\mu}_{jk}, \boldsymbol{\Sigma}_{jk}\}_{k=1}^K\}_{j=1}^J\}$.

Algorithm 3 Generative process of continuous density hidden Markov model

Require: T, Θ

Draw s_1 from $\text{Mult}(s_1|\{\pi_j\}_{j=1}^J)$
Draw v_1 from $\text{Mult}(v_1|\{\omega_{s_1k}\}_{k=1}^K)$
Draw $\mathbf{o}_1$ from $\mathcal{N}(\mathbf{o}_1|\boldsymbol{\mu}_{s_1v_1}, \boldsymbol{\Sigma}_{s_1v_1})$
for $t = 2, \cdots, T$ **do**
    Draw s_t from $\text{Mult}(s_t|\{a_{s_{t-1}j}\}_{j=1}^J)$
    Draw v_t from $\text{Mult}(v_t|\{\omega_{s_tk}\}_{k=1}^K)$
    Draw $\mathbf{o}_t$ from $\mathcal{N}(\mathbf{o}_t|\boldsymbol{\mu}_{s_tv_t}, \boldsymbol{\Sigma}_{s_tv_t})$
end for

Similarly to the GMM, dependencies of the above variables are expressed in Eq. (3.49), and we provide the generative process of the CDHMM in Algorithm 3. Figure 3.4 shows the graphical model of the CDHMM that uses the plates for the number of HMM states J, and the number of GMM components K. The graphical model of the CDHMM is rather complicated since the state transition makes it difficult to use the plate representation for t.

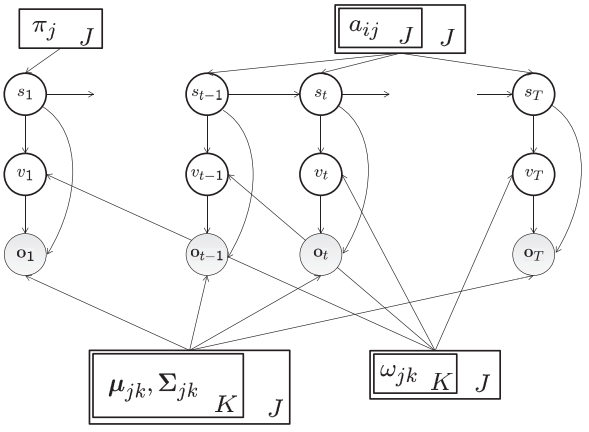

Figure 3.4 Graphical model of continuous density hidden Markov model given parameters Θ.

https://doi.org/10.1017/CBO9781107295360.004 Published online by Cambridge University Press

Thus, we provide the graphical model and generative process examples of our typical target model, CDHMM. Note that these are rather simple based on a non-Bayesian approach since we do not deal with the model parameters as probabilistic variables (do not use circle representations for them).

3.3 Forward–backward and Viterbi algorithms

This section describes how to compute the likelihood values of HMMs (given a latent variable) by using the famous forward–backward and Viterbi algorithms, which are used to solve the statistical speech recognition problem in Eq. (3.3). Although the direct computation of the likelihood values causes the combinatorial explosion problem, these algorithms can provide feasible computational costs. These algorithms are also used to estimate the model parameters, which is described in Section 3.4

3.3.1 Forward–backward algorithm

Basically, the direct computation of marginal likelihood $p(\mathbf{O}|\Theta) = \sum_S p(\mathbf{O}, S|\Theta)$ in Eq. (3.30) involves on the order of $2T \cdot J^T$ calculations, since at every $t = 1, 2, \cdots, T$, there are J possible states that can be reached (i.e., there are J^T possible state sequences), and for each such state sequence about $2T$ calculations are required for each term in the sum of Eq. (3.30). By considering the latent variable of mixture components of the CDHMM in Eq. (3.44), the direct computation of marginal likelihood $p(\mathbf{O}|\Theta) = \sum_{S,V} p(\mathbf{O}, S, V|\Theta)$ is further increased to the order of $2T \cdot J^T K^T$. However, this large computation problem could be tackled by applying the forward–backward algorithm (Rabiner & Juang 1986). According to this algorithm, the *forward variable* $\alpha_t(j)$ is defined by

$$\alpha_t(j) \triangleq p(\mathbf{o}_1, \cdots, \mathbf{o}_t, s_t = j|\Theta). \tag{3.50}$$

This forward variable is known as the probability of the partial observation sequence $\{\mathbf{o}_1, \mathbf{o}_2, \cdots, \mathbf{o}_t\}$ until time t and state j at time t given the current HMM parameters Θ. We also define an emission probability given HMM state j at frame t as

$$b_j(\mathbf{o}_t) \triangleq p(\mathbf{o}_t|s_t = j, \Theta_j). \tag{3.51}$$

In the CDHMM, this is represented by the likelihood function of the GMM, as described in Eq. (3.33). A forward procedure, which is a dynamic programming method, is inductively derived to speed up computation of $p(\mathbf{O}|\Theta)$ as follows:

Forward algorithm

- Initialization

$$\begin{aligned}\alpha_1(j) &= p(\mathbf{o}_1, s_1 = j|\Theta) \\ &= p(\mathbf{o}_1|s_1 = j, \Theta_j)p(s_1 = j|\Theta_j) \\ &= \pi_j b_j(\mathbf{o}_1), \quad 1 \le j \le J.\end{aligned} \tag{3.52}$$

https://doi.org/10.1017/CBO9781107295360.004 Published online by Cambridge University Press

- Induction

$$\begin{aligned}
\alpha_t(j) &= p(\mathbf{o}_1, \cdots, \mathbf{o}_t, s_t = j|\Theta) \\
&= \sum_{i=1}^{J} p(\mathbf{o}_1, \cdots, \mathbf{o}_t, s_{t-1} = i, s_t = j|\Theta) \\
&= \sum_{i=1}^{J} p(\mathbf{o}_t|s_t = j, \mathbf{o}_1, \cdots, \mathbf{o}_{t-1}, s_{t-1} = i, \Theta) \\
&\qquad \times p(s_t = j|\mathbf{o}_1, \cdots, \mathbf{o}_{t-1}, s_{t-1} = i, \Theta) \\
&\qquad \times p(\mathbf{o}_1, \cdots, \mathbf{o}_{t-1}, s_{t-1} = i|\Theta) \\
&= p(\mathbf{o}_t|s_t = j, \Theta_j) \sum_{i=1}^{J} p(s_t = j|s_{t-1} = i, \Theta) p(\mathbf{o}_1, \cdots, \mathbf{o}_{t-1}, s_{t-1} = i|\Theta) \\
&= \left(\sum_{i=1}^{J} \alpha_{t-1}(i) a_{ij} \right) b_j(\mathbf{o}_t), \qquad \begin{array}{l} 2 \le t \le T \\ 1 \le j \le J. \end{array}
\end{aligned} \tag{3.53}$$

- Termination

$$\begin{aligned}
p(\mathbf{O}|\Theta) &= \sum_{j=1}^{J} p(\mathbf{o}_1, \cdots, \mathbf{o}_T, s_T = j|\Theta) \\
&= \sum_{j=1}^{J} \alpha_T(j).
\end{aligned} \tag{3.54}$$

Thus, we can compute the likelihood value $p(\mathbf{O}|\Theta)$ recursively by using the forward variable $\alpha_t(j)$. This iterative algorithm for computing the forward variable is called the *forward algorithm*. In Eq. (3.52) and Eq. (3.53), we see that the calculation of $\{\alpha_t(j)|1 \le t \le T, 1 \le j \le J\}$ requires on the order of J^2T calculations which is dramatically reduced from $2T \cdot J^T$ as required in the direct calculation. Figure 3.5 illustrates the sequence of operations for computation of the forward variable $\alpha_t(j)$.

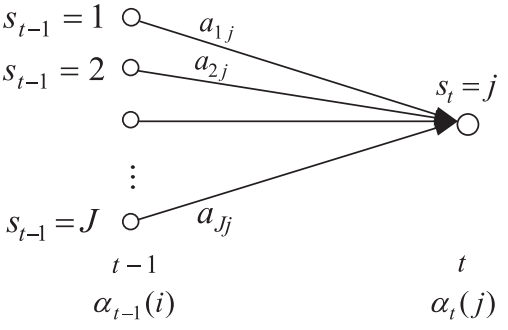

Figure 3.5 Propagation of forward variable from $\alpha_{t-1} = i$ to $\alpha_t = j$. All possible states at time $t-1$ are considered. Adapted from Rabiner & Juang (1993).

https://doi.org/10.1017/CBO9781107295360.004 Published online by Cambridge University Press

Similarly, we consider a backward variable which is defined as

$$\beta_t(j) \triangleq p(\mathbf{o}_{t+1}, \cdots, \mathbf{o}_T | s_t = j, \Theta). \tag{3.55}$$

This variable represents the probability of the partial observation sequence from $t+1$ to the end, given state j at time t and model parameters Θ. Again, we can inductively derive the following backward procedure:

Backward algorithm

- Initialization

$$\beta_T(j) = 1, \quad 1 \le j \le J. \tag{3.56}$$

- Induction

  By using the sum and product rules, we can rewrite $\beta_t(i)$ as

$$\begin{aligned}\beta_t(i) &= p(\mathbf{o}_{t+1}, \cdots, \mathbf{o}_T | s_t = i, \Theta) \\ &= \sum_{j=1}^{J} p(\mathbf{o}_{t+1}, \cdots, \mathbf{o}_T, s_{t+1} = j | s_t = i, \Theta) \\ &= \sum_{j=1}^{J} p(\mathbf{o}_{t+1}, \cdots, \mathbf{o}_T | s_{t+1} = j, s_t = i, \Theta) p(s_{t+1} = j | s_t = i, \Theta).\end{aligned} \tag{3.57}$$

  By using the conditional independence property of the HMM,

$$\begin{aligned}\beta_t(i) &= \sum_{j=1}^{J} p(\mathbf{o}_{t+1}, \cdots, \mathbf{o}_T | s_{t+1} = j, \Theta) p(s_{t+1} = j | s_t = i, \Theta) \\ &= \sum_{j=1}^{J} p(\mathbf{o}_{t+2}, \cdots, \mathbf{o}_T | s_{t+1} = j, \Theta) p(\mathbf{o}_{t+1} | s_{t+1} = j, \Theta_j) \\ &\qquad \times p(s_{t+1} = j | s_t = i, \Theta) \\ &= \sum_{j=1}^{J} a_{ij} b_j(\mathbf{o}_{t+1}) \beta_{t+1}(j), t = T-1, T-2, \cdots, 1, \quad 1 \le i \le J.\end{aligned} \tag{3.58}$$

- Termination

$$\begin{aligned}\beta_0 &\triangleq p(\mathbf{O}|\Theta) \\ &= \sum_{j=1}^{J} p(\mathbf{o}_1, \cdots, \mathbf{o}_T, s_1 = j | \Theta) \\ &= \sum_{j=1}^{J} p(\mathbf{o}_1, \cdots, \mathbf{o}_T | s_1 = j, \Theta) p(s_1 = j | \Theta) \\ &= \sum_{j=1}^{J} p(\mathbf{o}_1 | s_1 = j, \Theta) p(\mathbf{o}_2, \cdots, \mathbf{o}_t | s_1 = j, \Theta) p(s_1 = j | \Theta) \\ &= \sum_{j=1}^{J} \pi_j b_j(\mathbf{o}_1) \beta_1(j).\end{aligned} \tag{3.59}$$

https://doi.org/10.1017/CBO9781107295360.004 Published online by Cambridge University Press

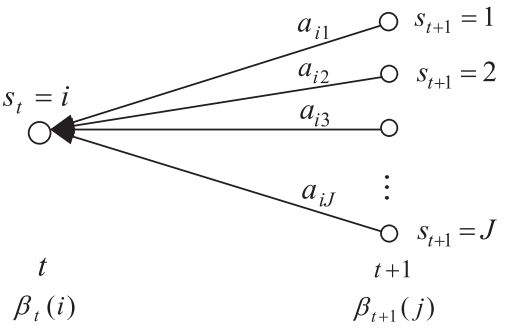

Figure 3.6 Propagation of backward variable from $\alpha_{t+1} = j$ to $\alpha_t = i$. All possible states at time $t+1$ are considered. Adapted from Rabiner & Juang (1993).

Thus, the backward algorithm can also compute the likelihood value $p(\mathbf{O}|\Theta)$ recursively by using the backward variable $\beta_t(j)$. In the initialization step of the backward procedure, $\beta_T(j)$ is arbitrarily assigned to be 1. In the induction step, the backward variable $\beta_t(i)$ is calculated in a backward fashion from $t = T-1$ to the beginning $t = 1$. Figure 3.6 shows the sequence of operations required for computation of the backward variable $\beta_t(i)$. This iterative algorithm for computing the backward variable is called the *backward algorithm*.

The forward variable $\alpha_t(j)$ and backward variable $\beta_t(j)$ are used to calculate the posterior probability of a specific case. For example, if we consider the posterior probability $p(s_t = j|\mathbf{O}, \Theta)$ when the HMM state is j at frame t, we first define

$$\gamma_t(j) \triangleq p(s_t = j|\mathbf{O}, \Theta), \tag{3.60}$$

which is useful in the later explanations. By using the sum and product rules, $\gamma_t(j)$ is represented by the likelihood ratio of

$$\begin{aligned}\gamma_t(j) &\triangleq \frac{p(s_t = j, \mathbf{O}|\Theta)}{p(\mathbf{O}|\Theta)} \\ &= \frac{p(\mathbf{O}, s_t = j|\Theta)}{\sum_{i=1}^{J} p(\mathbf{O}, s_t = i|\Theta)}.\end{aligned} \tag{3.61}$$

Now we focus on the joint distribution $p(s_t = j, \mathbf{O}|\Theta)$, which is rewritten by forward variable $\alpha_t(j)$ in Eq. (3.50) and backward variable $\beta_t(j)$ in Eq. (3.55), as follows:

$$\begin{aligned}p(s_t = j, \mathbf{O}|\Theta) &= p(\mathbf{O}|s_t = j, \Theta)p(s_t = j|\Theta) \\ &= p(\mathbf{o}_1, \cdots, \mathbf{o}_t|s_t = j, \Theta)p(\mathbf{o}_{t+1}, \cdots, \mathbf{o}_T|s_t = j, \Theta)p(s_t = j|\Theta) \\ &= \underbrace{p(\mathbf{o}_1, \cdots, \mathbf{o}_t, s_t = j|\Theta)}_{\alpha_t(j)} \underbrace{p(\mathbf{o}_{t+1}, \cdots, \mathbf{o}_T|s_t = j, \Theta)}_{\beta_t(j)}.\end{aligned} \tag{3.62}$$

From the first to second lines, we use the conditional independence assumption of the HMM. Thus, finally the posterior distribution $\gamma_t(j)$ is represented as:

$$\gamma_t(j) \triangleq p(s_t = j|\mathbf{O}, \Theta) = \frac{\alpha_t(j)\beta_t(j)}{\sum_{j'=1}^{J} \alpha_t(j')\beta_t(j')}. \tag{3.63}$$

https://doi.org/10.1017/CBO9781107295360.004 Published online by Cambridge University Press

This probability will be used to find the optimal state sequence based on the Viterbi algorithm, as mentioned in Section 3.3.2, as well as to estimate the HMM parameters based on the maximum likelihood method, as addressed in Section 3.4.

3.3.2 Viterbi algorithm

Finding the optimal state sequence $\hat{S} = \{\hat{s}_t | t = 1, \cdots, T\}$ of observation sequence $\mathbf{O}$, in terms of the maximum a-posteriori sense, is seen as a fundamental problem in the HMM framework. It is formulated as

$$\hat{S} = \arg\max_{S} p(S|\mathbf{O}, \Theta), \tag{3.64}$$

where $\arg\max_S$ find the most probable $\hat{S}$ from all possible state sequences, and this also involves on the order of J^T calculations, similar to the likelihood computation $p(\mathbf{O}|\Theta)$ in Section 3.3.1. In addition, since $p(\mathbf{O}|\Theta)$ does not depend on S, we can also rewrite Eq. (3.64) as

$$\begin{aligned}\hat{S} &= \arg\max_{S} p(S|\mathbf{O}, \Theta) = \arg\max_{S} p(S|\mathbf{O}, \Theta) p(\mathbf{O}|\Theta) \\ &= \arg\max_{S} p(S, \mathbf{O}|\Theta).\end{aligned} \tag{3.65}$$

That is, the optimal state sequences obtained in terms of the posterior distribution $p(S|\mathbf{O}, \Theta)$ and the joint distribution $p(S, \mathbf{O}|\Theta)$ are equivalent.

The optimal state sequence can be used to determine the segmentation of a speech sentence into phones or sub-phones, if we assume these as latent variables. In this case, the speech frames staying in the same state behave similarly, and the transition from one state to the other is treated as a segmentation boundary.

In addition, the optimal state sequence $\hat{S}$ is applied to approximate the calculation of the marginal likelihood function over S as follows:

$$p(\mathbf{O}|\Theta) = \sum_{S} p(S, \mathbf{O}|\Theta) \approx p(\hat{S}, \mathbf{O}|\Theta). \tag{3.66}$$

That is, the marginal likelihood function is approximately represented as the joint likelihood distribution of $\hat{S}$ and $\mathbf{O}$, which can be represented with Eq. (3.30) as

$$\begin{aligned}p(\hat{S}, \mathbf{O}|\Theta) &= \left(\pi_{\hat{s}_1} p(\mathbf{o}_1|\Theta_{\hat{s}_1}) \prod_{t=2}^{T} a_{\hat{s}_{t-1}\hat{s}_t} p(\mathbf{o}_t|\Theta_{\hat{s}_t}) \right) \\ &\triangleq \hat{p}.\end{aligned} \tag{3.67}$$

Therefore, the approximation of marginal likelihood using the optimal state sequence provides the solution to the *segmental* likelihood function $\hat{p}$ since likelihood calculation is based on the speech segmentation using $\hat{S}$. Considering all possible state sequences, $\{S\}$ is simplified to applying only the optimal state sequence $\hat{S}$. The number of calculations is further reduced to $2T$ as referred to Eq. (3.67).

https://doi.org/10.1017/CBO9781107295360.004 Published online by Cambridge University Press

We can individually determine the most likely state $\hat{s}_t$ for each time frame $1 \leq t \leq T$ according to the posterior probability as

$$\hat{s}_t = \arg \max_{1 \leq j \leq J} p(s_t = j | \mathbf{O}, \Theta). \tag{3.68}$$

The posterior probability $p(s_t = j|\mathbf{O}, \Theta)$ is calculated according to the forward-backward algorithm. As shown in Eq. (3.63), the joint distribution of observation sequence $\mathbf{O}$ and state $s_t = j$ at time t is calculated as a product of the forward variable, which accounts for partial observations $\{\mathbf{o}_1, \cdots, \mathbf{o}_t\}$ and state j at time t, and the backward variable, which accounts for the remainder of observations $\{\mathbf{o}_{t+1}, \cdots, \mathbf{o}_T\}$ given state j at time t with the normalization factor. However, Eq. (3.68) may not determine a *valid* optimal state sequence since the probability of occurrence of the *sequences* of states is not considered.

To cope with this problem, we need to find the single best state sequence by using Eq. (3.64) or an equivalent form of (3.65), directly. A dynamic programming method, called the Viterbi algorithm (Viterbi 1967), was proposed to efficiently find the single best state sequence. To do so, we define the highest probability along a single path, at time t, which accounts for the first t observations and ends in state j using a new notation:

$$\delta_t(j) \triangleq \max_{s_1, \cdots, s_{t-1}} p(s_1, \cdots, s_t = j, \mathbf{o}_1, \cdots, \mathbf{o}_t | \Theta). \tag{3.69}$$

By induction, a recursive formula of $\delta_{t+1}(j)$ from $\delta_t(j)$ is derived to calculate this probability. To derive the equation, we first focus on the joint distribution appearing in $\delta_{t+1}(j)$, which can be rewritten when $s_t = i$ and $s_{t+1} = j$ as:

$$\begin{aligned} & p(s_1, \cdots, i, j, \mathbf{o}_1, \cdots, \mathbf{o}_t, \mathbf{o}_{t+1} | \Theta) \\ & \quad = p(s_1, \cdots, i, \mathbf{o}_1, \cdots, \mathbf{o}_t | \Theta) p(j, \mathbf{o}_{t+1} | s_1, \cdots, i, \mathbf{o}_1, \cdots, \mathbf{o}_t, \Theta) \\ & \quad = p(s_1, \cdots, i, \mathbf{o}_1, \cdots, \mathbf{o}_t | \Theta) p(j | i, \Theta) p(\mathbf{o}_{t+1} | j, \Theta) \\ & \quad = p(s_1, \cdots, i, \mathbf{o}_1, \cdots, \mathbf{o}_t | \Theta) a_{ij} b_j(\mathbf{o}_{t+1}). \end{aligned} \tag{3.70}$$

Here we use the conditional independence of the HMM from the second to the third lines. Thus, by using Eq. (3.69), $\delta_{t+1}(j)$ is computed recursively from $\delta_{t+1}(i)$ as:

$$\begin{aligned} \delta_{t+1}(j) &= \max_{s_1, \cdots, i} p(s_1, \cdots, i, \mathbf{o}_1, \cdots, \mathbf{o}_t | \Theta) a_{ij} b_j(\mathbf{o}_{t+1}) \\ &= \left(\max_i \delta_t(i) a_{ij} \right) b_j(\mathbf{o}_{t+1}). \end{aligned} \tag{3.71}$$

We need to keep track of the state that maximized Eq. (3.71) so as to backtrack to the single best state sequence in the following Viterbi algorithm:

- Initialization

$$\delta_1(i) = \pi_i b_i(\mathbf{o}_1), \quad \psi_1(i) = 0, \quad 1 \leq i \leq J. \tag{3.72}$$

- Recursion

$$\delta_t(j) = \left(\max_{1 \leq i \leq J} \delta_{t-1}(i) a_{ij} \right) \cdot b_j(\mathbf{o}_{t+1}),$$

https://doi.org/10.1017/CBO9781107295360.004 Published online by Cambridge University Press

$$\psi_t(j) = \left(\arg \max_{1 \le i \le J} \delta_{t-1}(i) a_{ij} \right), \qquad \begin{matrix} 2 \le t \le T \\ 1 \le j \le J. \end{matrix} \tag{3.73}$$

- Termination

$$\begin{aligned} \hat{p} &= \max_{1 \le j \le J} \delta_T(i), \\ \hat{s}_T &= \arg \max_{1 \le j \le J} \delta_T(i). \end{aligned} \tag{3.74}$$

- State sequence backtracking

$$\hat{s}_t = \psi_{t+1}(\hat{s}_{t+1}), \qquad t = T-1, T-2, \cdots, 1. \tag{3.75}$$

In the termination step, the segmental likelihood function $\hat{p}$ is calculated and is equivalent to Eq. (3.67). It is noteworthy that this Viterbi algorithm is similar to the forward procedure in the forward–backward algorithm. The key difference is the maximization in Eq. (3.73) over previous states, which is used in place of a summation operation in Eq. (3.53). The variable $\delta_t(j)$ in the Viterbi algorithm is meaningfully related to the forward variable $\alpha_t(j)$ in the forward–backward algorithm.

Now, we summarize what we can compute from the HMM without taking on the combinatorial explosion problem. These values are used in the decoding step and the training step of estimating model parameters Θ, which is discussed in the next section.

- $p(\mathbf{O}|\Theta)$:
  The marginalized likelihood function from the forward or backward algorithm.
- $\gamma_t(j) \triangleq p(s_t = j|\mathbf{O}, \Theta)$:
  The posterior probability of $s_t = j$ from the forward–backward algorithm.
- $\hat{S} = \arg\max_S p(S|\mathbf{O}, \Theta) = \arg\max_S p(S, \mathbf{O}|\Theta)$:
  The optimal state sequence from the Viterbi algorithm.
- $p(\hat{S}, \mathbf{O}|\Theta)$:
  The segmental joint likelihood function from the Viterbi algorithm.

3.4 Maximum likelihood estimation and EM algorithm

Previous sections discuss the HMM-based speech modeling given model parameters Θ to compute the likelihood values and so on, efficiently based on the forward, backward, and Viterbi algorithms. One of the powerful properties of the HMM is that it also provides an efficient algorithm to obtain the model parameters based on the ML estimation from training data. The training algorithm is based on the EM algorithm, which can tackle the incomplete data problem in ML estimation. In what follows, we address how the EM algorithm is derived by applying Jensen's inequality and show the procedure of EM steps for estimation of HMM parameters, which involves latent variables in probabilistic functions (Nakagawa 1988, Huang *et al*. 1990, Rabiner & Juang 1993, Bilmes 1998).

https://doi.org/10.1017/CBO9781107295360.004 Published online by Cambridge University Press

3.4.1 Jensen's inequality

We start the discussion of HMM parameter estimation based on the Maximum Likelihood (ML) criterion, so that the optimal model parameters $\hat{\Theta}$ are computed by the likelihood function $p(\mathbf{O}|\Theta)$:[2]

$$\hat{\Theta} = \arg\max_{\Theta} p(\mathbf{O}|\Theta). \tag{3.76}$$

We assume that speech data $\mathbf{O}$ are observed and are generated by some distribution. However, as we discussed in Section 3.2, the HMM has additional hidden variables S (HMM state sequence) and V (GMM component sequence), which are not observed. In this situation where the model includes unobserved variables, $\mathbf{O}$ is called incomplete data (S and V). (Θ would also be included as unobserved variables in a broad sense, but this book considers latent variables as unobserved variables.). Conversely, the complete data $\mathbf{Y}$ are composed of the observed data $\mathbf{O}$ as well as the hidden variables $\{S, V\}$.

In general, the maximum likelihood estimation of the model parameter Θ for complete data $\mathbf{O}$ is difficult, since we need to consider the following equation:

$$\hat{\Theta} = \arg\max_{\Theta} \sum_{S,V} p(\mathbf{O}, S, V|\Theta). \tag{3.77}$$

As we discussed in Section 3.3, the summation over all possible S and V has to overcome the combinatorial explosion problem, and the direct optimization of Θ for this equation is not feasible. This problem is called the *incomplete data problem*.

According to the EM algorithm, the incomplete data problem in the ML estimation is resolved by iteratively and alternatively performing the expectation step (E-step) and the maximization step (M-step), which results in obtaining the local optimum solution of the model parameters. In the E-step, we calculate an auxiliary function $Q(\Theta'|\Theta)$, which is an expectation of the logarithm of the likelihood function using new HMM parameters Θ' given the current parameters Θ, i.e.

$$\begin{aligned} Q(\Theta'|\Theta) &= \mathbb{E}_{(S,V)}[\log p(\mathbf{O}, S, V|\Theta')|\mathbf{O}, \Theta] \\ &= \sum_{S} \sum_{V} p(S, V|\mathbf{O}, \Theta) \log p(\mathbf{O}, S, V|\Theta'). \end{aligned} \tag{3.78}$$

In the M-step, we maximize the auxiliary function instead of Eq. (3.77) with respect to the HMM parameters Θ' and estimate new parameters by

$$\hat{\Theta}' = \arg\max_{\Theta'} Q(\Theta'|\Theta). \tag{3.79}$$

The updated HMM parameters $\hat{\Theta}'$ are then treated as the current parameters for the next iteration of EM steps. This iterative estimation only obtains local optimum solutions, and not global optimum solutions. However, a careful setting of the initial model parameters would help the solution to reach appropriate parameter values, and moreover, the algorithm theoretically guarantees that the likelihood value is always increased as the number of iterations increases. This property is very useful in the implementation,

[2] The optimization with respect to the posterior distribution $p(\Theta|\mathbf{O})$ is more reasonable in the Bayesian sense, and it will be discussed in the following chapters.

https://doi.org/10.1017/CBO9781107295360.004 Published online by Cambridge University Press

since we can easily debug training source codes based on the EM algorithm by checking likelihood values.

Now, we prove how this indirect optimization for the auxiliary function $Q(\Theta'|\Theta)$ increases a likelihood value. For this proof, we first define the following logarithmic marginal function $L(\Theta')$:

$$L(\Theta') \triangleq \log p(\mathbf{O}|\Theta'). \tag{3.80}$$

Note that since the logarithmic function is a monotonically increasing function, we can check $L(\Theta')$ instead of $p(\mathbf{O}|\Theta')$ for the proof. We introduce the notation of complete data:

$$\mathbf{Y} \triangleq \{\mathbf{O}, S, V\}. \tag{3.81}$$

Then, the conditional distribution of $\mathbf{Y}$ given incomplete data $\mathbf{O}$ and parameters Θ has the following relationship from the product rule:

$$p(\mathbf{Y}|\mathbf{O}, \Theta') = \frac{p(\mathbf{Y}, \mathbf{O}|\Theta')}{p(\mathbf{O}|\Theta')} = \frac{p(\mathbf{Y}|\Theta')}{p(\mathbf{O}|\Theta')}, \tag{3.82}$$

where the likelihood function $p(\mathbf{Y}, \mathbf{O}|\Theta')$ is rewritten as $p(\mathbf{Y}|\Theta')$, since $\mathbf{Y}$ includes $\mathbf{O}$. Therefore, by taking the logarithmic operation for both sides in Eq. (3.82), we derive the equation for $L(\Theta')$ as

$$L(\Theta') = \log p(\mathbf{Y}|\Theta') - \log p(\mathbf{Y}|\mathbf{O}, \Theta'). \tag{3.83}$$

Then, by taking the expectation operation with respect to the posterior distribution of latent variables $p(S, V|\mathbf{O}, \Theta)$ for Eq. (3.83), Eq. (3.83) is represented as

$$\begin{aligned} L(\Theta') &= \mathbb{E}_{(S,V)}[\log p(\mathbf{Y}|\boldsymbol{\Theta}')|\mathbf{O}, \Theta] - \mathbb{E}_{(S,V)}[\log p(\mathbf{Y}|\mathbf{O}, \Theta')|\mathbf{O}, \Theta] \\ &= Q(\Theta'|\Theta) - H(\Theta'|\Theta), \end{aligned} \tag{3.84}$$

where $H(\Theta'|\Theta)$ is defined as follows:

$$H(\Theta'|\Theta) \triangleq \mathbb{E}_{(\mathbf{Y})}[\log p(\mathbf{Y}|\mathbf{O}, \Theta')|\mathbf{O}, \Theta]. \tag{3.85}$$

$Q(\Theta'|\Theta)$ is defined in Eq. (3.78). Then, we obtain this relation from Eq. (3.85):

$$Q(\Theta'|\Theta) = L(\Theta') + H(\Theta'|\Theta). \tag{3.86}$$

Here, Jensen's inequality is introduced to prove the inequality (Dempster *et al.* 1976),

$$H(\Theta'|\Theta) \leq H(\Theta|\Theta). \tag{3.87}$$

- **Jensen's inequality**
  If x is a random variable with mean value $\mathbb{E}_{(x)}[x] = \mu$ and $f(x)$ is a *convex function*, then

  $$\mathbb{E}_{(x)}[f(x)] \geq f[\mathbb{E}_{(x)}(x)], \tag{3.88}$$

  with equality if and only if x is a degenerate distribution at μ.

https://doi.org/10.1017/CBO9781107295360.004 Published online by Cambridge University Press

The inequality in Eq. (3.87) is derived by

$$\begin{aligned} H(\Theta|\Theta) - H(\Theta'|\Theta) &= \mathbb{E}_{(\mathbf{Y})}\left[\log \frac{p(\mathbf{Y}|\mathbf{O},\boldsymbol{\Theta})}{p(\mathbf{Y}|\mathbf{O},\boldsymbol{\Theta}')}\middle|\mathbf{O},\boldsymbol{\Theta}\right] \\ &= \int p(\mathbf{Y}|\mathbf{O},\Theta)\left(-\log\frac{p(\mathbf{Y}|\mathbf{O},\Theta')}{p(\mathbf{Y}|\mathbf{O},\Theta)}\right)d\mathbf{Y} \\ &= \mathrm{KL}(p(\mathbf{Y}|\mathbf{O},\Theta)\|p(\mathbf{Y}|\mathbf{O},\Theta')) \\ &\geq -\log\left(\int p(\mathbf{Y}|\mathbf{O},\Theta)\frac{p(\mathbf{Y}|\mathbf{O},\Theta')}{p(\mathbf{Y}|\mathbf{O},\Theta)}d\mathbf{Y}\right) = 0, \end{aligned} \tag{3.89}$$

where a convex function based on the negative logarithm $f(x) = -\log(x)$ is adopted. As shown in Eq. (3.89), the difference $H(\Theta|\Theta) - H(\Theta'|\Theta)$ is obtained as the *relative entropy* or *Kullback–Leibler (KL) divergence* (Kullback & Leibler 1951) between the distributions $p(\mathbf{Y}|\mathbf{O},\Theta)$ and $p(\mathbf{Y}|\mathbf{O},\Theta')$, that is

$$\mathrm{KL}(p(\mathbf{Y}|\mathbf{O},\Theta)\|p(\mathbf{Y}|\mathbf{O},\Theta')) \triangleq \int p(\mathbf{Y}|\mathbf{O},\Theta)\left(-\log\frac{p(\mathbf{Y}|\mathbf{O},\Theta')}{p(\mathbf{Y}|\mathbf{O},\Theta)}\right)d\mathbf{Y}. \tag{3.90}$$

Given Eqs. (3.87) and (3.86), it is straightforward to see that if Θ' satisfies

$$Q(\Theta'|\Theta) \geq Q(\Theta|\Theta), \tag{3.91}$$

then we can prove that

$$L(\Theta') = Q(\Theta'|\Theta) - H(\Theta'|\Theta) \geq Q(\Theta|\Theta) - H(\Theta|\Theta) = L(\Theta). \tag{3.92}$$

Since $L(\Theta) = \log p(\mathbf{O}|\Theta)$ and the logarithmic function is a monotonic function,

$$Q(\Theta'|\Theta) \geq Q(\Theta|\Theta) \Rightarrow p(\mathbf{O}|\Theta') \geq p(\mathbf{O}|\Theta). \tag{3.93}$$

Since $\hat{\Theta}' = \arg\max_{\Theta'} Q(\Theta'|\Theta)$ always satisfies the inequality Eq. (3.91), we prove that the parameter estimated by the EM procedure, $\hat{\Theta}'$, always increases the likelihood value as:

$$\hat{\Theta}' = \arg\max_{\Theta'} Q(\Theta'|\Theta) \Rightarrow p(\mathbf{O}|\Theta') \geq p(\mathbf{O}|\Theta). \tag{3.94}$$

Such an EM procedure is bound to monotonically increase the auxiliary function $Q(\Theta'|\Theta)$ as well as the original likelihood function $p(\mathbf{O}|\Theta')$. Note that since it is not a direct optimization of the original likelihood function, the optimization of the auxiliary function leads to a local optimum solution to the ML parameter estimation.

3.4.2 Expectation step

To find ML estimates of HMM parameters, we expand the auxiliary function in Eq. (3.78) and rewrite it by substituting the joint distribution of complete data likelihood (Eq. (3.49)) into Eq. (3.78) as

https://doi.org/10.1017/CBO9781107295360.004 Published online by Cambridge University Press

$$
\begin{aligned}
Q(\Theta'|\Theta) &= \mathbb{E}_{(S,V)}[\log p(\mathbf{O}, S, V|\Theta')|\mathbf{O}, \Theta] \\
&= \sum_{S,V} p(S, V|\mathbf{O}, \Theta)\Bigg(\log \pi'_{s_1} + \left(\sum_{t=2}^{T} \log a'_{s_{t-1}s_t}\right) \\
&\quad + \left(\sum_{t=1}^{T} \log \omega'_{s_t v_t} + \log \mathcal{N}(\mathbf{o}_t|\boldsymbol{\mu}'_{s_t v_t}, \boldsymbol{\Sigma}'_{s_t v_t})\right)\Bigg).
\end{aligned} \tag{3.95}
$$

Note that four terms depend on the initial weight π_j, state transition probability a_{ij}, mixture weight w_{jk}, and Gaussian parameters $\{\boldsymbol{\mu}_{jk}, \boldsymbol{\Sigma}_{jk}\}$, respectively. We provide the solution for each term:

- $Q(\boldsymbol{\pi}'|\boldsymbol{\pi})$
  We first focus on the first term depending on the initial weight π_j, and define the following auxiliary function for π_j:

$$
Q(\boldsymbol{\pi}'|\boldsymbol{\pi}) \triangleq \sum_{S,V} p(S, V|\mathbf{O}, \Theta) \log \pi'_{s_1}. \tag{3.96}
$$

  Since π'_{s_1} only depends on s_1, we obtain the following equation that marginalizes $p(S, V|\mathbf{O}, \Theta)$ over $S_{\setminus s_1} = \{s_2, \cdots, s_T\}$ and V as:

$$
\sum_{S_{\setminus s_1},V} p(S, V|\mathbf{O}, \Theta) = p(s_1|\mathbf{O}, \Theta). \tag{3.97}
$$

  Therefore, $Q(\boldsymbol{\pi}'|\boldsymbol{\pi})$ can be rewritten as

$$
\begin{aligned}
Q(\boldsymbol{\pi}'|\boldsymbol{\pi}) &= \sum_{s_1} p(s_1|\mathbf{O}, \Theta) \log \pi'_{s_1} \\
&= \sum_{j=1}^{J} p(s_1 = j|\mathbf{O}, \Theta) \log \pi'_j \\
&= \sum_{j=1}^{J} \gamma_1(j) \log \pi'_j,
\end{aligned} \tag{3.98}
$$

  where $\gamma_1(j)$ is an occupation probability defined in Eq. (3.60) as:

$$
\gamma_1(j) \triangleq p(s_1 = j|\mathbf{O}, \Theta). \tag{3.99}
$$

  This is computed from the forward–backward algorithm.
- $Q(\mathbf{A}'|\mathbf{A})$
  Next, we focus on the second term in Eq. (3.95), which depends on the state transition probability a_{ij}, and define the following auxiliary function for a_{ij}:

$$
Q(\mathbf{A}'|\mathbf{A}) \triangleq \sum_{S,V} p(S, V|\mathbf{O}, \Theta) \sum_{t=1}^{T-1} \log a'_{s_t s_{t+1}}. \tag{3.100}
$$

https://doi.org/10.1017/CBO9781107295360.004 Published online by Cambridge University Press

Here we replace the summation from $\sum_{t=2}^{T} \log a'_{s_{t-1}s_t}$ to $\sum_{t=1}^{T-1} \log a'_{s_t s_{t+1}}$ for notational convention. Similar to $Q(\boldsymbol{\pi}'|\boldsymbol{\pi})$, we obtain

$$\sum_{S_{\setminus s_t, s_{t+1}}, V} p(S, V|\mathbf{O}, \Theta) = p(s_t, s_{t+1}|\mathbf{O}, \Theta). \tag{3.101}$$

Therefore, by replacing the summation over t with the summation over S, V, we obtain

$$\begin{aligned} Q(\mathbf{A}'|\mathbf{A}) &= \sum_{t=1}^{T-1} \sum_{S,V} p(S, V|\mathbf{O}, \Theta) \log a'_{s_t s_{t+1}} \\ &= \sum_{t=1}^{T-1} \sum_{s_t, s_{t+1}} p(s_t, s_{t+1}|\mathbf{O}, \Theta) \log a'_{s_t s_{t+1}} \\ &= \sum_{t=1}^{T-1} \sum_{i=1}^{J} \sum_{j=1}^{J} p(s_t = i, s_{t+1} = j|\mathbf{O}, \Theta) \log a'_{ij} \\ &= \sum_{t=1}^{T-1} \sum_{i=1}^{J} \sum_{j=1}^{J} \xi_t(i, j) \log a'_{ij}, \end{aligned} \tag{3.102}$$

where $\xi_t(i, j)$ is an expected transition probability from $s_t = i$ to $s_{t+1} = j$, and is defined as:

$$\xi_t(i, j) \triangleq p(s_t = i, s_{t+1} = j|\mathbf{O}, \Theta). \tag{3.103}$$

Note that by using this technique, the summation over all possible sequences (that leads to a combinatorial explosion) can be replaced with a summation over the number of HMM states and the number of frames. Thus, this auxiliary function can be computed feasibly. For this computation, we need to obtain $\xi_t(i, j)$, which is discussed later as a variant of the forward–backward algorithm.

- $Q(\boldsymbol{\omega}'|\boldsymbol{\omega})$

  The third term in Eq. (3.95) depends on the mixture weight ω_{jk}, and so we define the following auxiliary function for ω_{jk}:

$$Q(\boldsymbol{\omega}'|\boldsymbol{\omega}) \triangleq \sum_{S,V} p(S, V|\mathbf{O}, \Theta) \sum_{t=1}^{T} \log \omega'_{s_t v_t}. \tag{3.104}$$

  Similarly to the case of a_{ij}, we first obtain the following equation:

$$\sum_{S_{\setminus s_t}, V_{\setminus v_t}} p(S, V|\mathbf{O}, \Theta) = p(s_t, v_t|\mathbf{O}, \Theta). \tag{3.105}$$

  Therefore,

$$\begin{aligned} Q(\boldsymbol{\omega}'|\boldsymbol{\omega}) &= \sum_{t=1}^{T} \sum_{S,V} p(S, V|\mathbf{O}, \Theta) \log \omega'_{s_t v_t} \\ &= \sum_{t=1}^{T} \sum_{s_t, v_t} p(s_t, v_t|\mathbf{O}, \Theta) \log \omega'_{s_t v_t} \end{aligned}$$

https://doi.org/10.1017/CBO9781107295360.004 Published online by Cambridge University Press

$$= \sum_{t=1}^{T}\sum_{j=1}^{J}\sum_{k=1}^{K} p(s_t = j, v_t = k|\mathbf{O}, \Theta) \log \omega'_{jk}$$
$$= \sum_{t=1}^{T}\sum_{j=1}^{J}\sum_{k=1}^{K} \gamma_t(j,k) \log \omega'_{jk}, \tag{3.106}$$

where $\gamma_t(j,k)$ is an expected occupation probability at $s_t = j$ and $v_t = k$, and is defined as:

$$\gamma_t(j,k) \triangleq p(s_t = j, v_t = k|\mathbf{O}, \Theta). \tag{3.107}$$

The computation of $\gamma_t(j,k)$ is also discussed later as a variant of the forward–backward algorithm.

- $Q(\boldsymbol{\mu}', \boldsymbol{\Sigma}'|\boldsymbol{\mu}, \boldsymbol{\Sigma})$

  Finally, the fourth term in Eq. (3.95) depends on the Gaussian parameters $\boldsymbol{\mu}_{jk}$ and $\boldsymbol{\Sigma}_{jk}$, and defines the following auxiliary function for $\boldsymbol{\mu}_{jk}$ and $\boldsymbol{\Sigma}_{jk}$:

$$Q(\boldsymbol{\mu}', \boldsymbol{\Sigma}'|\boldsymbol{\mu}, \boldsymbol{\Sigma}) \triangleq \sum_{S,V} p(S, V|\mathbf{O}, \Theta) \sum_{t=1}^{T} \log \mathcal{N}(\mathbf{o}_t|\boldsymbol{\mu}'_{s_t v_t}, \boldsymbol{\Sigma}'_{s_t v_t}). \tag{3.108}$$

  Similarly to $Q(\boldsymbol{\omega}'|\boldsymbol{\omega})$, by using Eq. (3.105), $Q(\boldsymbol{\mu}', \boldsymbol{\Sigma}'|\boldsymbol{\mu}, \boldsymbol{\Sigma})$ can be rewritten with $\gamma_t(j,k)$ as

$$Q(\boldsymbol{\mu}', \boldsymbol{\Sigma}'|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{t=1}^{T}\sum_{j=1}^{J}\sum_{k=1}^{K} \gamma_t(j,k) \log \mathcal{N}(\mathbf{o}_t|\boldsymbol{\mu}'_{jk}, \boldsymbol{\Sigma}'_{jk}). \tag{3.109}$$

  By using the definition of the multivariate Gaussian distribution in Appendix C.6:

$$\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) \triangleq (2\pi)^{-\frac{D}{2}} |\boldsymbol{\Sigma}|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^{\mathsf{T}} \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu})\right). \tag{3.110}$$

  Equation (3.109) can be rewritten as

$$\begin{aligned} &Q(\boldsymbol{\mu}', \boldsymbol{\Sigma}'|\boldsymbol{\mu}, \boldsymbol{\Sigma}) \\ &= \sum_{t=1}^{T}\sum_{j=1}^{J}\sum_{k=1}^{K} \gamma_t(j,k) \log \left((2\pi)^{-\frac{D}{2}} |\boldsymbol{\Sigma}'_{jk}|^{-\frac{1}{2}} \right. \\ &\quad \left. \times \exp\left(-\frac{1}{2}(\mathbf{o}_t - \boldsymbol{\mu}'_{jk})^{\mathsf{T}} (\boldsymbol{\Sigma}'_{jk})^{-1} (\mathbf{o}_t - \boldsymbol{\mu}'_{jk})\right)\right) \\ &\propto \sum_{t=1}^{T}\sum_{j=1}^{J}\sum_{k=1}^{K} -\frac{\gamma_t(j,k)}{2} \left(\log\left(|\boldsymbol{\Sigma}'_{jk}|\right) + (\mathbf{o}_t - \boldsymbol{\mu}'_{jk})^{\mathsf{T}} (\boldsymbol{\Sigma}'_{jk})^{-1} (\mathbf{o}_t - \boldsymbol{\mu}'_{jk})\right), \end{aligned} \tag{3.111}$$

  where $\propto$ denotes the proportional relationship in the logarithmic domain. That is, the normalization factor in the linear domain is changed for the normalization constant in the logarithmic domain, and we shall continue to use $\propto$ in this book. Therefore, the normalization constant term that does not depend on $\boldsymbol{\mu}$, $\boldsymbol{\Sigma}$ is omitted from this expression.

https://doi.org/10.1017/CBO9781107295360.004 Published online by Cambridge University Press

Thus, we summarize the auxiliary function $Q(\Theta'|\Theta)$:

$$Q(\Theta'|\Theta) = Q(\boldsymbol{\pi}'|\boldsymbol{\pi}) + Q(\mathbf{A}'|\mathbf{A}) + Q(\boldsymbol{\omega}'|\boldsymbol{\omega}) + Q(\boldsymbol{\mu}', \boldsymbol{\Sigma}'|\boldsymbol{\mu}, \boldsymbol{\Sigma}), \tag{3.112}$$

where each term is defined as follows:

$$Q(\boldsymbol{\pi}'|\boldsymbol{\pi}) = \sum_{j=1}^{J} \gamma_1(j) \log \pi'_j, \tag{3.113}$$

$$Q(\mathbf{A}'|\mathbf{A}) = \sum_{t=1}^{T-1} \sum_{i=1}^{J} \sum_{j=1}^{J} \xi_t(i,j) \log a'_{ij}, \tag{3.114}$$

$$Q(\boldsymbol{\omega}'|\boldsymbol{\omega}) = \sum_{t=1}^{T} \sum_{j=1}^{J} \sum_{k=1}^{K} \gamma_t(j,k) \log \omega'_{jk}, \tag{3.115}$$

$$\begin{aligned} Q(\boldsymbol{\mu}', \boldsymbol{\Sigma}'|\boldsymbol{\mu}, \boldsymbol{\Sigma}) \propto \sum_{t=1}^{T} \sum_{j=1}^{J} \sum_{k=1}^{K} -\frac{\gamma_t(j,k)}{2} \Big(& \log |\boldsymbol{\Sigma}'_{jk}| \\ & + (\mathbf{o}_t - \boldsymbol{\mu}'_{jk})^{\mathsf{T}} (\boldsymbol{\Sigma}'_{jk})^{-1} (\mathbf{o}_t - \boldsymbol{\mu}'_{jk}) \Big). \end{aligned} \tag{3.116}$$

As an equivalent form of Eq. (3.116), we also write the following auxiliary function $Q(\boldsymbol{\mu}', \mathbf{R}'|\boldsymbol{\mu}, \mathbf{R})$, which replaces the covariance matrix $\boldsymbol{\Sigma}$ with the precision matrix $\mathbf{R} = \boldsymbol{\Sigma}^{-1}$ as:

$$\begin{aligned} Q(\boldsymbol{\mu}', \mathbf{R}'|\boldsymbol{\mu}, \mathbf{R}) \propto & \sum_{t=1}^{T} \sum_{j=1}^{J} \sum_{k=1}^{K} -\frac{\gamma_t(j,k)}{2} \\ & \times \Big(-\log |\mathbf{R}'_{jk}| + (\mathbf{o}_t - \boldsymbol{\mu}'_{jk})^{\mathsf{T}} \mathbf{R}'_{jk} (\mathbf{o}_t - \boldsymbol{\mu}'_{jk}) \Big). \end{aligned} \tag{3.117}$$

This equivalent representation is used to make the computation simple in the following sections. Note that Eqs. (3.113)–(3.117) are represented by the following posterior distributions:

$$\xi_t(i,j) = p(s_t = i, s_{t+1} = j|\mathbf{O}, \Theta), \tag{3.118}$$

$$\gamma_t(j,k) = p(s_t = j, v_t = k|\mathbf{O}, \Theta). \tag{3.119}$$

Similarly to $\gamma_t(j) = p(s_t = j|\mathbf{O}, \Theta)$, as discussed in Section 3.3.1, these values are also computed efficiently by using the forward–backward algorithm.

First, $\gamma_t(j)$ has the following relationships with $\gamma_t(j,k)$ and $\xi_t(i,j)$:

$$\begin{aligned} \gamma_t(j) \triangleq p(s_t = j|\mathbf{O}, \Theta) &= \sum_{k=1}^{K} \gamma_t(j,k) \\ &= \sum_{i=1}^{J} \xi_t(i,j). \end{aligned} \tag{3.120}$$

The posterior probability $\gamma_t(j)$ can be calculated by using Eq. (3.63) based on the forward variables and backward variables $\{\alpha_t(j), \beta_t(j)\}$, but this can also be computed from $\xi_t(i,j)$ or $\gamma_t(j,k)$.

https://doi.org/10.1017/CBO9781107295360.004 Published online by Cambridge University Press

The posterior probability $\gamma_t(j,k)$ of occupying state j and Gaussian component k in Eq. (3.119) can be computed by the forward–backward algorithm. By using the sum and product rules, $\gamma_t(j,k)$ is represented by the posterior distribution $p(s_t = j|\mathbf{O},\Theta)$ and joint likelihood function of $p(\mathbf{O}, v_t = k|s_t = j,\Theta)$ as:

$$\begin{aligned}\gamma_t(j,k) &= p(s_t = j|\mathbf{O},\Theta)p(v_t = k|s_t = j,\mathbf{O},\Theta)\\ &= p(s_t = j|\mathbf{O},\Theta)\frac{p(\mathbf{O}, v_t = k|s_t = j,\Theta)}{\sum_{k'=1}^{K} p(\mathbf{O}, v_t = k'|s_t = j,\Theta)}.\end{aligned} \tag{3.121}$$

By using Eq. (3.33) for $p(\mathbf{O}, v_t = k|s_t = j,\Theta)$ and Eq. (3.63) for $p(s_t = j|\mathbf{O},\Theta)$, Eq. (3.121) is represented as follows:

$$\gamma_t(j,k) = \frac{\alpha_t(j)\beta_t(j)}{\sum_{j'=1}^{J}\alpha_t(j')\beta_t(j')}\cdot\frac{\omega_{jk}\mathcal{N}(\mathbf{o}_t|\boldsymbol{\mu}_{jk},\boldsymbol{\Sigma}_{jk})}{\sum_{k'=1}^{K}\omega_{jk'}\mathcal{N}(\mathbf{o}_t|\boldsymbol{\mu}_{jk'},\boldsymbol{\Sigma}_{jk'})}. \tag{3.122}$$

In a similar manner, we express the posterior probability $\xi_t(i,j)$ in Eq. (3.118) by

$$\xi_t(i,j) = \frac{p(s_t = i, s_{t+1} = j,\mathbf{O}|\Theta)}{\sum_{i'=1}^{J}\sum_{j'=1}^{J} p(s_t = i', s_{t+1} = j',\mathbf{O}|\Theta)}. \tag{3.123}$$

Now we focus on the joint likelihood function $p(s_t = i, s_{t+1} = j,\mathbf{O}|\Theta)$, which is factorized by

$$\begin{aligned}&p(s_t = i, s_{t+1} = j,\mathbf{O}|\Theta)\\ &\quad= p(\mathbf{O}|s_t = i, s_{t+1} = j,\Theta)p(s_t = i, s_{t+1} = j|\Theta)\\ &\quad= p(\mathbf{O}|s_t = i, s_{t+1} = j,\Theta)p(s_{t+1} = j|s_t = i,\Theta)p(s_t = i|\Theta).\end{aligned} \tag{3.124}$$

By using the conditional independence assumption of the HMM, we can rewrite the equation as:

$$\begin{aligned}&p(s_t = i, s_{t+1} = j,\mathbf{O}|\Theta)\\ &\quad= p(\mathbf{o}_1,\cdots,\mathbf{o}_t|s_t = i,\Theta)p(\mathbf{o}_{t+1},\cdots,\mathbf{o}_T|s_t = i, s_{t+1} = j,\Theta)\\ &\qquad\times p(s_{t+1} = j|s_t = i,\Theta)p(s_t = i|\Theta)\\ &\quad= p(\mathbf{o}_1,\cdots,\mathbf{o}_t|s_t = i,\Theta)p(\mathbf{o}_{t+1}|s_{t+1} = j,\Theta)p(\mathbf{o}_{t+2},\cdots,\mathbf{o}_T|s_{t+1} = j,\Theta)\\ &\qquad\times p(s_{t+1} = j|s_t = i,\Theta)p(s_t = i|\Theta)\\ &\quad= \underbrace{p(\mathbf{o}_1,\cdots,\mathbf{o}_t, s_t = i,\Theta)}_{=\alpha_t(i)}\underbrace{p(\mathbf{o}_{t+1}|s_{t+1} = j,\Theta)}_{=b_j(\mathbf{o}_{t+1})}\underbrace{p(\mathbf{o}_{t+2},\cdots,\mathbf{o}_T|s_{t+1} = j,\Theta)}_{=\beta_{t+1}(j)}\\ &\qquad\times\underbrace{p(s_{t+1} = j|s_t = i,\Theta)}_{=a_{ij}}.\end{aligned} \tag{3.125}$$

Thus, by using the forward variable $\alpha_t(i)$ in Eq. (3.50) and the backward variable $\beta_{t+1}(j)$ in Eq. (3.55), Eq. (3.123) is finally written as

https://doi.org/10.1017/CBO9781107295360.004 Published online by Cambridge University Press

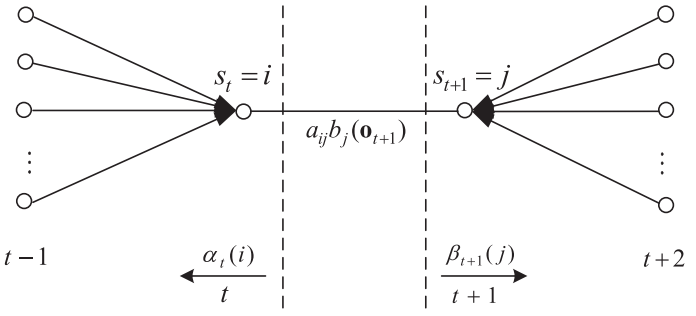

Figure 3.7 Calculation of posterior probability $\xi_t(i,j)$ based on forward variable $\alpha_t(i)$ and backward variable $\beta_{t+1}(j)$. Adapted from Rabiner & Juang (1993).

$$\begin{aligned}\xi_t(i,j) &= \frac{\alpha_t(i) a_{ij} b_j(\mathbf{o}_{t+1}) \beta_{t+1}(j)}{\sum_{i'=1}^{J} \sum_{j'=1}^{J} \alpha_t(i') a_{i'j'} b_{j'}(\mathbf{o}_{t+1}) \beta_{t+1}(j')} \\ &= \frac{\alpha_t(i) a_{ij} \left(\sum_{k=1}^{K} \omega_{jk} \mathcal{N}(\mathbf{o}_{t+1} | \boldsymbol{\mu}_{jk}, \boldsymbol{\Sigma}_{jk}) \right) \beta_{t+1}(j)}{\sum_{i'=1}^{J} \sum_{j'=1}^{J} \alpha_t(i') a_{i'j'} \left(\sum_{k=1}^{K} \omega_{j'k} \mathcal{N}(\mathbf{o}_{t+1} | \boldsymbol{\mu}_{j'k}, \boldsymbol{\Sigma}_{j'k}) \right) \beta_{t+1}(j')}. \end{aligned} \tag{3.126}$$

Figure 3.7 illustrates how the posterior probability $\xi_t(i,j)$ of visiting states i and j in consecutive time frames is calculated by applying the forward–backward algorithm.

In Eq. (3.63), Eq. (3.122) and Eq. (3.123), the posterior probabilities $\gamma_t(j)$, $\gamma_t(j,k)$, and $\xi_t(i,j)$ are calculated, respectively, through a kind of soft computation, i.e., the assignment information is represented by the probabilistic values for the elements i, j, and k. Instead of this soft computation, a simple and efficient approximation is to find the *segmental ML estimates* based on a hard computation where only the single best state sequence $\hat{S} = \{\hat{s}_t\}$ and mixture component sequence $\hat{V} = \{\hat{v}_t\}$ are considered with 0 (not assigned) or 1 (assigned) values for the elements i, j, and k. This computation complexity is significantly reduced. For this consideration, the calculation of posterior probabilities is simplified as

$$\gamma_t(j) = \delta(\hat{s}_t, j), \tag{3.127}$$

$$\gamma_t(j,k) = \delta(\hat{s}_t, j)\delta(\hat{v}_t, k), \tag{3.128}$$

$$\xi_t(i,j) = \delta(\hat{s}_t, i)\delta(\hat{s}_{t+1}, j), \tag{3.129}$$

where $\delta(a, a')$ denotes a Kronecker delta function that returns 1 when $a = a'$ and 0 otherwise. These probabilities are 1 for the cases of the best states i, j, and Gaussians k, and 0 for all of the other cases. Note that $\hat{s}_t$ is computed by using the Viterbi algorithm that maximizes the following segmental joint likelihood function as discussed in Section 3.3.2:

$$\hat{S} = \{\hat{s}_1, \cdots, \hat{s}_T\} = \arg\max_{S} p(S, \mathbf{O}|\Theta). \tag{3.130}$$

The value of $\hat{v}_t$, given HMM state j, is computed by

$$\hat{v}_t = \arg\max_{k} \omega_{jk} \mathcal{N}(\mathbf{o}_t | \boldsymbol{\mu}_{jk}, \boldsymbol{\Sigma}_{jk}). \tag{3.131}$$

The E-step in the EM algorithm is completed by calculating these posterior probabilities.

https://doi.org/10.1017/CBO9781107295360.004 Published online by Cambridge University Press

3.4.3 Maximization step

In the maximization step, we aim to maximize $Q(\boldsymbol{\pi}'|\boldsymbol{\pi})$, $Q(A'|A)$, $Q(\boldsymbol{\omega}'|\boldsymbol{\omega})$, and $Q(\boldsymbol{\mu}', \boldsymbol{\Sigma}'|\boldsymbol{\mu}, \boldsymbol{\Sigma})$ with respect to π', A', $\boldsymbol{\omega}'$, and $\{\boldsymbol{\mu}', \boldsymbol{\Sigma}'\}$ so as to estimate ML parameters π', A', $\boldsymbol{\omega}'$, and $\{\boldsymbol{\mu}', \boldsymbol{\Sigma}'\}$, respectively. However, the constraints of probability parameters

$$\sum_{j=1}^{J} \pi'_j = 1, \quad \sum_{j=1}^{J} a'_{ij} = 1, \forall i, \quad \sum_{k=1}^{K} \omega'_{jk} = 1, \forall j, \tag{3.132}$$

have to be imposed in the constrained optimization problem. For example, when considering the estimation of initial state probabilities $\boldsymbol{\pi}' = \{\pi'_j\}$, we construct a Lagrange function (or Lagrangian):

$$\widetilde{Q}(\boldsymbol{\pi}'|\boldsymbol{\pi}) = \sum_{j=1}^{J} \gamma_1(j) \log \pi'_j + \eta \left(\sum_{j=1}^{J} \pi'_j - 1 \right), \tag{3.133}$$

by combining the original auxiliary function in Eq. (3.113) and the constraint in Eq. (3.132) with an additional Lagrange multiplier η as a scaling factor. Then we differentiate this Lagrangian with respect to individual probability parameter π'_j and set it to zero to obtain

$$\begin{aligned} &\frac{\partial \widetilde{Q}(\boldsymbol{\pi}'|\boldsymbol{\pi})}{\partial \pi'_j} = \gamma_1(j) \frac{1}{\pi'_j} + \eta = 0 \\ &\Rightarrow \quad \hat{\pi}'_j = -\frac{1}{\eta} \gamma_1(j). \end{aligned} \tag{3.134}$$

By substituting Eq. (3.134) into the constrains in Eq. (3.132), we obtain

$$\begin{aligned} &\sum_{j=1}^{J} \pi'_j = \sum_{j=1}^{J} \left(-\frac{1}{\eta} \right) \gamma_1(j) = 1 \\ &\Rightarrow \quad \eta = -\sum_{j=1}^{J} \gamma_1(j). \end{aligned} \tag{3.135}$$

The ML estimate of new initial state probability is derived by substituting Eq. (3.135) into Eq. (3.134):

$$\hat{\pi}'_j = \frac{\gamma_1(j)}{\sum_{j'=1}^{J} \gamma_1(j')} = \gamma_1(j). \tag{3.136}$$

In the same manner, we derive the ML estimates of new state transition probability and new mixture component probability, which are provided by

$$\hat{a}'_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \sum_{i'=1}^{J} \xi_t(i',j)} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(j)}, \tag{3.137}$$

$$\hat{\omega}'_{jk} = \frac{\sum_{t=1}^{T} \gamma_t(j,k)}{\sum_{t=1}^{T} \sum_{k'=1}^{K} \gamma_t(j,k')} = \frac{\sum_{t=1}^{T} \gamma_t(j,k)}{\sum_{t=1}^{T} \gamma_t(j)}. \tag{3.138}$$

https://doi.org/10.1017/CBO9781107295360.004 Published online by Cambridge University Press

These results show that the initial state probability, state transition probability, and mixture weight can be computed using the ratio of the occupation statistics.

On the other hand, when estimating new HMM mean vectors $\boldsymbol{\mu}' = \{\boldsymbol{\mu}'_{jk}\}$, we individually differentiate $Q(\boldsymbol{\mu}', \boldsymbol{\Sigma}'|\boldsymbol{\mu}, \boldsymbol{\Sigma})$ in Eq. (3.117) with respect to $\boldsymbol{\mu}'_{jk}$ for each Gaussian component k in state j and set it to zero:

$$\begin{aligned}
&\frac{\partial Q(\boldsymbol{\mu}', \boldsymbol{\Sigma}'|\boldsymbol{\mu}, \boldsymbol{\Sigma})}{\partial \boldsymbol{\mu}'_{jk}} \\
&= \frac{\partial}{\partial \boldsymbol{\mu}'_{jk}} \sum_{t=1}^{T} \sum_{j'=1}^{J} \sum_{k'=1}^{K} -\frac{\gamma_t(j', k')}{2} \left(\log \left(|\boldsymbol{\Sigma}'_{j'k'}| \right) + (\mathbf{o}_t - \boldsymbol{\mu}'_{j'k'})^\top (\boldsymbol{\Sigma}'_{j'k'})^{-1} (\mathbf{o}_t - \boldsymbol{\mu}'_{j'k'}) \right) \\
&= \frac{\partial}{\partial \boldsymbol{\mu}'_{jk}} \sum_{t=1}^{T} -\frac{\gamma_t(j, k)}{2} (\mathbf{o}_t - \boldsymbol{\mu}'_{jk})^\top (\boldsymbol{\Sigma}'_{jk})^{-1} (\mathbf{o}_t - \boldsymbol{\mu}'_{jk}) \\
&= (\boldsymbol{\Sigma}'_{jk})^{-1} \sum_{t=1}^{T} \gamma_t(j, k)(\mathbf{o}_t - \boldsymbol{\mu}'_{jk}) = 0,
\end{aligned} \tag{3.139}$$

where we use the following vector derivative rule in Eq. (B.9):

$$\frac{\partial \mathbf{a}^\top \mathbf{b}}{\partial \mathbf{a}} = \frac{\partial \mathbf{b}^\top \mathbf{a}}{\partial \mathbf{a}} = \mathbf{b}. \tag{3.140}$$

Therefore, the ML estimate of the HMM mean vector is derived as shown by

$$\begin{aligned}
&\sum_{t=1}^{T} \gamma_t(j, k)(\mathbf{o}_t - \boldsymbol{\mu}'_{jk}) = 0 \\
&\Rightarrow \quad \hat{\boldsymbol{\mu}}'_{jk} = \frac{\sum_{t=1}^{T} \gamma_t(j, k) \mathbf{o}_t}{\sum_{t=1}^{T} \gamma_t(j, k)}.
\end{aligned} \tag{3.141}$$

Note that the mean vector is represented as the first-order expected value of $\mathbf{o}_t$ by using the occupation probability of $\gamma_t(j, k)$.

At the same time, the new HMM covariance matrices $\boldsymbol{\Sigma}' = \{\boldsymbol{\Sigma}'_{jk}\}$ or their inverse matrices $(\boldsymbol{\Sigma}')^{-1} = \{(\boldsymbol{\Sigma}'_{jk})^{-1} \triangleq \mathbf{R}'_{jk}\}$ (also called the precision matrices) are estimated by differentiation of $Q(\boldsymbol{\mu}', \mathbf{R}'|\boldsymbol{\mu}, \mathbf{R})$ in Eq. (3.117) with respect to $\mathbf{R}'_{jk}$ for each Gaussian k at each state j and setting it to zero:

$$\begin{aligned}
&\frac{\partial Q(\boldsymbol{\mu}', \mathbf{R}'|\boldsymbol{\mu}, \mathbf{R})}{\partial \mathbf{R}'_{jk}} \\
&= \frac{\partial}{\partial \mathbf{R}'_{jk}} \sum_{t=1}^{T} -\frac{\gamma_t(j, k)}{2} \left(-\log \left(|\mathbf{R}'_{jk}| \right) + (\mathbf{o}_t - \boldsymbol{\mu}'_{jk})^\top \mathbf{R}'_{jk} (\mathbf{o}_t - \boldsymbol{\mu}'_{jk}) \right) \\
&= \frac{1}{2} \sum_{t=1}^{T} \gamma_t(j, k)(\mathbf{R}'_{jk})^{-1} - \frac{1}{2} \sum_{t=1}^{T} \gamma_t(j, k)(\mathbf{o}_t - \boldsymbol{\mu}'_{jk})(\mathbf{o}_t - \boldsymbol{\mu}'_{jk})^\top = 0,
\end{aligned} \tag{3.142}$$

https://doi.org/10.1017/CBO9781107295360.004 Published online by Cambridge University Press

where we use the following matrix derivative rule in Eqs. (B.8) and (B.10):

$$\frac{\partial \log |\mathbf{A}|}{\partial \mathbf{A}} = \mathbf{A}^{-1}, \tag{3.143}$$

$$\frac{\partial \mathbf{a}^\top \mathbf{C}\mathbf{b}}{\partial \mathbf{C}} = \mathbf{a}\mathbf{b}^\top. \tag{3.144}$$

Thus, the new estimates of the HMM precision and covariance matrices are derived by using the ML estimate $\hat{\boldsymbol{\mu}}'_{jk}$ for $\boldsymbol{\mu}'_{jk}$ in Eq. (3.142) as

$$\begin{aligned}\hat{\mathbf{R}}'_{jk} &= \left(\frac{\sum_{t=1}^{T} \gamma_t(j,k)(\mathbf{o}_t - \hat{\boldsymbol{\mu}}'_{jk})(\mathbf{o}_t - \hat{\boldsymbol{\mu}}'_{jk})^\top}{\sum_{t=1}^{T} \gamma_t(j,k)} \right)^{-1}, \\ \hat{\boldsymbol{\Sigma}}'_{jk} &= \frac{\sum_{t=1}^{T} \gamma_t(j,k)(\mathbf{o}_t - \hat{\boldsymbol{\mu}}'_{jk})(\mathbf{o}_t - \hat{\boldsymbol{\mu}}'_{jk})^\top}{\sum_{t=1}^{T} \gamma_t(j,k)}.\end{aligned} \tag{3.145}$$

Interestingly, the calculation of the derived covariance matrices $\boldsymbol{\Sigma} = \{\boldsymbol{\Sigma}_{jk}\}$ in Eq. (3.145) is interpreted as the weighted ensemble expectation and covariance matrices, as well as the calculation of the ML mean vectors $\boldsymbol{\mu} = \{\boldsymbol{\mu}_{jk}\}$ in Eq. (3.141). The occupation probability $\gamma_t(j,k)$ of state j and mixture component k at time t is treated as the weighting factor in calculation of the weighted expectation function. Note that Eq. (3.145) is

$$\begin{aligned}\hat{\boldsymbol{\Sigma}}'_{jk} &= \frac{\sum_{t=1}^{T} \gamma_t(j,k)\mathbf{o}_t\mathbf{o}_t^\top}{\sum_{t=1}^{T} \gamma_t(j,k)} - 2\frac{\sum_{t=1}^{T} \gamma_t(j,k)\mathbf{o}_t(\hat{\boldsymbol{\mu}}'_{jk})^\top}{\sum_{t=1}^{T} \gamma_t(j,k)} + \hat{\boldsymbol{\mu}}'_{jk}(\hat{\boldsymbol{\mu}}'_{jk})^\top \\ &= \frac{\sum_{t=1}^{T} \gamma_t(j,k)\mathbf{o}_t\mathbf{o}_t^\top}{\sum_{t=1}^{T} \gamma_t(j,k)} - 2\hat{\boldsymbol{\mu}}'_{jk}(\hat{\boldsymbol{\mu}}'_{jk})^\top + \hat{\boldsymbol{\mu}}'_{jk}(\hat{\boldsymbol{\mu}}'_{jk})^\top \\ &= \frac{\sum_{t=1}^{T} \gamma_t(j,k)\mathbf{o}_t\mathbf{o}_t^\top}{\sum_{t=1}^{T} \gamma_t(j,k)} - \hat{\boldsymbol{\mu}}'_{jk}(\hat{\boldsymbol{\mu}}'_{jk})^\top.\end{aligned} \tag{3.146}$$

Thus, the ML estimate of the covariance matrix is computed from the second-order statistic and the ML estimate of the mean vector.

Now, we summarize the ML estimates of the CDHMM as follows:

$$\hat{\pi}'_j = \gamma_1(j), \tag{3.147}$$

$$\hat{a}'_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \sum_{j'=1}^{J} \xi_t(i,j')}, \tag{3.148}$$

$$\hat{\omega}'_{jk} = \frac{\sum_{t=1}^{T} \gamma_t(j,k)}{\sum_{t=1}^{T} \sum_{k'=1}^{K} \gamma_t(j,k')}, \tag{3.149}$$

$$\hat{\boldsymbol{\mu}}'_{jk} = \frac{\sum_{t=1}^{T} \gamma_t(j,k)\mathbf{o}_t}{\sum_{t=1}^{T} \gamma_t(j,k)}, \tag{3.150}$$

$$\hat{\boldsymbol{\Sigma}}'_{jk} = \frac{\sum_{t=1}^{T} \gamma_t(j,k)\mathbf{o}_t\mathbf{o}_t^\top}{\sum_{t=1}^{T} \gamma_t(j,k)} - \hat{\boldsymbol{\mu}}'_{jk}(\hat{\boldsymbol{\mu}}'_{jk})^\top. \tag{3.151}$$

https://doi.org/10.1017/CBO9781107295360.004 Published online by Cambridge University Press

If we consider the diagonal covariance matrix, the dth diagonal element of Eq. (3.151) is modified as follows:

$$\hat{\Sigma}'_{jkd} = \frac{\sum_{t=1}^{T} \gamma_t(j,k) o_{td}^2}{\sum_{t=1}^{T} \gamma_t(j,k)} - (\hat{\mu}'_{jkd})^2. \tag{3.152}$$

To compute these estimated values, we need to compute the following values:

$$\begin{aligned}
\xi(i,j) &\triangleq \sum_{t=1}^{T-1} \xi_t(i,j) \\
\gamma(j,k) &\triangleq \sum_{t=1}^{T} \gamma_t(j,k) \\
\boldsymbol{\gamma}_{jk} &\triangleq \sum_{t=1}^{T} \gamma_t(j,k) \mathbf{o}_t \\
\boldsymbol{\Gamma}_{jk} &\triangleq \sum_{t=1}^{T} \gamma_t(j,k) \mathbf{o}_t \mathbf{o}_t^{\mathsf{T}}.
\end{aligned} \tag{3.153}$$

These statistics are sufficient to compute the parameters, and are called *sufficient statistics* of the CDHMM. In particular, $\gamma(j,k)$, $\boldsymbol{\gamma}_{jk}$, and $\boldsymbol{\Gamma}_{jk}$ are called the 0th, 1st, and 2nd order sufficient statistics of the Gaussian at HMM state j and mixture component k, respectively. The sufficient statistic is first mentioned in Section 2.1.3, where the estimation problems are rather simple, and they do not include latent variables. In the latent models, the probabilistic assignment information of the occupation probabilities $\gamma_1(j)$, $\gamma_t(j,k)$, $\xi_t(i,j)$ is important to obtain the sufficient statistics. These statistics, composed of the occupation probabilities $\gamma_1(j)$, $\gamma_t(j,k)$, $\xi_t(i,j)$, are computed from the forward–backward algorithm, as discussed in the expectation step (Section 3.4.2).

Based on these sufficient statistics, we can rewrite the auxiliary function as follows:

$$Q(\boldsymbol{\pi}'|\boldsymbol{\pi}) = \sum_{j=1}^{J} \gamma_1(j) \log \pi'_j, \tag{3.154}$$

$$Q(\mathbf{A}'|\mathbf{A}) = \sum_{i=1}^{J} \sum_{j=1}^{J} \xi(i,j) \log a'_{ij}, \tag{3.155}$$

$$Q(\boldsymbol{\omega}'|\boldsymbol{\omega}) = \sum_{j=1}^{J} \sum_{k=1}^{K} \gamma(j,k) \log \omega'_{jk}, \tag{3.156}$$

$$\begin{aligned}
Q(\boldsymbol{\mu}', \boldsymbol{\Sigma}'|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{j=1}^{J} \sum_{k=1}^{K} -\frac{\gamma(j,k)}{2} \Big(D \log(2\pi) + \log|\boldsymbol{\Sigma}'_{jk}| + \boldsymbol{\mu}'^{\mathsf{T}}_{jk} (\boldsymbol{\Sigma}'_{jk})^{-1} \boldsymbol{\mu}'_{jk} \Big) \\
+ \boldsymbol{\mu}'^{\mathsf{T}}_{jk} (\boldsymbol{\Sigma}'_{jk})^{-1} \boldsymbol{\gamma}_{jk} - \frac{1}{2} \mathrm{tr} \Big[(\boldsymbol{\Sigma}'_{jk})^{-1} \boldsymbol{\Gamma}_{jk} \Big].
\end{aligned} \tag{3.157}$$

These forms are used in the analytical discussions in later chapters.

https://doi.org/10.1017/CBO9781107295360.004 Published online by Cambridge University Press

This total EM algorithm for iteratively estimating the HMM parameters is called the *Baum–Welch algorithm* (Baum, Petrie, Soules *et al*. 1970), and is based on the forward-backward algorithm, the accumulation of the sufficient statistics, and the update of the HMM parameters, as shown in Algorithm 4. The Baum–Welch algorithm can be extended based on the Bayesian approach (see Section 4.3 and Section 7.3).

Algorithm 4 Baum–Welch algorithm

Require: $\Theta \leftarrow \Theta^{\text{init}}$

1: **repeat**
2: Compute the forward variable $\alpha_t(j)$ from the forward algorithm
3: Compute the backward variable $\beta_t(j)$ from the backward algorithm
4: Compute the occupation probabilities $\gamma_1(j)$, $\gamma_t(j,k)$, and $\xi_t(i,j)$
5: Accumulate the sufficient statistics $\xi(i,j)$, $\gamma(j,k)$, $\boldsymbol{\gamma}_{jk}$, and $\boldsymbol{\Gamma}_{jk}$
6: Estimate the new HMM parameters $\hat{\Theta}'$
7: Update the HMM parameters $\Theta \leftarrow \hat{\Theta}'$
8: **until** Convergence

In the implementation of Viterbi or segmental ML estimation, we apply the current HMM parameter estimates $\Theta = \{\pi_j, a_{ij}, \omega_{jk}, \boldsymbol{\mu}_{jk}, \boldsymbol{\Sigma}_{jk}\}$ and the Viterbi algorithm to perform Viterbi decoding to find the state alignment. Given the best state sequence $\hat{S}$ and mixture component sequence $\hat{V}$, new ML estimates $\Theta' = \{\boldsymbol{\mu}'_{jk}, \boldsymbol{\Sigma}'_{jk}\}$ are computed as the ensemble mean vector and covariance matrix, where $\sum_{t=1}^{T} \gamma_t(j,k)$ is seen as the count N_{jk} of training samples $\mathbf{O} = \{\mathbf{o}_t\}$ which are aligned in state j and Gaussian component k (Juang & Rabiner 1990). This method of using the Viterbi training instead of the forward–backward algorithm to obtain (a part of) occupation probabilities in the training step is called *segmental K-means training* or *Viterbi training*.

We should note that to make the Baum–Welch algorithm work in real speech data, we need to consider some heuristics in the ML EM algorithm. For example, how to provide initial values is always an important question, and it is usually resolved by using a simple algorithm (e.g., K-means clustering or random initialization). The other important issue comes from the data sparseness, e.g., some mixture components or hidden states cannot have sufficient data assigned in the Viterbi or forward–backward algorithm, which makes the parameter estimation (especially covariance parameter estimation) unstable. This can be heuristically avoided by setting a threshold to update these parameters, or setting minimum threshold values for covariance parameters. These problems can be solved by the Bayesian approaches.

Despite the above problems, the Baum–Welch algorithm based on the EM algorithm is widely used in current CDHMM training. This Baum–Welch algorithm has several advantages. For example, the algorithm is unique in that the computational cost of the E step is much more than that of the M-step, since the E step computational cost depends on the training data size. However, the E-step can be parallelized with many computers by distributing a chunk of data to a computer and computing a sufficient statistic of the chunk independently of the other computers. Therefore, the Baum–Welch algorithm

https://doi.org/10.1017/CBO9781107295360.004 Published online by Cambridge University Press

has a very nice data scalability, which enables it to use a large amount of training data. In addition, within the algorithm, the likelihood always increases by the EM theory. Therefore, by monitoring the likelihood values, we can easily detect errors and bugs during a training procedure.

3.5 Maximum likelihood linear regression for hidden Markov model

As we discussed in the previous section, CDHMM parameters can be estimated by statistical ML methods, the effectiveness of which depends on the quality and quantity of available data that should distribute according to the statistical features of the intended signal space or conditions. As there is no sure way of collecting sufficient data to cover all conditions, adaptive training of HMM parameters from a set of previously obtained parameters to a new set that befits a specific environment with a small amount of new data is an important research issue.

In speech recognition, one approach is to view the adaptation of model parameters to new data (e.g., speaker adaptation) as a transformation problem; that is, the new set of model parameters is a transformed version of the old set: $\Theta_{n+1} = f(\Theta_n, \{\mathbf{o}\}_n)$, where $\{\mathbf{o}\}_n$ denotes the new set of data available at moment n for the existing model parameters Θ_n to adapt to. Most frequently and practically, the function f is chosen to be of an affine transformation type (Digalakis, Ritischev & Neumeyer 1995, Leggetter & Woodland 1995):

$$\boldsymbol{\theta}_{n+1} = \mathbf{A}\boldsymbol{\theta}_n + \mathbf{b}, \tag{3.158}$$

where various parts of the model parameters, e.g., the mean vectors or the variances, are envisaged in a vector space. The adaptation algorithm therefore involves deriving the affine map components, $\mathbf{A}$ and $\mathbf{b}$, from the adaptation data $\{\mathbf{o}\}_n$. A number of algorithms have been proposed for this purpose (see Lee & Huo (2000), and Shinoda (2010) for details).

The linear regression method for HMM parameters estimates the affine transformation parameters from a set of adaptation data, usually limited in size. The transformation with the estimated parameters is then applied to the previously trained HMMs, resulting in the set of "adapted models." Note that for automatic speech recognition, the number of the Gaussian distributions or simply Gaussians, which are used as component distributions in forming state-dependent mixture distributions, is typically in the thousands or more. If each mean vector in the set of Gaussians is to be modified by a unique transformation matrix, the number of "adaptation parameters" can be quite large. The main problem of this method is thus how to improve "generalization capability" by avoiding the over-training problem when the amount of adaptation data is small. To solve the problem, we introduce a model selection approach.

The model selection approach was originally proposed within the estimation of linear transformation parameters by using the maximum likelihood EM algorithm, as discussed in Section 3.4. The technique is called maximum likelihood linear regression (MLLR). MLLR proposes to share one linear transformation in a cluster of many Gaussians in the HMM set, thereby effectively reducing the number of free parameters that can then be trained with a small amount of adaptation data. The Gaussian clusters

https://doi.org/10.1017/CBO9781107295360.004 Published online by Cambridge University Press

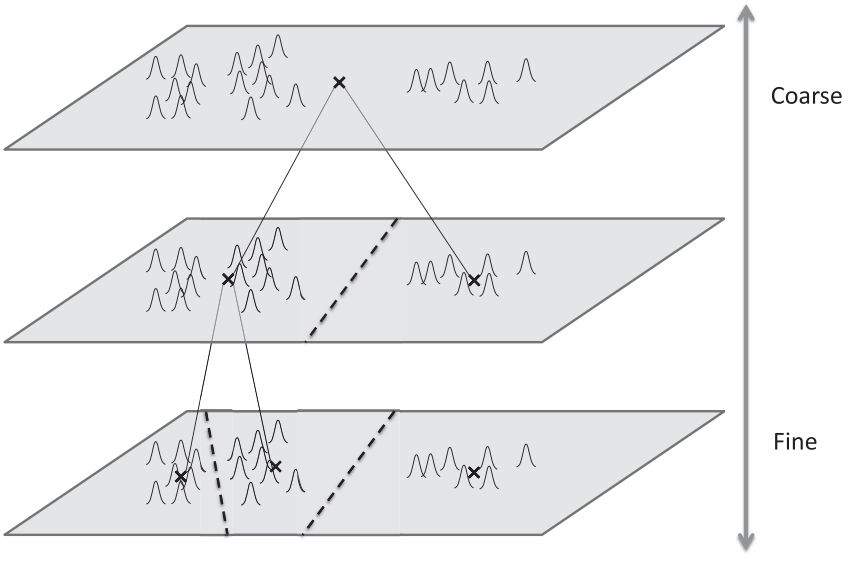

Figure 3.8 Gaussian tree representation of linear regression parameters.

are usually constructed as a tree structure, as shown in Figure 3.8, which is predetermined and fixed throughout adaptation. This tree (called a regression tree) is constructed based on a centroid splitting algorithm, described in Young, Evermann, Gales *et al*. (2006). This algorithm first makes two centroid vectors from a random perturbation of the global mean vector computed from Gaussians assigned to a target leaf node. Then it splits a set of these Gaussians according to the Euclidean distance between Gaussian mean vectors and two centroid vectors. The two sets of Gaussians obtained are assigned to child nodes, and this procedure is continued to finally build a tree.

The utility of the tree structure is commensurate with the amount of adaptation data; namely, if we have a small amount of data, it uses only coarse clusters (e.g., the root node of a tree in the top layer of Figure 3.8) where the number of free parameters in the linear transformation matrices is small. On the other hand, if we have a sufficiently large amount of data, it can use fine clusters where the number of free parameters in the linear transformation matrices is large, potentially improving the precision of the estimated parameters. This framework needs to select appropriate Gaussian clusters according to the amount of data, i.e., it needs an appropriate model selection function. Usually, model selection is performed by setting a threshold value manually (e.g., the total number of speech frames assigned to a set of Gaussians in a node).

3.5.1 Linear regression for hidden Markov models

This section briefly explains a solution for the linear regression parameters for HMMs within a maximum likelihood EM algorithm framework. It uses a solution based on a

https://doi.org/10.1017/CBO9781107295360.004 Published online by Cambridge University Press

variance normalized representation of Gaussian mean vectors to simplify the solution.[3] In this section, we only focus on the transformation of Gaussian mean vectors in CDHMMs.

First, we review the basic EM algorithm of the conventional HMM parameter estimation, as discussed in Section 3.4, to set the notational convention and to align with the subsequent development of the MLLR approach. Let $\mathbf{O} \triangleq \{\mathbf{o}_t \in \mathbb{R}^D | t = 1, \cdots, T\}$ be a sequence of D dimensional feature vectors for T speech frames. The latent variables in a continuous density HMM are composed of HMM states and mixture components of GMMs. A sequence of HMM states is represented by $S \triangleq \{s_t | t = 1, \cdots, T\}$, where the value of s_t denotes an HMM state index at frame t. Similarly, a sequence of mixture components is represented by $V \triangleq \{v_t | t = 1, \cdots, T\}$, where the value of v_t denotes a mixture component index at frame t. As introduced in Eq. (3.78), the EM algorithm deals with the following auxiliary function as an optimization function instead of directly using the model likelihood:

$$\begin{aligned} Q(\Theta'|\Theta) &= \mathbb{E}_{(S,V)}[\log p(\mathbf{O}, S, V|\Theta')|\mathbf{O}, \Theta] \\ &= \sum_S \sum_V p(S, V|\mathbf{O}, \Theta) \log p(\mathbf{O}, S, V|\Theta'), \end{aligned} \tag{3.159}$$

where Θ is a set of HMM parameters and $p(\mathbf{O}, S, V|\Theta)$ is a complete data likelihood given Θ. $p(S, V|\mathbf{O}, \Theta)$ is the posterior distribution of the latent variables given the previously estimated HMM parameters Θ. Equation (3.78) is an expected value, and is efficiently computed by using the forward–backward algorithm as the E-step of the EM algorithm, as we discussed in Section 3.3.

The M-step of the EM algorithm estimates HMM parameters, as follows:

$$\Theta^{\mathrm{ML}} = \arg\max_{\Theta'} Q(\Theta'|\Theta). \tag{3.160}$$

The E-step and the M-step are performed iteratively until convergence, and finally we obtain the HMM parameters as a close approximation of the stationary point solution.

Now we focus on the linear transformation parameters within the EM algorithm. We prepare a transformation parameter matrix $\mathbf{W}_j \in \mathbb{R}^{D \times (D+1)}$ for each leaf node j in a Gaussian tree. Here, we assume that the Gaussian tree is pruned by a model selection approach as a model structure M, and the set of leaf nodes in the pruned tree is represented by $\mathcal{J}_M$. Hereinafter, we use Z to denote a joint event of S and V (i.e., $Z \triangleq \{S, V\}$). This will much simplify the following development pertaining to the adaptation of the mean and the covariance parameters. Similarly to Eq. (3.159), the auxiliary function with respect to a set of transformation parameters $\mathbf{\Lambda}_{\mathcal{J}_M} = \{\mathbf{W}_j | j = 1, \cdots, |\mathcal{J}_M|\}$ can be represented as follows:

$$\begin{aligned} Q(\mathbf{\Lambda}'_{\mathcal{J}_M}|\mathbf{\Lambda}_{\mathcal{J}_M}) &= \mathbb{E}_{(Z)}\left[\log p(\mathbf{O}, Z|\mathbf{\Lambda}'_{\mathcal{J}_M}, \Theta)\right] \\ &= \sum_{k=1}^{K} \sum_{t=1}^{T} \gamma_t(k) \log \mathcal{N}(\mathbf{o}_t|\boldsymbol{\mu}_k^{ad'}, \mathbf{\Sigma}_k). \end{aligned} \tag{3.161}$$

[3] This is first described in Gales & Woodland (1996) as normalized domain MLLR. The structural Bayes approach Shinoda & Lee (2001) for bias vector estimation in HMM adaptation also uses this normalized representation.

https://doi.org/10.1017/CBO9781107295360.004 Published online by Cambridge University Press

Here k denotes a unique mixture component index of all Gaussians in the target HMMs (for all phoneme HMMs in a speech recognition case), and K is the total number of Gaussians. $\gamma_t(k) \triangleq p(v_t = k|\mathbf{O}; \Theta, \mathbf{\Lambda}_{\mathcal{J}_M})$ is the posterior probability of mixture component k at t, derived from the previously estimated transformation parameters $\mathbf{\Lambda}_{\mathcal{J}_M}$.[4] Expression $\boldsymbol{\mu}_k^{ad}$ is a transformed mean vector with $\mathbf{\Lambda}_{\mathcal{J}_M}$, and the concrete form of this vector is discussed in the next paragraph. In the Q function, we disregard the parameters of the state transition probabilities and the mixture weights, since they do not depend on the optimization with respect to $\mathbf{\Lambda}_{\mathcal{J}_M}$. Expression $\mathcal{N}(\cdot|\boldsymbol{\mu}, \boldsymbol{\Sigma})$ denotes a Gaussian distribution with mean parameter $\boldsymbol{\mu}$ and covariance matrix parameter $\boldsymbol{\Sigma}$, and is defined in Appendix C.6 as follows:

$$\mathcal{N}(\mathbf{o}_t|\boldsymbol{\mu}_k^{ad}, \boldsymbol{\Sigma}_k) \triangleq C_{\mathcal{N}}(\boldsymbol{\Sigma}_k) \exp\left(-\frac{1}{2}\mathrm{tr}\left[(\boldsymbol{\Sigma}_k)^{-1}(\mathbf{o}_t - \boldsymbol{\mu}_k^{ad})(\mathbf{o}_t - \boldsymbol{\mu}_k^{ad})^{\mathsf{T}}\right]\right). \quad (3.162)$$

We use the trace based representation. Factor $C_{\mathcal{N}}(\boldsymbol{\Sigma}_k)$ is a normalization factor, and is defined as follows:

$$C_{\mathcal{N}}(\boldsymbol{\Sigma}_k) \triangleq (2\pi)^{-\frac{D}{2}}|\boldsymbol{\Sigma}_k|^{-\frac{1}{2}}. \quad (3.163)$$

In the following paragraphs, we derive Eq. (3.161) as a function of $\mathbf{\Lambda}_{\mathcal{J}_M}$ to optimize $\mathbf{\Lambda}_{\mathcal{J}_M}$.

We consider the concrete form of the transformed mean vector $\boldsymbol{\mu}_k^{\mathrm{ad}}$ based on the variance normalized representation. We first define the Cholesky decomposition matrix $\mathbf{C}_k$ as follows:

$$\boldsymbol{\Sigma}_k \triangleq \mathbf{C}_k(\mathbf{C}_k)^{\mathsf{T}}, \quad (3.164)$$

where $\mathbf{C}_k$ is a $D \times D$ triangular matrix. If the Gaussian k is included in a set of Gaussians $\mathcal{K}_j$ in leaf node j (i.e., $k \in \mathcal{K}_j$), the affine transformation of a Gaussian mean vector in a covariance normalized space $(\mathbf{C}_k)^{-1}\boldsymbol{\mu}_k^{\mathrm{ad}}$ is represented as follows:

$$\begin{aligned}(\mathbf{C}_k)^{-1}\boldsymbol{\mu}_k^{\mathrm{ad}} &= \mathbf{W}_j \begin{pmatrix} 1 \\ (\mathbf{C}_k)^{-1}\boldsymbol{\mu}_k^{\mathrm{ini}} \end{pmatrix} \\ \Rightarrow \boldsymbol{\mu}_k^{\mathrm{ad}} &= \mathbf{C}_k\mathbf{W}_j \begin{pmatrix} 1 \\ (\mathbf{C}_k)^{-1}\boldsymbol{\mu}_k^{\mathrm{ini}} \end{pmatrix} \triangleq \mathbf{C}_k\mathbf{W}_j\boldsymbol{\xi}_k, \end{aligned} \quad (3.165)$$

where $\boldsymbol{\xi}_k$ is an augmented normalized vector of an initial (non-adapted) Gaussian mean vector $\boldsymbol{\mu}_k^{\mathrm{ini}}$ and j is a leaf node index that holds a set of Gaussians. Thus, transformation parameter $\mathbf{W}_j$ is shared among a set of Gaussians $\mathcal{K}_j$. The clustered structure of the Gaussians is usually represented as a binary tree where a set of Gaussians belongs to each node.

[4] k denotes a combination of all possible HMM states and mixture components. In the common HMM representation (e.g., in this chapter), k can be represented by these two indexes in Eq. (3.109).

https://doi.org/10.1017/CBO9781107295360.004 Published online by Cambridge University Press

Now we focus on how to obtain the Q function of $\mathbf{\Lambda}_{\mathcal{J}_M}$. By following the equations in Example 2.3 with considering the occupation probability $\gamma_t(k)$ in Eq. (2.39), Eq. (3.161) is represented as follows:

$$\begin{aligned}
&Q(\mathbf{\Lambda}'_{\mathcal{J}_M}|\mathbf{\Lambda}_{\mathcal{J}_M}) \\
&\quad = \sum_{j\in\mathcal{J}_M}\sum_{k\in\mathcal{K}_j}\sum_{t=1}^{T}\gamma_t(k)\log\mathcal{N}(\mathbf{o}_t|\boldsymbol{\mu}_k^{\mathrm{ad}'},\mathbf{\Sigma}_k) \\
&\quad = \sum_{j\in\mathcal{J}_M}\sum_{k\in\mathcal{K}_j}\Bigg((\boldsymbol{\mu}_k^{\mathrm{ad}'})^{\mathsf{T}}(\mathbf{\Sigma}_k)^{-1}\sum_{t=1}^{T}\gamma_t(k)\mathbf{o}_t - \frac{1}{2}\mathrm{tr}\left[(\mathbf{\Sigma}_k)^{-1}\sum_{t=1}^{T}\gamma_t(k)\mathbf{o}_t\mathbf{o}_t^{\mathsf{T}}\right] \\
&\qquad\qquad + \sum_{t=1}^{T}\gamma_t(k)\left(\log C_{\mathcal{N}}(\mathbf{\Sigma}_k) - \frac{1}{2}(\boldsymbol{\mu}_k^{\mathrm{ad}'})^{\mathsf{T}}(\mathbf{\Sigma}_k)^{-1}\boldsymbol{\mu}_k^{\mathrm{ad}'}\right)\Bigg) \\
&\quad = \sum_{j\in\mathcal{J}_M}\sum_{k\in\mathcal{K}_j}\Bigg((\boldsymbol{\mu}_k^{\mathrm{ad}'})^{\mathsf{T}}(\mathbf{\Sigma}_k)^{-1}\boldsymbol{\gamma}_k - \frac{1}{2}\mathrm{tr}\left[(\mathbf{\Sigma}_k)^{-1}\mathbf{\Gamma}_k\right] \\
&\qquad\qquad + \gamma_k\left(\log C_{\mathcal{N}}(\mathbf{\Sigma}_k) - \frac{1}{2}(\boldsymbol{\mu}_k^{\mathrm{ad}'})^{\mathsf{T}}(\mathbf{\Sigma}_k)^{-1}\boldsymbol{\mu}_k^{\mathrm{ad}'}\right)\Bigg),
\end{aligned} \tag{3.166}$$

where γ_k, $\boldsymbol{\gamma}_k$, and $\mathbf{\Gamma}_k$ are defined as follows:

$$\begin{cases}
\gamma_k = \displaystyle\sum_{t=1}^{T}\gamma_t(k) \\
\boldsymbol{\gamma}_k = \displaystyle\sum_{t=1}^{T}\gamma_t(k)\mathbf{o}_t \\
\mathbf{\Gamma}_k = \displaystyle\sum_{t=1}^{T}\gamma_t(k)\mathbf{o}_t\mathbf{o}_t^{\mathsf{T}}.
\end{cases} \tag{3.167}$$

As introduced in Eq. (3.153), these are the 0th, 1st, and 2nd order sufficient statistics of Gaussians in HMMs, respectively. Then, the Q function of $\mathbf{\Lambda}_{\mathcal{J}_M}$ is represented by substituting Eq. (3.165) into Eq. (3.166) as follows:

$$\begin{aligned}
&Q(\mathbf{\Lambda}'_{\mathcal{J}_M}|\mathbf{\Lambda}_{\mathcal{J}_M}) \\
&\quad = \sum_{j\in\mathcal{J}_M}\sum_{k\in\mathcal{K}_j}\Bigg((\mathbf{C}_k\mathbf{W}'_j\boldsymbol{\xi}_k)^{\mathsf{T}}(\mathbf{\Sigma}_k)^{-1}\boldsymbol{\gamma}_k - \frac{1}{2}\mathrm{tr}\left[(\mathbf{\Sigma}_k)^{-1}\mathbf{\Gamma}_k\right] \\
&\qquad\qquad + \gamma_k\left(\log C_{\mathcal{N}}(\mathbf{\Sigma}_k) - \frac{1}{2}(\mathbf{C}_k\mathbf{W}'_j\boldsymbol{\xi}_k)^{\mathsf{T}}(\mathbf{\Sigma}_k)^{-1}\mathbf{C}_k\mathbf{W}'_j\boldsymbol{\xi}_k\right)\Bigg) \\
&\quad = \sum_{j\in\mathcal{J}_M}\sum_{k\in\mathcal{K}_j}\Bigg(\mathrm{tr}\left[\mathbf{W}_j'^{\mathsf{T}}(\mathbf{C}_k)^{-1}\boldsymbol{\gamma}_k\boldsymbol{\xi}_k^{\mathsf{T}}\right] - \frac{1}{2}\mathrm{tr}\left[(\mathbf{\Sigma}_k)^{-1}\mathbf{\Gamma}_k\right] \\
&\qquad\qquad + \gamma_k\left(\log C_{\mathcal{N}}(\mathbf{\Sigma}_k) - \frac{1}{2}\mathrm{tr}\left[\mathbf{W}_j'^{\mathsf{T}}\mathbf{W}'_j\boldsymbol{\xi}_k\boldsymbol{\xi}_k^{\mathsf{T}}\gamma_k\right]\right)\Bigg)
\end{aligned}$$

https://doi.org/10.1017/CBO9781107295360.004 Published online by Cambridge University Press

$$= \sum_{j \in \mathcal{J}_M} \left(\sum_{k \in \mathcal{K}_j} \gamma_k \log C_{\mathcal{N}}(\boldsymbol{\Sigma}_k) - \frac{1}{2} \mathrm{tr} \left[\mathbf{W}_j'^{\mathsf{T}} \mathbf{W}_j' \boldsymbol{\Xi}_j - 2 \mathbf{W}_j'^{\mathsf{T}} \mathbf{Z}_j + \sum_{k \in \mathcal{K}_j} (\boldsymbol{\Sigma}_k)^{-1} \boldsymbol{\Gamma}_k \right] \right), \tag{3.168}$$

where $\boldsymbol{\Xi}_j$ and $\mathbf{Z}_j$ are 0th and 1st order sufficient statistics of linear regression parameters defined as:

$$\begin{cases} \boldsymbol{\Xi}_j \triangleq \sum_{k \in \mathcal{K}_j} \boldsymbol{\xi}_k \boldsymbol{\xi}_k^{\mathsf{T}} \gamma_k \\ \mathbf{Z}_j \triangleq \sum_{k \in \mathcal{K}_j} (\mathbf{C}_k)^{-1} \boldsymbol{\gamma}_k \boldsymbol{\xi}_k^{\mathsf{T}}. \end{cases} \tag{3.169}$$

Here $\mathbf{Z}_j$ is a $D \times (D+1)$ matrix and $\boldsymbol{\Xi}_j$ is a $(D+1) \times (D+1)$ symmetric matrix. To derive Eq. (3.168), we use the fact that the trace of the scalar value is equal to the original scalar value, the cyclic property, and the distributive property of the trace as in Appendix B:

$$a = \mathrm{tr}[a], \tag{3.170}$$

$$\mathrm{tr}[\mathbf{ABC}] = \mathrm{tr}[\mathbf{BCA}], \tag{3.171}$$

$$\mathrm{tr}[\mathbf{A}(\mathbf{B} + \mathbf{C})] = \mathrm{tr}[\mathbf{AB} + \mathbf{AC}]. \tag{3.172}$$

We also use the definition of the Cholesky decomposition in Eq. (3.164).

Since Eq. (3.168) is represented as a quadratic form with respect to $\mathbf{W}_j$, we can obtain the optimal $\mathbf{W}_j^{\mathrm{ML}}$ in the sense of ML, similar to the discussion in Section 3.4.3. By differentiating the Q function with respect to $\mathbf{W}_j$, we can derive the following equation:

$$\frac{\partial}{\partial \mathbf{W}_j'} Q(\boldsymbol{\Lambda}'_{\mathcal{J}_M} | \boldsymbol{\Lambda}_{\mathcal{J}_M}) = 0. \Rightarrow \mathbf{Z}_j - \mathbf{W}_j^{\mathrm{ML}} \boldsymbol{\Xi}_j = 0. \tag{3.173}$$

Here, we use the following matrix formulas for the derivation in Appendix B.3:

$$\begin{aligned} &\frac{\partial}{\partial \mathbf{X}} \mathrm{tr}[\mathbf{X}'\mathbf{A}] = \mathbf{A} \\ &\frac{\partial}{\partial \mathbf{X}} \mathrm{tr}[\mathbf{X}'\mathbf{X}\mathbf{A}] = 2\mathbf{X}\mathbf{A}. \quad (\mathbf{A} \text{ is a symmetric matrix}) \end{aligned} \tag{3.174}$$

Thus, we can obtain the following analytical solution:

$$\mathbf{W}_j^{\mathrm{ML}} = \mathbf{Z}_j \boldsymbol{\Xi}_j^{-1}. \tag{3.175}$$

Therefore, the optimized mean vector parameter is represented as:

$$\boldsymbol{\mu}_k^{\mathrm{ad}\,\mathrm{ML}} = \mathbf{C}_k \mathbf{Z}_j \boldsymbol{\Xi}_j^{-1} \boldsymbol{\xi}_k. \tag{3.176}$$

Therefore, $\boldsymbol{\mu}_k^{\mathrm{ad}}$ is analytically obtained by using the statistics ($\mathbf{Z}_j$ and $\boldsymbol{\Xi}_j$ in Eq. (3.169)) and initial HMM parameters ($\mathbf{C}_k$ and $\boldsymbol{\xi}_k$). This solution corresponds to the M-step of the EM algorithm, and the E-step is performed by the forward–backward algorithm, similar to that of HMMs, to compute these statistics. The training procedure is shown in Algorithm 5.

https://doi.org/10.1017/CBO9781107295360.004 Published online by Cambridge University Press

Algorithm 5 Maximum likelihood linear regression

Require: Θ and $\mathbf{\Lambda}_{\mathcal{J}_M} \leftarrow \mathbf{\Lambda}^{\text{init}}_{\mathcal{J}_M}$

1: **repeat**
2: Compute the occupation probability $\gamma_t(k)$.
3: Accumulate the sufficient statistics γ_k, $\boldsymbol{\gamma}_k$, $\mathbf{\Gamma}_k$, $\mathbf{Z}_j$, and $\mathbf{\Xi}_j$
4: Estimate the transformation parameters $\mathbf{\Lambda}^{\text{ML}}_{\mathcal{J}_M}$
5: Update the HMM parameters Θ
6: **until** Convergence

MLLR is one of the most popular techniques for acoustic modeling, and there are many variants of transformation types for HMMs, e.g., Sankar & Lee (1996), Chien, Lee & Wang (1997), Chen, Liau, Wang *et al*. (2000), Mak, Kwok & Ho (2005) and Delcroix *et al*. (2009). In addition to speech recognition, there are many other applications which are based on the adaptive training of HMMs (e.g., speech synthesis (Tamura, Masuko, Tokuda *et al*. 2001), speaker verification (Stolcke, Ferrer, Kajarekar *et al*. 2005), face recognition (Sanderson, Bengio & Gao 2006) and activity recognition (Maekawa & Watanabe 2011)).

3.6 *n*-gram with smoothing techniques

As we discussed in Section 3.1, a language model (LM) is known as crucial prior information for large vocabulary continuous speech recognition (LVCSR), according to the Bayes decision rule:

$$\hat{W} = d^{\text{MAP}}(\mathbf{O}) = \arg\max_{W} \underbrace{p(\mathbf{O}|W)}_{\text{acoustic model}} \times \underbrace{p(W)}_{\text{language model}} . \tag{3.177}$$

Many other applications include document classification, information retrieval, optical character recognition, machine translation, writing correction, and bio-informatics. An overview of language modeling in LVCSR systems has been given in Chen & Goodman (1999), Kita (1999), Rosenfeld (2000), Bellegarda (2004), and Saon & Chien (2012*b*).

A language model is a probability distribution $p(W)$ over a sequence of word strings $W = \{w_1, \cdots, w_i, \cdots, w_J\} \triangleq w_1^J$ that describes how likely it is that the word sequence W occurs as a sentence in some domain of interest.[5] Recall that word w is represented by a string, and it is an element of a set of distinct words $\mathcal{V}$, which is also called a vocabulary or dictionary. Here, w_i is a word at position i. The word string $w_i \in \mathcal{V}$ and the continuous speech vector $\mathbf{o}_t \in \mathbb{R}^D$ are both sequential patterns but in different data types and different time scales.

[5] Some languages do not have word boundaries explicitly marked by white-space in a text (e.g., Japanese and Chinese). Therefore, to process a text for language molding, these languages need an additional word segmentation step (Sproat, Gale, Shih & Chang 1996, Matsumoto, Kitauchi, Yamashita *et al*. 1999, Kudo 2005).

https://doi.org/10.1017/CBO9781107295360.004 Published online by Cambridge University Press

To get used to this notation, we provide several examples to represent the following word sequence:

my wife used my car.

This sentence has five words and one period at the end of the sentence. By regarding the period as one word,[6] this sentence is totally composed of a six-word sequence (i.e., $J = 6$), and can be represented by

$$w_1^6 = \{w_1 = \text{“my”}, w_2 = \text{“wife”}, w_3 = \text{“used”}, w_4 = \text{“my”}, w_5 = \text{“car”}, w_6 = \text{“.”}\}. \tag{3.178}$$

Note that the vocabulary for this sentence is composed of distinct unique words represented as:

$$\mathcal{V} = \{\text{“my”}, \text{“wife”}, \text{“used”}, \text{“car”}, \text{“.”}\} \tag{3.179}$$

and the vocabulary size in this example is

$$|\mathcal{V}| = 5. \tag{3.180}$$

We can also introduce the following summation over vocabulary $\mathcal{V}$, which is important in this section:

$$\sum_{w_i \in \mathcal{V}} f(w_i). \tag{3.181}$$

This summation is performed over each vocabulary in Eq. (3.179), and not over a position i in Eq. (3.178).

Basically, the prior word probability is employed to characterize the regularities in natural language. The probability of a word sequence $\{w_1, \cdots, w_J\}$ is represented based on the product rule as:

$$\begin{aligned} p_\Theta(w_1, \cdots, w_J) &= p(w_J | w_1, \cdots, w_{J-1}) p(w_1, \cdots, w_{J-1}) \\ &= p(w_J | w_1, \cdots, w_{J-1}) p(w_{J-1} | w_1, \cdots, w_{J-2}) p(w_1, \cdots, w_{J-2}) \\ &\vdots \\ &= \prod_{i=1}^{J} p(w_i | w_1^{i-1}), \end{aligned} \tag{3.182}$$

where Θ denotes the n-gram parameters, namely the n-gram probabilities, which is explained later. Here, to describe the word sequence from ith word to nth word, we use the following notation:

$$\{w_i, \cdots, w_n\} \triangleq w_i^n. \tag{3.183}$$

We also define the following special cases:

$$\begin{aligned} w_i^i &= w_i \\ w_i^n &= \emptyset, \quad \text{when } i > n \end{aligned} \tag{3.184}$$

[6] In the implementation, we additionally define the start of a sentence with an auxiliary symbol for practical use, which makes the number of words seven in this example.

https://doi.org/10.1017/CBO9781107295360.004 Published online by Cambridge University Press

where $\emptyset$ denotes an empty set. For example, the word sequences $\{w_1, \cdots, w_{i-1}\}$ are represented as

$$w_1^{i-1} \triangleq \{w_1, \cdots, w_{i-1}\} \tag{3.185}$$

and so on. When $i = 1$ in Eq. (3.182), the conditional distribution is written as:

$$p(w_1|w_1^0) \triangleq p(w_1), \text{ where } w_1^0 = \emptyset. \tag{3.186}$$

However, it makes the model larger, as the number of words in a sequence is larger, and we need to model it with the fixed model size. Thus, Eq. (3.182) is approximated with the $(n-1)$th order Markov assumption by multiplying the probabilities of a predicted word w_i conditional on its preceding $n-1$ words $\{w_{i-n+1}, \cdots, w_{i-1}\}$:

$$p_\Theta(w_1^J) \approx \prod_{i=1}^{J} p(w_i|w_{i-n+1}^{i-1}). \tag{3.187}$$

This model is called an n-gram model. Usually n is taken to be from 2 to 5, which depends on the size of the training data and applications. When $n = 1$ in Eq. (3.187), the conditional distribution is written as:

$$p(w_i|w_i^{i-1}) \triangleq p(w_i), \quad w_i^{i-1} \triangleq \emptyset, \tag{3.188}$$

which is called a *unigram* model that does not depend on any history of words.

The n-gram parameter is defined as the weight given to a conditional word sequence. The probabilistic distribution of $p(w_i|w_{i-n+1}^{i-1})$ is parameterized as with a multinomial distribution as:

$$p(w_i|w_{i-n+1}^{i-1}) \triangleq \theta_{w_i|w_{i-n+1}^{i-1}}, \tag{3.189}$$

where

$$\begin{aligned} &\sum_{w_i \in \mathcal{V}} \theta_{w_i|w_{i-n+1}^{i-1}} = 1 \\ &\theta_{w_i|w_{i-n+1}^{i-1}} \geq 0 \quad \forall w_i \in \mathcal{V}. \end{aligned} \tag{3.190}$$

Note that the number of distinct n-gram parameters for $\theta_{w_i|w_{i-n+1}^{i-1}}$ would be the index to the power of the vocabulary size $|\mathcal{V}|$, i.e., $|\mathcal{V}|^n$. In this section, we use the following notation to present a set of n-gram parameters:

$$\begin{aligned} \Theta_n &\triangleq \{\theta_{w_i|w_{i-n+1}^{i-1}} | \forall w_i \in \mathcal{V}, \cdots, w_{i-n+1} \in \mathcal{V}\} \\ &\triangleq \{p(w_i|w_{i-n+1}^{i-1})\}. \end{aligned} \tag{3.191}$$

The number of parameters is a very large since the vocabulary size of LVCSR would be more than 50 000, and the main problem of language modeling is how to compactly represent these parameters.

The straightforward way to estimate the multinomial distribution of an n-gram $\theta_{w_i|w_{i-n+1}^{i-1}} = p(w_i|w_{i-n+1}^{i-1})$ from a text corpus $\mathcal{D}$ is to compute the ML estimate by

https://doi.org/10.1017/CBO9781107295360.004 Published online by Cambridge University Press

$$\begin{aligned}\theta^{\mathrm{ML}}_{w_i|w_{i-n+1}^{i-1}} &= p^{\mathrm{ML}}(w_i|w_{i-n+1}^{i-1}) \\ &= \arg\max_{\theta_{w_i|w_{i-n+1}^{i-1}}} p(\mathcal{D}|\theta_{w_i|w_{i-n+1}^{i-1}}),\end{aligned} \tag{3.192}$$

where the multinomial likelihood function is obtained from the definition in Appendix C.2 by

$$p(\mathcal{D}|\{\theta_{w_i|w_{i-n+1}^{i-1}}|w_i \in \mathcal{V}\}) = \prod_{w_i \in \mathcal{V}} (\theta_{w_i|w_{i-n+1}^{i-1}})^{c(w_{i-n+1}^{i})}, \tag{3.193}$$

where $c(w_{i-n+1}^{i-1})$ denotes the number of occurrences of word sequence w_{i-n+1}^{i-1} in training corpus $\mathcal{D}$. To estimate the parameter, similarly to the state transitions and mixture weights in the HMM in Section 3.4.3, we introduce a Lagrange multiplier η and solve the constrained optimization problem by maximizing

$$\sum_{w_i \in \mathcal{V}} c(w_{i-n+1}^{i}) \log \theta_{w_i|w_{i-n+1}^{i-1}} + \eta \left(\sum_{w_i \in \mathcal{V}} \theta_{w_i|w_{i-n+1}^{i-1}} - 1 \right). \tag{3.194}$$

Setting the derivative of Eq. (3.194) with respect to $\theta_{w_i|w_{i-n+1}^{i-1}}$ to zero, we obtain

$$\theta_{w_i|w_{i-n+1}^{i-1}} = -\frac{1}{\eta} c(w_{i-n+1}^{i}). \tag{3.195}$$

By substituting this result into constraint Eq. (3.190), we find the value of the Lagrange multiplier

$$\begin{aligned}&\sum_{w_i \in \mathcal{V}} \theta_{w_i|w_{i-n+1}^{i-1}} = -\frac{1}{\eta} \sum_{w_i \in \mathcal{V}} c(w_{i-n+1}^{i}) = 1 \\ &\Rightarrow \eta = -\sum_{w_i \in \mathcal{V}} c(w_{i-n+1}^{i}),\end{aligned} \tag{3.196}$$

and the ML solution in the form of

$$\begin{aligned}\theta^{\mathrm{ML}}_{w_i|w_{i-n+1}^{i-1}} &= \frac{c(w_{i-n+1}^{i})}{\sum_{w_i} c(w_{i-n+1}^{i})} \\ &= \frac{c(w_{i-n+1}^{i})}{c(w_{i-n+1}^{i-1})}.\end{aligned} \tag{3.197}$$

Without loss of generality, we neglect the notation Θ in the following expressions. The goal of the most popularly used language models, trigram models, is to determine the probability of a word given the previous two words $p(w_i|w_{i-2}, w_{i-1})$, which is estimated as the number of times the word sequence $\{w_{i-2}, w_{i-1}, w_i\}$ occurs in some corpus of training data divided by the number of times the word sequence $\{w_{i-2}, w_{i-1}\}$ occurs.

However, again the number of n-gram parameters depends on the number of word combinations in a word sequence $\{w_{i-n+1}, \cdots, w_{i-1}, w_i\}$, which is counted as the number of different words $w_n \in \mathcal{V}$ at different temporal positions from $i - n + 1$ to i. This number is exponentially increased by involving large n. Although n-gram is effective

https://doi.org/10.1017/CBO9781107295360.004 Published online by Cambridge University Press

at exploiting local lexical regularities, it suffers from the inadequacies of *training data* and *long-distance information* due to too many word combinations and too narrow an n-gram window size, respectively. These limitations substantially weaken the regularization of the trained n-gram models and the prediction for unseen words. The limitation of n-gram window size could be resolved by exploiting large-span latent semantic information (Hofmann 1999*b*, Bellegarda 2000), which is addressed in Section 3.7. In what follows, we address different *smoothing* solutions to the problem of insufficient training data in the n-gram model.

3.6.1 Class-based model smoothing

A simple and meaningful approach to tackle the data sparseness problem is to consider the transition probabilities between classes rather than words, namely to adopt the class-based n-gram language model (Brown, Desouza, Mercer *et al*. 1992, Chen 2009):

$$p(w_i|w_{i-n+1}^{i-1}) \approx p(w_i|c_i)p(c_i|c_{i-n+1}^{i-1}), \tag{3.198}$$

where $c_i \in \mathcal{C}$ is the class assignment of word w_i, $p(w_i|c_i)$ is the probability of word w_i, generated from class c_i, and $p(c_i|c_{i-n+1}^{i-1})$ is the class n-gram. The class assignments of different words are determined beforehand according to the word clustering using the metric of *mutual information*. An existing linguistic class (e.g., part of speech) is also used to provide the class assignments.

The word probability given a class $p(w_i|c_i)$ is usually estimated by using the ML estimation, similarly to Eq. (3.197), as:

$$p^{\mathsf{ML}}(w_i|c_i) = \theta_{w_i|c_i}^{\mathsf{ML}} = \frac{c(w_i, c_i)}{\sum_{w_i \in \mathcal{V}} c(w_i, c_i)}, \tag{3.199}$$

where $c(w_i, c_i)$ is the number of word counts labeled by both $\{w_i, c_i\}$ in the corpus $\mathcal{D}$. The class n-gram probability $p(c_i|c_{i-n+1}^{i-1})$ is estimated by using the ML estimation (replacing the word counts in Eq. (3.197) with class counts), or a smoothing technique, explained in the following sections.

The model parameters are composed of

$$\Theta = \{\{p(w_i|c_i)\}, \{p(c_i|c_{i-n+1}^{i-1})\}\}. \tag{3.200}$$

Both are represented by the multinomial distributions. Since the number of distinct classes $|\mathcal{C}|$ is much smaller than the vocabulary size $|\mathcal{V}|$, the model size is significantly reduced. Model parameters could be reliably estimated. For example, the number of parameters of $\{\{p(w_i|c_i)\}$ is $|\mathcal{V}||\mathcal{C}|$, and the number of parameters of $\{p(c_i|c_{i-n+1}^{i-1})\}$ is $|\mathcal{C}|^n$. Since $|\mathcal{V}| \gg |\mathcal{C}|$, the total number of parameters of a class n-gram model $|\mathcal{V}||\mathcal{C}| + |\mathcal{C}|^n$ is much smaller than an n-gram model $|\mathcal{V}|^n$. The class-based n-gram is also seen as a smoothed language model.

3.6.2 Jelinek–Mercer smoothing

As reported in Chen & Goodman (1999), it is usual to deal with the issue of data sparseness in an n-gram model by using a linear interpolation method where the nth order

https://doi.org/10.1017/CBO9781107295360.004 Published online by Cambridge University Press

language model $p^{\text{interp}}(w_i|w_{i-n+1}^{i-1})$ is estimated by interpolating with the $(n-1)$th order language model in a form of

$$p^{\text{interp}}(w_i|w_{i-n+1}^{i-1}) = \lambda_{w_{i-n+1}^{i-1}} p^{\text{ML}}(w_i|w_{i-n+1}^{i-1}) + (1-\lambda_{w_{i-n+1}^{i-1}}) p^{\text{interp}}(w_i|w_{i-n+2}^{i-1}), \tag{3.201}$$

where $\lambda_{w_{i-n+1}^{i-1}}$ denotes the interpolation weight which is estimated for each w_{i-n+1}^{i-1} in accordance with the ML method. The reason the interpolation weight $\lambda_{w_{i-n+1}^{i-1}}$ does not depend on w_i comes from the constraint of the sum-to-one property of an n-gram model. The nth order smoothed model is calculated recursively as a linear interpolation between the nth order ML model and the $(n-1)$th order smoothed model, that is

$$\begin{aligned} p^{\text{interp}}(w_i|w_{i-n+1}^{i-1}) &= \lambda_{w_{i-n+1}^{i-1}} p^{\text{ML}}(w_i|w_{i-n+1}^{i-1}) + (1-\lambda_{w_{i-n+1}^{i-1}}) \\ &\quad \times \left(\lambda_{w_{i-n+2}^{i-1}} p^{\text{ML}}(w_i|w_{i-n+2}^{i-1}) + (1-\lambda_{w_{i-n+2}^{i-1}}) p^{\text{interp}}(w_i|w_{i-n+3}^{i-1})\right) \\ &= \cdots . \end{aligned} \tag{3.202}$$

To carry through this recursive process to the end, we can take the smoothed unigram or first-order model to be the ML distribution in Eq. (3.192), i.e., when $n = 1$,

$$\begin{aligned} p^{\text{interp}}(w_i|w_i^{i-1}) &= p^{\text{interp}}(w_i) \\ &= p^{\text{ML}}(w_i) = \theta_{w_i}^{\text{ML}}, \end{aligned} \tag{3.203}$$

where $\theta_{w_i}^{\text{ML}}$ is obtained from Eq. (3.197) as:

$$\theta_{w_i}^{\text{ML}} = \frac{c(w_i)}{\sum_{w_i} c(w_i)}. \tag{3.204}$$

Or we can take the smoothed zeroth-order model to be the discrete uniform distribution (Appendix C.1),

$$p^{\text{interp}}(w_i) = \lambda p^{\text{ML}}(w_i) + (1-\lambda)\text{Unif}(w_i), \tag{3.205}$$

where the uniform distribution is defined with the vocabulary size $|\mathcal{V}|$ as:

$$\text{Unif}(w_i) \triangleq \frac{1}{|\mathcal{V}|}. \tag{3.206}$$

Now, let us consider the original interpolation equation Eq. (3.201). The parameter $\lambda_{w_{i-n+1}^{i-1}}$ is estimated individually for the w_{i-n+1}^{i-1} that maximizes the probability of some of the data. Practically, the selection could be done for buckets of parameters. In general, this class of interpolated models is also known as the n-gram model with Jelinek–Mercer smoothing (Jelinek & Mercer 1980), which is a standard form of *interpolation smoothing*. The smoothing techniques in the following sections (Witten–Bell (WB) smoothing in Section 3.6.3, absolute discount in Section 3.6.4, Kneser–Ney (KN) smoothing in Eq. (3.232), and PLSA smoothing in Eq. (3.319)) provide specific interpolation weights in this standard form, as shown in Table 3.1.

https://doi.org/10.1017/CBO9781107295360.004 Published online by Cambridge University Press

An important constraint of the n-gram model is that the summation over w_i goes to 1, as shown in Eq. (3.190). The Jelinek–Mercer smoothing (and the following smoothing techniques) satisfies this condition, i.e.,

$$\sum_{w_i \in \mathcal{V}} p^{\text{interp}}(w_i | w_{i-n+1}^{i-1}) = 1. \tag{3.207}$$

We can prove this condition recursively from the unigram case to the n-gram case. First, it is obvious that for the unigram models for the Eqs. (3.203) and (3.205) cases:

$$\sum_{w_i \in \mathcal{V}} p^{\text{interp}}(w_i) = 1. \tag{3.208}$$

Next, the bi-gram case is also proved by:

$$\begin{aligned}
\sum_{w_i \in \mathcal{V}} p^{\text{interp}}(w_i | w_{i-1}) &= \sum_{w_i \in \mathcal{V}} \left(\lambda_{w_{i-1}} p^{\text{ML}}(w_i | w_{i-1}) + (1 - \lambda_{w_{i-1}}) p^{\text{interp}}(w_i) \right) \\
&= \lambda_{w_{i-1}} \left(\sum_{w_i \in \mathcal{V}} p^{\text{ML}}(w_i | w_{i-1}) \right) + (1 - \lambda_{w_{i-1}}) \left(\sum_{w_i \in \mathcal{V}} p^{\text{interp}}(w_i) \right) \\
&= \lambda_{w_{i-1}} + (1 - \lambda_{w_{i-1}}) = 1,
\end{aligned} \tag{3.209}$$

where we use Eq. (3.208) and $\sum_{w_i} p^{\text{ML}}(w_i | \cdot) = 1$. That is proven in the n-gram case, trivially. The important property of this proof of the sum-to-one condition is that the summation over w_i does not depend on the interpolation weight $\lambda_{w_{i-n+1}^{i-1}}$.

3.6.3 Witten–Bell smoothing

Witten–Bell smoothing (Witten & Bell 1991) is considered to be an instance of *interpolation smoothing* as addressed in Section 3.6.2 by setting a specific value for the interpolation parameter $\lambda_{w_{i-n+1}^{i-1}}$. The Witten–Bell smoothing first defines the following number based on the number of unique words that follow the history w_{i-n+1}^{i-1}:

$$N_{1+}(w_{i-n+1}^{i-1}, \bullet) \triangleq |\{w_i | c(w_{i-n+1}^{i-1}, w_i) > 0\}|. \tag{3.210}$$

The notation N_{1+} represents the number of distinct words that have one or more counts, and the $\bullet$ represents any possible words at i with this condition. By using $N_{1+}(w_{i-n+1}^{i-1}, \bullet)$, the Witten–Bell smoothing assigns the factor $1 - \lambda_{w_{i-n+1}^{i-1}}$ for the lower-order model (the second term in Eq. (3.201)) where

$$\begin{aligned}
1 - \lambda_{w_{i-n+1}^{i-1}} &\triangleq \frac{N_{1+}(w_{i-n+1}^{i-1}, \bullet)}{\sum_{w_i} c(w_{i-n+1}^{i}) + N_{1+}(w_{i-n+1}^{i-1}, \bullet)} \\
&= \frac{N_{1+}(w_{i-n+1}^{i-1}, \bullet)}{c(w_{i-n+1}^{i-1}) + N_{1+}(w_{i-n+1}^{i-1}, \bullet)}.
\end{aligned} \tag{3.211}$$

https://doi.org/10.1017/CBO9781107295360.004 Published online by Cambridge University Press

This factor is interpreted as the frequency with which we should use the lower-order model to predict the next word. It is meaningful that more unique words appearing after history words w_{i-n+1}^{i-1}, i.e., larger $N_{1+}(w_{i-n+1}^{i-1}, \bullet)$, implies that more reliable values of the $(n-1)$-grams are estimated, and the higher weight factor $1 - \lambda_{w_{i-n+1}^{i-1}}$ should be assigned for $(n-1)$-grams. Similarly, the rest of the weight $\lambda_{w_{i-n+1}^{i-1}}$ (the first term in Eq. (3.201)) is represented from Eq. (3.211) as

$$\begin{aligned}\lambda_{w_{i-n+1}^{i-1}} &= 1 - \frac{N_{1+}(w_{i-n+1}^{i-1}, \bullet)}{\sum_{w_i} c(w_{i-n+1}^{i}) + N_{1+}(w_{i-n+1}^{i-1}, \bullet)} \\ &= \frac{\sum_{w_i} c(w_{i-n+1}^{i})}{\sum_{w_i} c(w_{i-n+1}^{i}) + N_{1+}(w_{i-n+1}^{i-1}, \bullet)}. \end{aligned} \tag{3.212}$$

This is the modified ML estimate with the unique word count $N_{1+}(w_{i-n+1}^{i-1}, \bullet)$.

According to the interpolation smoothing in Eq. (3.201) and Eqs. (3.211) and (3.212), the Witten–Bell smoothing is expressed by

$$\begin{aligned}&p^{\mathsf{WB}}(w_i|w_{i-n+1}^{i-1}) \\ &\quad = \frac{c(w_{i-n+1}^{i})p^{\mathsf{ML}}(w_i|w_{i-n+1}^{i-1}) + N_{1+}(w_{i-n+1}^{i-1}, \bullet)p^{\mathsf{WB}}(w_i|w_{i-n+2}^{i-1})}{\sum_{w_i} c(w_{i-n+1}^{i}) + N_{1+}(w_{i-n+1}^{i-1}, \bullet)}. \end{aligned} \tag{3.213}$$

Note that the factor used in the Witten–Bell smoothing in Eq. (3.211) only depends on w_{i-n+1}^{i-1}, and has the same dependency as $\lambda_{w_{i-n+1}^{i-1}}$. Therefore, it is trivial that the Witten–Bell smoothing satisfies the sum-to-one condition, as it is a special solution of the interpolation smoothing (Eq. (3.201)) that satisfies the sum-to-one condition.

3.6.4 Absolute discounting

Absolute discounting is also considered to be an instance of *interpolation smoothing* as addressed in Section 3.6.2. However, the equation form is not represented as the Jelinek–Mercer form, which we set down again as follows for comparison:

$$p^{\mathsf{interp}}(w_i|w_{i-n+1}^{i-1}) = \lambda_{w_{i-n+1}^{i-1}} p^{\mathsf{ML}}(w_i|w_{i-n+1}^{i-1}) + (1 - \lambda_{w_{i-n+1}^{i-1}}) p^{\mathsf{interp}}(w_i|w_{i-n+2}^{i-1}). \tag{3.214}$$

Recall that Eq. (3.214) has the weight λ depending on the previous word sequence w_{i-n+1}^{i-1}, and is composed of the ML probability $p^{\mathsf{ML}}(w_i|w_{i-n+1}^{i-1})$ and lower-order probability $p^{\mathsf{interp}}(w_i|w_{i-n+2}^{i-1})$. However, in absolute discounting, instead of multiplying the higher-order ML model by a factor $\lambda_{w_{i-n+1}^{i-1}}$, the higher-order distribution is created by subtracting a fixed discount d for the case of non-zero count. The absolute discounting is defined by

$$p^{\mathsf{ABS}}(w_i|w_{i-n+1}^{i-1}) \triangleq \frac{\max\{c(w_{i-n+1}^{i}) - d, 0\}}{\sum_{w_i} c(w_{i-n+1}^{i})} + (1 - \lambda_{w_{i-n+1}^{i-1}}) p^{\mathsf{ABS}}(w_i|w_{i-n+2}^{i-1}), \tag{3.215}$$

where $0 \leq d \leq 1$ denotes a discounting parameter.

https://doi.org/10.1017/CBO9781107295360.004 Published online by Cambridge University Press

The interpolation weight $(1 - \lambda_{w_{i-n+1}^{i-1}})$ is formed with the unique word count $N_{1+}(w_{i-n+1}^{i-1}, \bullet)$ defined in Eq. (3.210) as:

$$1 - \lambda_{w_{i-n+1}^{i-1}} = \frac{dN_{1+}(w_{i-n+1}^{i-1}, \bullet)}{\sum_{w_i} c(w_{i-n+1}^{i})}. \tag{3.216}$$

Note that the weight does not depend on w_i, but w_{i-n+1}^{i-1}, similarly to the Jelinek–Mercer form in Eq. (3.214). The way to find weight parameter $1 - \lambda_{w_{i-n+1}^{i-1}}$ is again based on the sum-to-one condition. Consider the condition

$$\sum_{w_i} p^{\mathsf{ABS}}(w_i|\cdot) = 1. \tag{3.217}$$

Then, by taking the summation over w_i for both sides in Eq. (3.215), Eq. (3.215) can be rewritten as:

$$\begin{aligned} 1 &= \sum_{w_i} p^{\mathsf{ABS}}(w_i|w_{i-n+1}^{i-1}) \\ &= \sum_{w_i} \frac{\max\{c(w_{i-n+1}^{i}) - d, 0\}}{\sum_{w_i} c(w_{i-n+1}^{i})} + (1 - \lambda_{w_{i-n+1}^{i-1}}) \sum_{w_i} p^{\mathsf{ABS}}(w_i|w_{i-n+2}^{i-1}) \\ &= \frac{\sum_{w_i} \max\{c(w_{i-n+1}^{i}) - d, 0\}}{\sum_{w_i} c(w_{i-n+1}^{i})} + (1 - \lambda_{w_{i-n+1}^{i-1}}). \end{aligned} \tag{3.218}$$

Therefore,

$$\begin{aligned} 1 - \lambda_{w_{i-n+1}^{i-1}} &= 1 - \frac{\sum_{w_i} \max\{c(w_{i-n+1}^{i}) - d, 0\}}{\sum_{w_i} c(w_{i-n+1}^{i})} \\ &= \frac{\sum_{w_i} c(w_{i-n+1}^{i}) - \sum_{w_i} \max\{c(w_{i-n+1}^{i}) - d, 0\}}{\sum_{w_i} c(w_{i-n+1}^{i})} \\ &= \frac{\sum_{w_i} \left(c(w_{i-n+1}^{i}) - \max\{c(w_{i-n+1}^{i}) - d, 0\}\right)}{\sum_{w_i} c(w_{i-n+1}^{i})}. \end{aligned} \tag{3.219}$$

Now we focus on the numerator in Eq. (3.219) that represents the total discount value from d. By considering the cases when $c(w_{i-n+1}^{i}) > 0$ and $c(w_{i-n+1}^{i}) = 0$, we can derive the following equation:

$$c(w_{i-n+1}^{i}) - \max\{c(w_{i-n+1}^{i}) - d, 0\} = \begin{cases} d & \text{if } c(w_{i-n+1}^{i}) > 0 \\ 0 & \text{if } c(w_{i-n+1}^{i}) = 0. \end{cases} \tag{3.220}$$

Therefore, by substituting Eq. (3.220) into Eq. (3.219), and by using the unique word count $N_{1+}(w_{i-n+1}^{i-1}, \bullet)$, Eq. (3.219) is finally represented as:

$$\begin{aligned} 1 - \lambda_{w_{i-n+1}^{i-1}} &= \frac{\sum_{w_i | c(w_{i-n+1}^{i}) > 0} d}{\sum_{w_i} c(w_{i-n+1}^{i})} \\ &= \frac{dN_{1+}(w_{i-n+1}^{i-1}, \bullet)}{\sum_{w_i} c(w_{i-n+1}^{i})}, \end{aligned} \tag{3.221}$$

https://doi.org/10.1017/CBO9781107295360.004 Published online by Cambridge University Press

where $\sum_{w_i|c(w_{i-n+1}^{i})>0}$ means that the summation is undertaken for a subset of distinct words in the vocabulary $\mathcal{V}$ that satisfies the condition $c(w_{i-n+1}^{i}) > 0$. Since d does not depend on w_i, the numerator can be represented with d times the number of distinct words w_i that satisfies the condition $c(w_{i-n+1}^{i}) > 0$, which corresponds to $N_{1+}(w_{i-n+1}^{i-1}, \bullet)$. Thus, we prove Eq. (3.216).

This weight means that the discount d in an observed n-gram event $c(w_{i-n+1}^{i-1} w_i) > 0$ at current word w_i is distributed to compensate for those unseen events $c(w_{i-n+1}^{i-1} w_i) = 0$ where a lower-order model $p^{\mathsf{ABS}}(w_i|w_{i-n+2}^{i-1})$ is adopted. The discount d is shared for all n-grams and could be measured by using the total number of n-grams with exactly one and two counts, i.e., $c(w_{i-n+1}^{i-1}) = 1$ and $c(w_{i-n+1}^{i-1}) = 2$ in the training data, respectively (Ney, Essen & Kneser 1994).

In summary, the absolute discounting is represented by

$$p^{\mathsf{ABS}}(w_i|w_{i-n+1}^{i-1}) \triangleq \frac{\max\{c(w_{i-n+1}^{i}) - d, 0\}}{\sum_{w_i} c(w_{i-n+1}^{i})} + \frac{dN_{1+}(w_{i-n+1}^{i-1}, \bullet)}{\sum_{w_i} c(w_{i-n+1}^{i})} p^{\mathsf{ABS}}(w_i|w_{i-n+2}^{i-1}). \tag{3.222}$$

Note that the absolute discounting does not include the exact ML probabilities (although they can be included by re-arranging the first term). The weight parameter for the lower-order model is proportional to the value $N_{1+}(w_{i-n+1}^{i-1}, \bullet)$, which has a similarity to that in Witten–Bell smoothed n-grams, as seen in Eq. (3.211).

Thus, we have explained Witten–Bell smoothing in Section 3.6.3 and absolute discount in Section 3.6.4 as instances of the interpolation (Jelinek–Mercer) smoothing techniques. The next section introduces another type of well-known smoothing technique called *backoff smoothing*, with Katz smoothing as an example.

3.6.5 Katz smoothing

Katz smoothing (Katz 1987) was developed by intuitively combining higher-order models with lower-order models through scaling of the ML distribution. Taking a bi-gram as an example, the Katz smoothing is performed by calculating the probability by considering the cases where the co-occurrence count $c(w_{i-1}, w_i)$ of w_{i-1} and w_i is zero or positive as follows:

$$p^{\mathsf{KZ}}(w_i|w_{i-1}) \triangleq \begin{cases} d_{c(w_{i-1},w_i)} p^{\mathsf{ML}}(w_i|w_{i-1}) & \text{if } c(w_{i-1}, w_i) > 0 \\ \alpha(w_{i-1}) p^{\mathsf{KZ}}(w_i) & \text{if } c(w_{i-1}, w_i) = 0, \end{cases} \tag{3.223}$$

where $d_r < 1$ is a discount ratio that reduces the probability estimated from the ML. The discount ratio is usually is obtained by the Good–Turing estimate (Good 1953), and this book does not describe it in detail. The expression $\alpha(w_{i-1})$ is a scaling factor that only depends on the previous word w_{i-1}, while $p^{\mathsf{KZ}}(w_i)$ is a unigram probability, and we usually use the ML unigram probability for the bi-gram case, i.e.,

$$p^{\mathsf{KZ}}(w_i) = p^{\mathsf{ML}}(w_i). \tag{3.224}$$

https://doi.org/10.1017/CBO9781107295360.004 Published online by Cambridge University Press

Now we focus on the scaling factor $\alpha(w_{i-1})$, which is obtained by the sum-to-one condition of $p^{\mathsf{KZ}}(w_i|\cdot)$ and $p^{\mathsf{ML}}(w_i|\cdot)$ that must be satisfied for any probabilistic distributions. By summing over w_i in both sides of Eq. (3.223), the equation can be rewritten as

$$\sum_{w_i} p^{\mathsf{KZ}}(w_i|w_{i-1}) = \begin{cases} \sum_{w_i} d_{c(w_{i-1},w_i)} p^{\mathsf{ML}}(w_i|w_{i-1}) & \text{if } c(w_{i-1},w_i) > 0 \\ \sum_{w_i} \alpha(w_{i-1}) p^{\mathsf{ML}}(w_i) & \text{if } c(w_{i-1},w_i) = 0. \end{cases} \tag{3.225}$$

This leads to the following equation:

$$1 = \sum_{w_i|c(w_{i-1},w_i)>0} d_{c(w_{i-1},w_i)} p^{\mathsf{ML}}(w_i|w_{i-1}) + \alpha(w_{i-1}) \sum_{w_i|c(w_{i-1},w_i)=0} p^{\mathsf{ML}}(w_i), \tag{3.226}$$

where $\sum_{w_i|c(w_{i-1},w_i)>0}$ or $\sum_{w_i|c(w_{i-1},w_i)=0}$ means that the summation is undertaken for a subset of distinct words in the vocabulary $\mathcal{V}$ that satisfies the condition $c(w_{i-1},w_i) > 0$ or $c(w_{i-1},w_i) = 0$. Thus, we obtain that

$$\begin{aligned} \alpha(w_{i-1}) &= \frac{1 - \sum_{w_i|c(w_{i-1},w_i)>0} d_{c(w_{i-1},w_i)} p^{\mathsf{ML}}(w_i|w_{i-1})}{\sum_{w_i|c(w_{i-1},w_i)=0} p^{\mathsf{ML}}(w_i)} \\ &= \frac{1 - \sum_{w_i|c(w_{i-1},w_i)>0} d_{c(w_{i-1},w_i)} p^{\mathsf{ML}}(w_i|w_{i-1})}{1 - \sum_{w_i|c(w_{i-1},w_i)>0} p^{\mathsf{ML}}(w_i)}. \end{aligned} \tag{3.227}$$

This smoothing technique can be generalized to the n-gram probability as

$$p^{\mathsf{KZ}}(w_i|w_{i-n+1}^{i-1}) \triangleq \begin{cases} d_{w_{i-n+1}^{i}} p^{\mathsf{ML}}(w_i|w_{i-n+1}^{i-1}) & \text{if } c(w_{i-n+1}^{i}) > 0 \\ \alpha(w_{i-n+1}^{i-1}) p^{\mathsf{KZ}}(w_i|w_{i-n+2}^{i-1}) & \text{if } c(w_{i-n+1}^{i}) = 0, \end{cases} \tag{3.228}$$

where

$$\alpha(w_{i-n+1}^{i-1}) = \frac{1 - \sum_{w_i|c(w_{i-n+1}^{i})>0} d_{c(w_{i-n+1}^{i})} p^{\mathsf{ML}}(w_i|w_{i-n+1}^{i-1})}{1 - \sum_{w_i|c(w_{i-n+1}^{i})>0} p^{\mathsf{KZ}}(w_i|w_{i-n+2}^{i-1})}. \tag{3.229}$$

This smoothing scheme is known as a realization of *backoff smoothing*. This smoothing is obtained by the weighted multiplication of the ML probability and is different from interpolation (Jelinek–Mercer) smoothing, as discussed in Section 3.6.2, which is obtained by the weighted summation of the ML probability. Note that both smoothing techniques include some free parameters (weight λ in interpolation smoothing and the discount factor d in backoff smoothing), but the other parameters are obtained by using the sum-to-one constraint of the probability distribution. Similarly to the meaning of λ, backoff smoothing relies more on the lower-order n-gram probability when d is small (close to 0), while backoff smoothing relies more on the ML n-gram probability when d is large (close to 1). The next section describes a famous (modified) Kneser–Ney smoothing, which can be realized with both interpolation and backoff smoothing methods.

3.6.6 Kneser–Ney smoothing

Kneser–Ney (KN) smoothing (Kneser & Ney 1995) can be interpreted as an extension of the absolute discount approach, as discussed in Section 3.6.4. When the highest-order

https://doi.org/10.1017/CBO9781107295360.004 Published online by Cambridge University Press

probability case is considered, it is same as the absolute discount. However, the unique property of KN smoothing is that it provides a special lower-order probability based on the numbers of distinct words.

For example, in the highest-order probability case, the original Kneser–Ney method was developed as the following backoff smoothed model:

$$p^{\mathsf{KN}}(w_i|w_{i-n+1}^{i-1}) = \begin{cases} \frac{\max\{c(w_{i-n+1}^{i})-d,0\}}{\sum_{w_i} c(w_{i-n+1}^{i})} & \text{if } c(w_{i-n+1}^{i}) > 0 \\ \gamma(w_{i-n+1}^{i-1}) p^{\mathsf{KN}}(w_i|w_{i-n+2}^{i-1}) & \text{if } c(w_{i-n+1}^{i}) = 0, \end{cases} \tag{3.230}$$

where $\gamma(w_{i-n+1}^{i-1})$ is chosen to make the distribution sum to 1 and has the form

$$\gamma(w_{i-n+1}^{i-1}) = \frac{dN_{1+}(w_{i-n+1}^{i-1}, \bullet)}{\sum_{w_i} c(w_{i-n+1}^{i})}. \tag{3.231}$$

Alternatively, the Kneser–Ney model could be estimated according to the interpolation smoothing scheme in the highest-order probability case based on

$$p^{\mathsf{KN}}(w_i|w_{i-n+1}^{i-1}) = \frac{\max\{c(w_{i-n+1}^{i}) - d, 0\}}{\sum_{w_i} c(w_{i-n+1}^{i})} + \frac{dN_{1+}(w_{i-n+1}^{i-1}, \bullet)}{\sum_{w_i} c(w_{i-n+1}^{i})} p^{\mathsf{KN}}(w_i|w_{i-n+2}^{i-1}). \tag{3.232}$$

Note that the first term of Eq. (3.232) is calculated from a modification of the ML probability, and Eq. (3.232) is exactly the same as the absolute discounting in Eq. (3.222), except for the lower-order probability $p^{\mathsf{KN}}(w_i|w_{i-n+2}^{i-1})$. However, in the lower-order probability, instead of using Eq. (3.232) recursively by changing n to $n-1$, the Kneser–Ney smoothing provides more intuitive probability than the ML probability for the first term by considering the continuation of words.

For a simple explanation, we consider the bi-gram case (the highest-order probability is a bi-gram probability), i.e., Eq. (3.232) is represented as:

$$p^{\mathsf{KN}}(w_i|w_{i-1}) = \frac{\max\{c(w_{i-1}, w_i) - d, 0\}}{\sum_{w_i} c(w_{i-1}, w_i)} + \frac{dN_{1+}(w_{i-1}, \bullet)}{\sum_{w_i} c(w_{i-1}, w_i)} p^{\mathsf{KN}}(w_i). \tag{3.233}$$

The question here is whether we really use the unigram probability for $p^{\mathsf{KN}}(w_i)$. This is often illustrated with an example of the bi-gram "San Francisco" (Chen & Goodman 1999, Jurafsky 2014). In the training data of the *Wall Street Journal* (WSJ0) corpus (Paul & Baker 1992), the bi-gram "San Francisco" appears 3222 times, while the word "Francisco" appeared 3329 times. The other word "glasses" appears 185 times. That is

$$\begin{aligned} c(w_{i-1} = \text{“San”}, w_i = \text{“Francisco”}) &= 3222 \\ c(w_i = \text{“Francisco”}) &= 3329 \\ c(w_i = \text{“glasses”}) &= 185. \end{aligned} \tag{3.234}$$

From these statistics, we can say that:

- The word "Francisco" almost always (96.8%) follows "San," and does not follow the other words in most cases.
- However, the word "Francisco" is more common (18 times more) than "glasses."

https://doi.org/10.1017/CBO9781107295360.004 Published online by Cambridge University Press

Then, let us consider the following sentences:

1. **I can't see without my reading "glasses."**
2. **I can't see without my reading "Francisco."**

We consider the case when there are no "reading glasses" and "reading Francisco" in a training corpus, and the lower-order probability $p^{\mathsf{KN}}(w_i)$ (the second term in (3.233)) is used to compute these sentence probabilities. Intuitively, we want to make the probability of the first sentence with "glasses" higher than the other. However, if we use the count-oriented (ML-like) probability for $p^{\mathsf{KN}}(w_i)$, the sentence with "Francisco" is assigned higher probability because the simple word count of "Francisco" is much larger than that of "glasses." This problem can be avoided by considering the *continuity* of word sequences rather than word counts in the lower-order probability.[7]

The KN smoothing provides the following unigram probability that does not use the word count (ML-like) information:

$$p^{\mathsf{KN}}(w_i) \triangleq \frac{N_{1+}(\bullet, w_i)}{N_{1+}(\bullet, \bullet)}. \tag{3.235}$$

Here, from the definition of the unique count N_{1+} in Eq. (3.210), $N_{1+}(\bullet, w_i)$ and $N_{1+}(\bullet, \bullet)$ are represented as follows:

$$N_{1+}(\bullet, w_i) = |\{w_{i-1} | c(w_{i-1}, w_i) > 0\}|, \tag{3.236}$$

$$N_{1+}(\bullet, \bullet) = |\{\{w_i, w_{i-1}\} | c(w_{i-1}, w_i) > 0\}| = \sum_{w_i} N_{1+}(\bullet, w_i). \tag{3.237}$$

In the previous example,

$$\begin{aligned} N_{1+}(\bullet, w_i = \text{"Francisco"}) &= 58 \\ N_{1+}(\bullet, w_i = \text{"glasses"}) &= 88. \end{aligned} \tag{3.238}$$

The unique count of $N_{1+}(\bullet, w_i = \text{"glasses"})$ is larger than that of $N_{1+}(\bullet, w_i = \text{"Francisco"})$. Thus, the probability of the sentence with "glasses" becomes larger when we use the number of unique words in (3.235), which is a more intuitive result in this example.

The probability based on the unique counts can be generalized from the unigram case in Eq. (3.235) to higher-order n-gram probabilities, except for the highest-order n-gram that uses the absolute discounting from Eq. (3.232). For example, $p^{\mathsf{KN}}(w_i | w_{i-n+2}^{i-1})$, which is the second term in Eq. (3.232), is represented as follows:

$$\begin{aligned} p^{\mathsf{KN}}(w_i | w_{i-n+2}^{i-1}) = &\frac{\max\{N_{1+}(\bullet, w_{i-n+2}^{i-1}) - d, 0\}}{N_{1+}(\bullet, w_{i-n+2}^{i-1}, \bullet)} \\ &+ \frac{d N_{1+}(w_{i-n+2}^{i-1}, \bullet)}{N_{1+}(\bullet, w_{i-n+2}^{i-1}, \bullet)} p^{\mathsf{KN}}(w_i | w_{i-n+3}^{i-1}), \end{aligned} \tag{3.239}$$

[7] Actually "reading glasses" has appeared twice in the WSJ0 corpus, and the lower-order probability cannot be used so much in the WSJ0 language model when we set d very small. This is another solution to resolve this discontinuity problem by correcting a very large size of corpus.

https://doi.org/10.1017/CBO9781107295360.004 Published online by Cambridge University Press

where

$$N_{1+}(\bullet, w_{i-n+2}^{i-1}) \triangleq |\{w_{i-n+1} | c(w_{i-n+1}^{i-1}) > 0\}|$$
$$N_{1+}(\bullet, w_{i-n+2}^{i-1}, \bullet) \triangleq |\{w_{i-n+1}, w_i | c(w_{i-n+1}^{i}) > 0\}| = \sum_{w_i} N_{1+}(\bullet, w_{i-n+2}^{i-1}). \quad (3.240)$$

Thus, we can obtain the interpolation version of the Kneser–Ney smoothing for all cases, which is summarized as follows:

- The highest-order n-gram probability

$$p^{\mathsf{KN}}(w_i | w_{i-n+1}^{i-1}) = \frac{\max\{c(w_{i-n+1}^{i}) - d, 0\}}{\sum_{w_i} c(w_{i-n+1}^{i})} + \frac{dN_{1+}(w_{i-n+1}^{i-1}, \bullet)}{\sum_{w_i} c(w_{i-n+1}^{i})} p^{\mathsf{KN}}(w_i | w_{i-n+2}^{i-1}); \quad (3.241)$$

- Lower-order n-gram probability

$$p^{\mathsf{KN}}(w_i | w_{i-n+2}^{i-1}) = \frac{\max\{N_{1+}(\bullet, w_{i-n+2}^{i-1}) - d, 0\}}{N_{1+}(\bullet, w_{i-n+2}^{i-1}, \bullet)} + \frac{dN_{1+}(w_{i-n+2}^{i-1}, \bullet)}{N_{1+}(\bullet, w_{i-n+2}^{i-1}, \bullet)} p^{\mathsf{KN}}(w_i | w_{i-n+3}^{i-1}); \quad (3.242)$$

- Unigram probability

$$p^{\mathsf{KN}}(w_i) = \frac{N_{1+}(\bullet, w_i)}{N_{1+}(\bullet, \bullet)}. \quad (3.243)$$

In general, the interpolated n-gram model yields better performance than the backoff n-gram model (Chen & Goodman 1999).

Modified Kneser–Ney smoothing

In Kneser–Ney smoothing and discounting smoothing, a discount parameter d in Eq. (3.241) plays an important role in striking a balance between the target-order n-gram probability and the lower-order one. Then, it is a simple question whether the single discount parameter d is precise enough to handle the balance. Basically, the distribution of distinct words follows the *power-law* property where only a few frequent words form a high proportion of the total, but a lot of rare words occur only *one* or *two* times, i.e.,

$$|\{w_i | c(w_i) = 1\}| \gg |\{w_i | c(w_i) = 2\}| \gg |\{w_i | c(w_i) > 2\}|. \quad (3.244)$$

This power-law property comes from the "rich-get-richer behavior" in natural language, and this property also applies to n-grams, as well as words. Thus, Kneser–Ney smoothing is modified by separately using three discount parameters d_1, d_2, and d_{3+} for those

https://doi.org/10.1017/CBO9781107295360.004 Published online by Cambridge University Press

n-grams with one, two, and three or more counts, respectively, instead of using a single discount d for all non-zero counts, as follows:

$$d(c) \triangleq \begin{cases} 0 & \text{if } c = 0 \\ d_1 & \text{if } c = 1 \\ d_2 & \text{if } c = 2 \\ d_{3+} & \text{if } c \geq 3. \end{cases} \tag{3.245}$$

These parameters are empirically determined as (Chen & Goodman 1999):

$$d_{\text{base}} = \frac{m_1}{m_1 + 2m_2}, \tag{3.246}$$

$$d_1 = 1 - 2d_{\text{base}}\frac{m_2}{m_1}, \tag{3.247}$$

$$d_2 = 2 - 3d_{\text{base}}\frac{m_3}{m_2}, \tag{3.248}$$

$$d_{3+} = 3 - 4d_{\text{base}}\frac{m_4}{m_3}, \tag{3.249}$$

where m_j is the total number of *n*-grams appearing j times in a training corpus, i.e.,

$$m_j \triangleq \sum_{w_{i-n}^i : c(w_{i-n}^i) = j} c(w_{i-n}^i). \tag{3.250}$$

Based on the new discounting parameter $d(c)$ in Eq. (3.245), a modified Kneser–Ney (MKN) smoothing is conducted to estimate the smoothed *n*-grams using

$$p^{\text{MKN}}(w_i|w_{i-n+1}^{i-1}) = \frac{c(w_{i-n+1}^i) - d(c(w_{i-n+1}^{i-1}))}{\sum_{w_i} c(w_{i-n+1}^i)} + \gamma(w_{i-n+1}^{i-1}) p^{\text{MKN}}(w_i|w_{i-n+2}^{i-1}), \tag{3.251}$$

where $\gamma(w_{i-n+1}^{i-1})$ is derived by considering the sum-to-one condition. By taking the summation over w_i in both sides of Eq. (3.251), Eq. (3.251) is represented as follows:

$$\begin{aligned} \sum_{w_i} p^{\text{MKN}}(w_i|w_{i-n+1}^{i-1}) &= 1 \\ &= \sum_{w_i} \frac{c(w_{i-n+1}^i) - d(c(w_{i-n+1}^{i-1}))}{\sum_{w_i} c(w_{i-n+1}^i)} + \gamma(w_{i-n+1}^{i-1}) \sum_{w_i} p^{\text{MKN}}(w_i|w_{i-n+2}^{i-1}) \\ &= \sum_{w_i} \frac{c(w_{i-n+1}^i) - d(c(w_{i-n+1}^{i-1}))}{\sum_{w_i} c(w_{i-n+1}^i)} + \gamma(w_{i-n+1}^{i-1}). \end{aligned} \tag{3.252}$$

https://doi.org/10.1017/CBO9781107295360.004 Published online by Cambridge University Press

Now we focus on the first term of the 4th line in Eq. (3.252). By using Eq. (3.245), this term can be represented as:

$$\begin{aligned}
&\sum_{w_i} c(w_{i-n+1}^{i}) - d(c(w_{i-n+1}^{i-1})) \\
&= \sum_{w_i} c(w_{i-n+1}^{i}) - d_1 \sum_{w_i | c(w_{i-n+1}^{i-1})=1} 1 - d_2 \sum_{w_i | c(w_{i-n+1}^{i-1})=2} 1 - d_{3+} \sum_{w_i | c(w_{i-n+1}^{i-1}) \geq 3} 1 \\
&= \sum_{w_i} c(w_{i-n+1}^{i}) - d_1 N_1(w_{i-n+1}^{i-1}, \bullet) - d_2 N_2(w_{i-n+1}^{i-1}, \bullet) - d_{3+} N_{3+}(w_{i-n+1}^{i-1}, \bullet).
\end{aligned} \tag{3.253}$$

Thus, by substituting Eq. (3.253) into Eq. (3.252), we obtain

$$\gamma(w_{i-n+1}^{i-1}) = \frac{d_1 N_1(w_{i-n+1}^{i-1}, \bullet) + d_2 N_2(w_{i-n+1}^{i-1}, \bullet) + d_{3+} N_{3+}(w_{i-n+1}^{i-1}, \bullet)}{\sum_{w_i} c(w_{i-n+1}^{i})}. \tag{3.254}$$

The MKN smoothed n-grams were shown to perform better than the previously described smoothed n-grams in terms of perplexity and word error rates for LVCSR tasks (Chen & Goodman 1999).

Table 3.1 summarizes different language model smoothing methods and their corresponding smoothing techniques including interpolation smoothing and backoff smoothing. (Modified) Kneser–Ney smoothing, addressed in Section 3.6.6, can be implemented in both interpolation smoothing and backoff smoothing. Bayesian approaches extend these standard n-gram language models to MAP estimation of an n-gram language model in Section 4.7, a hierarchical Dirichlet language model in Section 5.3, and a hierarchical Pitman–Yor language model in Section 8.5. Since n-gram language models always have to address the sparse data problem, Bayesian approaches provide an elegant solution to deal with the problem theoretically.

The next section considers another generative model of a text that can deal with document information rather than a simple word sequence modeled by an n-gram model.

Table 3.1 Summary of different language model smoothing methods in terms of interpolation smoothing and backoff smoothing.

	Interpolation smoothing	Backoff smoothing
Jelinek–Mercer smoothing	Section 3.6.2	
Witten–Bell smoothing	Section 3.6.3	
Absolute discount	Section 3.6.4	
Katz smoothing		Section 3.6.5
Kneser–Ney smoothing	Eq. (3.232)	Eq. (3.230)
PLSA smoothing	Eq. (3.319)	

https://doi.org/10.1017/CBO9781107295360.004 Published online by Cambridge University Press

3.7 Latent semantic information

One of the main limitations of the standard n-gram model is the inadequacy of long-distance information caused by n-gram window size. How to extract semantically meaningful information outside an n-gram window becomes crucial to achieve large-span language modeling. Traditionally, the cache-based language model (Kuhn & De Mori 1990) was proposed to exploit long-distance information where the short-term pattern in history words is continuously captured for word prediction. On the other hand, we may characterize long-distance information by finding long-term semantic dependencies which are seen as global constraints. The short-term statistics within an n-gram window serve as local constraints. Combining local and global constraints provides complete and structural information for word sequence probabilities.

In the literature, *latent semantic analysis* (LSA) (Berry, Dumais & O'Brien 1995) has been popular for many years (Manning & Schütze 1999) to construct latent topic space in information retrieval areas to evaluate the similarity between a query and a document in that space. LSA was extended to probabilistic latent semantic analysis (PLSA) (Hofmann 1999*b*, Hofmann 2001) by dealing with latent semantics as latent variables, an approach which is based on the maximum likelihood theory with the EM algorithm. Thus, PLSA provides an additional generative model to an n-gram for a text that considers long-term semantic information in the text. LSA and PLSA have been applied to develop large-span language models or topic-based language models in Bellegarda (2000) and in Gildea & Hofmann (1999), respectively, by combining these with n-gram language models, and these are addressed below.

3.7.1 Latent semantic analysis

Latent semantic analysis (LSA) introduces an additional longer-term index *document* d to word w. The document usually holds several to thousands of sentences, and the definition (e.g., a paragraph, article, journal paper, etc.) depends on target applications. Suppose there are M documents in a corpus $\mathcal{D}$, which include words with vocabulary size $|\mathcal{V}|$. LSA focuses on the $|\mathcal{V}| \times M$ word-document matrix $\mathbf{W}$:

$$\mathbf{W} = \begin{bmatrix} \omega_{1,1} & \cdots & \omega_{1,M} \\ \vdots & \vdots & \vdots \\ \omega_{|\mathcal{V}|,1} & \cdots & \omega_{|\mathcal{V}|,M} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\omega}_1 & \cdots & \boldsymbol{\omega}_M \end{bmatrix}. \tag{3.255}$$

The element of a word-document matrix $\omega_{v,m}$ is often represented based on a co-occurrence based count $c(w_{(v)}, d_m)$, which is the number of occurrences of word $w_{(v)}$ in document d_m. The word $w_{(v)} \in \mathcal{V}$ is a word in vocabulary $\mathcal{V}$ pointed by an ordered index $v \in \{1, \cdots, |\mathcal{V}|\}$, and (v) in the subscript does not denote a position in a word sequence. The co-occurrence element is often weighted by considering the importance of words in documents. As we discuss later, tf–idf (term frequency–inverse document frequency) or information-theoretic measure is used as an instance of the co-occurrence element. Note that in this word-document matrix representation, we only consider the count information (or related information) for each distinct word $w_{(v)}$ and do not consider the

https://doi.org/10.1017/CBO9781107295360.004 Published online by Cambridge University Press

sequential information of words (e.g., w_i) in a document. This feature representation of words is also called *bag of words* (Lewis 1998, Joachims 2002, Blei *et al*. 2003). The column vector $\boldsymbol{\omega}_m \in \mathbb{R}^{|\mathcal{V}|}$ can represent information about document m with a vector. This representation is called a vector space model of a text.

Document similarity

The problem of dealing with this matrix is that the column vector $\boldsymbol{\omega}_m$ is a sparse representation from natural language, and it is difficult to obtain the semantic information from this representation. For example, if we consider the similarity between documents d_m and $d_{m'}$, from the word-document matrix $\mathbf{W}$ with well-known cosine similarity, the cosine similarity is defined as

$$\cos(\mathbf{a}, \mathbf{b}) \triangleq \frac{\mathbf{a}^\intercal \mathbf{b}}{\|\mathbf{a}\| \|\mathbf{b}\|}. \tag{3.256}$$

The cosine similarity has the following property, and it is often used to measure the similarity in natural language processing:

$$-1 \leq \cos(\mathbf{a}, \mathbf{b}) \leq 1, \quad \cos(\mathbf{a}, \mathbf{a}) = 1. \tag{3.257}$$

The similarity between documents d_m and $d_{m'}$ based on the $|\mathcal{V}|$ dimensional space can be calculated as follows:

$$\mathrm{Sim}_{|\mathcal{V}|}(d_m, d_{m'}) = \cos(\boldsymbol{\omega}_m, \boldsymbol{\omega}_{m'}) = \frac{\sum_{v=1}^{|\mathcal{V}|} \omega_{vm} \omega_{vm'}}{\|\boldsymbol{\omega}_m\| \|\boldsymbol{\omega}_{m'}\|}. \tag{3.258}$$

Since ω_{vm} and $\omega_{vm'}$ are very sparse, most of the products are zero, and the cosine similarity cannot obtain meaningful similarity scores. In addition, the number of dimensions (vocabulary size $|\mathcal{V}|$, the number of documents M, or both) is too large to use. Therefore, LSA conducts a singular value decomposition (SVD) over the $|\mathcal{V}| \times M$ word-document matrix $\mathbf{W}$ and obtains

$$\mathbf{W} \approx \mathbf{U}\mathbf{S}\mathbf{V}^\intercal, \tag{3.259}$$

as shown in Figure 3.9. In Eq. (3.259), $\mathbf{S}$ is a $K \times K$ diagonal matrix with reduced dimension, $K < \min(|\mathcal{V}|, M)$, $\mathbf{U}$ is a $|\mathcal{V}| \times K$ matrix whose columns are the first K eigenvectors derived from word-by-word correlation matrix $\mathbf{W}\mathbf{W}^\intercal$, and $\mathbf{V}$ is an $M \times K$ matrix whose columns are the first K eigenvectors derived from the document-by-document correlation matrix $\mathbf{W}^\intercal\mathbf{W}$.

Document similarity in LSA

After the projection, instead of focusing on the original $|\mathcal{V}| \times M$ word-document matrix $\mathbf{W}$, LSA focuses on the factored $K \times M$ matrix $\mathbf{V}^\intercal$. Each column of $\mathbf{S}\mathbf{V}^\intercal$ characterizes the location of a particular document in the reduced K-dimensional semantic space. Therefore, we define the following document vector $\mathbf{v}_m \in \mathbb{R}^K$ as an mth column vector of $\mathbf{V}^\intercal$, which is a lower dimensional vector than $\boldsymbol{\omega}_m$:

$$\mathbf{V}^\intercal \triangleq \begin{bmatrix} \mathbf{v}_1 & \cdots & \mathbf{v}_m & \cdots & \mathbf{v}_M \end{bmatrix}. \tag{3.260}$$

https://doi.org/10.1017/CBO9781107295360.004 Published online by Cambridge University Press

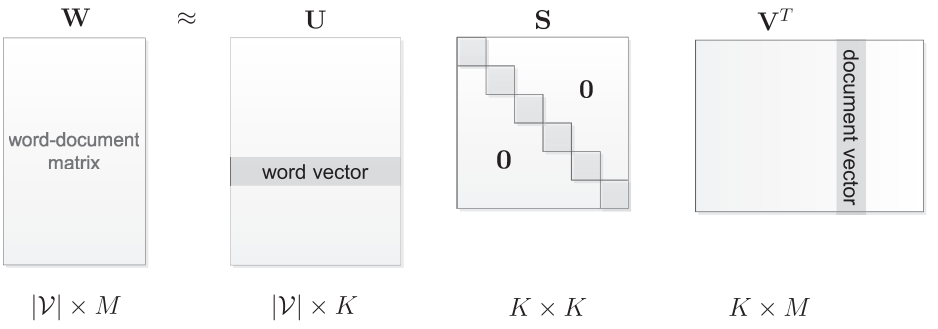

Figure 3.9 Singular value decomposition for latent semantic analysis.

This can be weighted by the diagonal matrix $\mathbf{S}$. Therefore, the similarity between documents d_m and $d_{m'}$ based on this lower K-dimensional representation is calculated as

$$\text{Sim}_K(d_m, d_{m'}) = \cos(\mathbf{S}\mathbf{v}_m, \mathbf{S}\mathbf{v}_{m'}) = \frac{\mathbf{v}_m^\mathsf{T}\mathbf{S}^2\mathbf{v}_{m'}}{\|\mathbf{S}\mathbf{v}_m\|\|\mathbf{S}\mathbf{v}_{m'}\|}. \tag{3.261}$$

Compared with Eq. (3.258), $\mathbf{S}\mathbf{v}_m$ is a denser feature, and can provide more (semantically) meaningful information (Manning & Schütze 1999).

Now, we consider the application based on information retrieval. We have a query q, which is composed of several words and can be regarded as an instance of a document. Then the problem of information retrieval is to search similar documents to query q, and computing the similarity between q and existing document d_m is very important. To evaluate the cosine similarity between the existing document d_m and this query q in the K-dimensional space, we first transform the corresponding occurrence vector $\boldsymbol{\omega}_q$ to the K-dimensional vector $\mathbf{v}_q$ as follows:

$$\mathbf{v}_q = \mathbf{S}^{-1}\mathbf{U}^\mathsf{T}\boldsymbol{\omega}_q. \tag{3.262}$$

Thus, the cosine similarity between q and d_m can be computed based on Eq. (3.261) as follows:

$$\begin{aligned} &\text{Sim}_K(d_m, q) = \cos(\mathbf{S}\mathbf{v}_m, \mathbf{S}\mathbf{v}_q) \propto \mathbf{v}_m^\mathsf{T}\mathbf{S}\mathbf{S}\mathbf{S}^{-1}\mathbf{U}^\mathsf{T}\boldsymbol{\omega}_q = (\mathbf{S}\mathbf{v}_m)^\mathsf{T}\mathbf{U}^\mathsf{T}\boldsymbol{\omega}_q \\ \Rightarrow\ &\text{Sim}_K(d_m, q) = \frac{(\mathbf{S}\mathbf{v}_m)^\mathsf{T}\mathbf{U}^\mathsf{T}\boldsymbol{\omega}_q}{\|\mathbf{S}\mathbf{v}_m\|\|\mathbf{U}^\mathsf{T}\boldsymbol{\omega}_q\|}. \end{aligned} \tag{3.263}$$

Thus, we can also obtain the similarity between the query q and document d_m with more (semantically) meaningful space, which can be used for information retrieval.

Word-document matrix

The discussion so far does not explicitly introduce a suitable value for the co-occurrence element (v, m) of the word-document matrix $\mathbf{W}$. Actually it is empirically determined, and the straightforward representation is based on the co-occurrence based count $c(w_{(v)}, d_m)$:

https://doi.org/10.1017/CBO9781107295360.004 Published online by Cambridge University Press

$$[\mathbf{W}]_{(v,m)} = \omega_{v,m} = c(w_{(v)}, d_m). \tag{3.264}$$

This is the most simple representation of $\mathbf{W}$, and this representation is actually extended as a probabilistic version of LSA (PLSA) in the next section. Another popular representation is based on tf–idf (Salton & Buckley 1988, Manning & Schütze 1999), which represents the element as follows:

$$\begin{aligned}[\mathbf{W}]_{(v,m)} = \omega_{v,m} &= \mathrm{tf}(w_{(v)}, d_m)\mathrm{idf}(w_{(v)}) \\ &= \frac{c(w_{(v)}, d_m)}{\sum_{j=1}^{|\mathcal{V}|} c(w_{(j)}, d_m)} \log \frac{M}{|\{d_m | c(w_{(v)}, d_m) > 0\}|},\end{aligned} \tag{3.265}$$

where $\mathrm{tf}(w_{(v)}, d_m)$ is called *term frequency*, defined as

$$\mathrm{tf}(w_{(v)}, d_m) \triangleq \frac{c(w_{(v)}, d_m)}{\sum_{j=1}^{|\mathcal{V}|} c(w_{(j)}, d_m)}. \tag{3.266}$$

This is computed from the co-occurrence based count $c(w_{(v)}, d_m)$ in Eq. (3.264) with a normalization factor. The quantity $\mathrm{idf}(w_{(v)})$ is called the *inverse document frequency*, and it is computed from the number of documents $|\{d_m | c(w_{(v)}, d_m) > 0\}|$ that include word $w_{(v)}$:

$$\mathrm{idf}(w_{(v)}) \triangleq \log \frac{M}{|\{d_m | c(w_{(v)}, d_m) > 0\}|}. \tag{3.267}$$

$\mathrm{idf}(w_{(v)})$ would score a lower weight for the co-occurrence element when the word $w_{(v)}$ has appeared in many documents, since such a word would be less important.

Bellegarda (2000) also proposes another weight based on an *information-theoretic measure* as follows:

$$[\mathbf{W}]_{(v,m)} = \omega_{v,m} = (1 - \varepsilon_{w_{(v)}}) \frac{c(w_{(v)}, d_m)}{\sum_{j=1}^{|\mathcal{V}|} c(w_{(v)}, d_m)}. \tag{3.268}$$

Compared with Eq. (3.265), the inverse document frequency in Eq. (3.265) is replaced with $(1 - \varepsilon_{w_{(v)}})$, where ε_v denotes the *normalized entropy* of word $w_{(v)}$ in a corpus, defined as

$$\varepsilon_{w_{(v)}} \triangleq -\frac{1}{\log M} \sum_{m=1}^{M} \frac{c(w_{(v)}, d_m)}{\sum_{j=1}^{M} c(w_{(v)}, d_j)} \log \frac{c(w_{(v)}, d_m)}{\sum_{j=1}^{M} c(w_{(v)}, d_j)}. \tag{3.269}$$

The entropy of $w_{(v)}$ would be increased when $w_{(v)}$ is distributed among many documents, which decreases the weight $(1 - \varepsilon_{w_{(v)}})$. Therefore, similarly to tf–idf, the word distributed in many documents would be less important, and have lower weight for the co-occurrence element.

3.7.2 LSA language model

The LSA language model was proposed to capture long-range word dependencies through discovery of latent topics from a text corpus. When incorporating LSA into an n-gram model, the prediction of word w_i making use of n-gram probability $p(w_i | w_{i-n+1}^{i-1})$ is calculated from two information sources in history:

https://doi.org/10.1017/CBO9781107295360.004 Published online by Cambridge University Press

1. The n-gram history words
$$h_{i-1}^{(n)} \triangleq w_{i-n+1}^{i-1}. \tag{3.270}$$
2. The long-term topic information of all history words w_1^{i-1}, which is represented by co-occurrence count (Eq. (3.264)), tf–idf (Eq. (3.265)), or information-theoretic measure (Eq. (3.268)), as discussed in the previous section:
$$h_{i-1}^{(l)} \triangleq \boldsymbol{\omega}_{i-1}. \tag{3.271}$$
The K-dimensional projected vector $\tilde{\mathbf{v}}_{i-1} \in \mathbb{R}^D$, from the history vector in the original space $\boldsymbol{\omega}_{i-1} \in \mathbb{R}^{|\mathcal{V}|}$ onto the LSA space, is obtained based on Eq. (3.262) as:
$$\tilde{\mathbf{v}}_{i-1} \triangleq \mathbf{S}\mathbf{v}_{i-1} = \mathbf{U}^{\mathsf{T}}\boldsymbol{\omega}_{i-1}. \tag{3.272}$$

By using these two types of history information, the LSA language model is represented by using the product and sum rules as:
$$\begin{aligned} p^{\mathsf{LSA}}(w_i|w_{i-n+1}^{i-1}) &\triangleq p(w_i|h_{i-1}^{(n)}, h_{i-1}^{(l)}) \\ &= \frac{p(w_i|h_{i-1}^{(n)})p(h_{i-1}^{(l)}|w_i, h_{i-1}^{(n)})}{p(h_{i-1}^{(l)}|h_{i-1}^{(n)})} \\ &= \frac{p(w_i|h_{i-1}^{(n)})p(h_{i-1}^{(l)}|w_i, h_{i-1}^{(n)})}{\sum_{w_j\in\mathcal{V}} p(w_j|h_{i-1}^{(n)})p(h_{i-1}^{(l)}|w_j, h_{i-1}^{(n)})}. \end{aligned} \tag{3.273}$$

From the definition of $h_{i-1}^{(n)}$ in Eq. (3.270), $p(w_i|h_{i-1}^{(n)})$ is represented as the following n-gram probability:
$$p(w_i|h_{i-1}^{(n)}) = p(w_i|w_{i-n+1}^{i-1}). \tag{3.274}$$
This can be calculated based on Section 3.6. Next, we focus on the distribution of the long-term topic information $p(h_{i-1}^{(l)}|w_i, h_{i-1}^{(n)})$ in Eq. (3.273). By using the definitions of $h_{i-1}^{(l)}$ and $h_{i-1}^{(n)}$ in Eqs. (3.270) and (3.271), the distribution can be rewritten as:
$$p(h_{i-1}^{(l)}|w_i, h_{i-1}^{(n)}) = p(\boldsymbol{\omega}_{i-1}|w_{i-n+1}^{i}) \approx p(\boldsymbol{\omega}_{i-1}|w_i). \tag{3.275}$$
The above approximation assumes that the document vector $\boldsymbol{\omega}_{i-1}$ only depends on w_i and does not depend on the word history w_{i-n+1}^{i-1}, in accordance with the discussion in Bellegarda (2000). This approximation would be effective when w_i is a content word. In addition, by using the Bayes theorem, $p(\boldsymbol{\omega}_{i-1}|w_i)$ can be further rewritten as follows:
$$p(\boldsymbol{\omega}_{i-1}|w_i) \propto \frac{p(w_i|\boldsymbol{\omega}_{i-1})}{p(w_i)}, \tag{3.276}$$
where $p(w_i)$ is easily calculated from a unigram probability. The $p(w_i|\boldsymbol{\omega}_{i-1})$ is a unigram probability given the document vector. Thus, the LSA language model is finally represented as follows:
$$p^{\mathsf{LSA}}(w_i|w_{i-n+1}^{i-1}) = \frac{p(w_i|w_{i-n+1}^{i-1})\frac{p(w_i|\boldsymbol{\omega}_{i-1})}{p(w_i)}}{\sum_{w_j} p(w_j|w_{i-n+1}^{i-1})\frac{p(w_j|\boldsymbol{\omega}_{i-1})}{p(w_j)}}. \tag{3.277}$$

https://doi.org/10.1017/CBO9781107295360.004 Published online by Cambridge University Press

Here, how to compute $p(w_i|\boldsymbol{\omega}_{i-1})$ is an important problem. This is performed in the projected LSA space, rather than the original high dimensional space.

The $p(w_i|\boldsymbol{\omega}_{i-1})$ is determined by the cosine measure between word w_i and pseudo document w_1^{i-1} in the LSA space, as we discussed in Eq. (3.261). First, w_i is converted with the corresponding co-occurrence vector in the $|\mathcal{V}|$ dimensional space as

$$\boldsymbol{\omega}_{w_i} = [0, \cdots, 0, \overset{v}{\overset{\vee}{1}}, 0, \cdots, 0]^\mathsf{T}. \tag{3.278}$$

Vector $\boldsymbol{\omega}_{w_i}$ is a one shot vector in the $|\mathcal{V}|$ dimensional space, where the element is 1 when $w_i = w_{(v)}$ and 0 otherwise. The document vector in the LSA space for $\boldsymbol{\omega}_{w_i}$ is obtained from Eq. (3.262) as

$$\tilde{\mathbf{v}}_{w_i} = \mathbf{S}\mathbf{v}_{w_i} = \mathbf{U}^\mathsf{T}\boldsymbol{\omega}_{w_i}. \tag{3.279}$$

Therefore, based on this equation and Eq. (3.272), the cosine similarity in the LSA K-dimensional space is obtained as:

$$\mathrm{Sim}_K(w_i, h_{i-1}^{(l)}) = \cos\left(\tilde{\mathbf{v}}_{w_i}, \tilde{\mathbf{v}}_{i-1}\right). \tag{3.280}$$

However, since a cosine value goes to negative, it cannot be used as a probability of $p(w_i|\boldsymbol{\omega}_{i-1})$ that must satisfy the non-negativity. Coccaro & Jurafsky (1998) propose the following value as a probabilistic distribution:

$$p(w_i|\boldsymbol{\omega}_{i-1}) = \frac{\left(\cos\left(\tilde{\mathbf{v}}_{w_i}, \tilde{\mathbf{v}}_{i-1}\right) - \min_{w_i' \in \mathcal{V}} \cos\left(\tilde{\mathbf{v}}_{w_i'}, \tilde{\mathbf{v}}_{i-1}\right)\right)^\gamma}{Z}, \tag{3.281}$$

where Z is a normalization constant. The minimum cosine similarity prevents a negative value, and it is also scaled by a tuning parameter γ.

In Eq. (3.280), the pseudo document vector $\boldsymbol{\omega}_{i-1}$ is recursively updated from w_{i-1} to w_i by using

$$\boldsymbol{\omega}_i = \frac{n_i - 1}{n_i}\boldsymbol{\omega}_{i-1} + \frac{1 - \varepsilon_{w_i}}{n_i}\boldsymbol{\omega}_{w_i}, \tag{3.282}$$

where n_i denotes the number of words appearing in word sequence w_1^i:

$$n_i \triangleq \sum_{v=1}^{|\mathcal{V}|} c(w_{(v)}, w_1^i), \tag{3.283}$$

and ε_{w_i} is a normalized entropy, as we defined in Eq. (3.269). Eq. (3.282) is obtained as a pseudo document vector in matrix $\mathbf{W}$, which is defined according to the entry value of matrix $\mathbf{W}$ as given in Eq. (3.268). This updating is equivalent to computing

$$\tilde{\mathbf{v}}_i = \mathbf{S}\mathbf{v}_i = \frac{1}{n_i}[(n_i - 1)\tilde{\mathbf{v}}_{i-1} + (1 - \varepsilon_{w_i})\tilde{\mathbf{v}}_{w_i}]. \tag{3.284}$$

This equation is obtained in accordance with Eqs. (3.272) and (3.279). Having the updating formula for $\boldsymbol{\omega}_{i-1}$ or $\tilde{\mathbf{v}}_{i-1}$ for pseudo document vector or history words w_1^{i-1} and the projection vector $\tilde{\mathbf{v}}_{w_i}$ of current word w_i in common topic space, we calculate the n-gram

https://doi.org/10.1017/CBO9781107295360.004 Published online by Cambridge University Press

based on topic information in Eq. (3.280) and substitute it into an LSA language model according to Eq. (3.273).

Thus, we can consider a long-term effect in a language model by using LSA. However, to integrate it with a language model, we need to provide a probabilistic treatment for LSA, e.g., Eq. (3.281), which is a rather heuristic approach. The next section introduces a fully probabilistic formulation of LSA, PLSA with an elegant inference and statistical learning methods based on the EM algorithm.

3.7.3 Probabilistic latent semantic analysis

LSA was extended to the probabilistic latent semantic analysis (PLSA) and applied for document representation in Hofmann (1999*b*) and Hofmann (2001). We first introduce a J_m-length word sequence, given a document d_m, as

$$w_1^{J_m} = \{w_1, \cdots, w_i, \cdots, w_{J_m}\} = \{w_i \in \mathcal{V} | i = 1, \cdots, J_m\} \text{ for } m = 1, \cdots, M. \quad (3.285)$$

We also introduce a set of M documents as

$$d_1^M = \{d_1, \cdots, d_M\} = \{d_m | m = 1, \cdots, M\}. \quad (3.286)$$

The first step of the ML procedure within the statistical model framework is to provide a joint likelihood function of $\mathcal{D}$ of the word sequence and document, which can be represented as:

$$p(\mathcal{D}) = p(\{w_1^{J_m}\}_{m=1}^M, d_1^M), \quad (3.287)$$

where $\mathcal{D}$ is a set of the word sequences and documents in a corpus, which is defined as:

$$\mathcal{D} \triangleq \{\{w_1^{J_m}\}_{m=1}^M, d_1^M\}. \quad (3.288)$$

Equation (3.287) can be factorized by using the product rule and the conditional independence assumption of a word generative model given document d_m as:

$$\begin{aligned} p(\mathcal{D}) &= p(\{w_1^{J_m}\}_{m=1}^M | d_1^M) p(d_1^M) \\ &= \prod_{m=1}^M p(w_1^{J_m} | d_m) p(d_m). \end{aligned} \quad (3.289)$$

This can be modeled by the word-document representation (e.g., word-document matrix $\mathbf{W}$ in Eq. (3.255)), as we discussed in Section 3.7.1. However, similarly to the LSA case, this representation has too many sparse variables, and it is difficult to deal with this representation directly. Instead, we introduce a latent topic variable $k \in \{1, \cdots, K\}$, where K is the number of topics and each topic corresponds to a higher concept of words (e.g., politics and sports, if we deal with news articles). This model is called a *latent topic model*. Then we also introduce a latent topic sequence $z_1^{J_m}$ for a corresponding word sequence $w_1^{J_m}$ for a document d_m as

$$\begin{aligned} z_1^{J_m} &= \{z_i \in \{1, \cdots, K\} | i = 1, \cdots, J_m\} \\ Z &= \{z_1^{J_m}\}_{m=1}^M. \end{aligned} \quad (3.290)$$

https://doi.org/10.1017/CBO9781107295360.004 Published online by Cambridge University Press

Then, we model that the words $w_1^{J_m}$ are conditionally independent given the corresponding topic $z_1^{J_m}$, and the probability of w_i given d_m is represented by the following mixture model:

$$p(w_i|d_m) = \sum_{k=1}^{K} p(w_i|z_i = k)p(z_i = k|d_m). \tag{3.291}$$

This model factorizes the word probability into the topic-dependent unigram probability $p(w_i|z_i = k)$ and the topic proportion probability $p(z_i = k|d_m)$. This factorization corresponds to representing the huge size of a $|\mathcal{V}| \times M$ word-document matrix with the reduced subspace K used in an LSA, as discussed in Section 3.7.1.

Thus, we can construct a likelihood function of $\mathcal{D}$ as

$$p(\mathcal{D}|\Theta) = \prod_{m=1}^{M} \prod_{i=1}^{J_m} \sum_{k=1}^{K} p(w_i|z_i = k)p(z_i = k|d_m)p(d_m). \tag{3.292}$$

This model for representing a generative process of the word sequences and documents is called probabilistic latent semantic analysis (PLSA). Here, the PLSA model parameters $\Theta = \{p(w_{(v)}|k), p(k|d_m)\}$ consist of two sets of multinomial parameters. The complete data likelihood function of $\mathcal{D}$ and Z can be represented as:

$$p(\mathcal{D}, Z|\Theta) = \prod_{m=1}^{M} \prod_{i=1}^{J_m} p(w_i|z_i)p(z_i|d_m)p(d_m). \tag{3.293}$$

Figure 3.10 provides a graphical model of PLSA and Algorithm 6 provides a generative process of PLSA.

Algorithm 6 Generative process of probabilistic latent semantic analysis.

Require: $M, J_m, \omega_m^d, \omega_k^z, \omega_{kv}^w$
1: **for** $m = 1, \cdots, M$ **do**
2: Draw d_m from Mult$(\cdot|\{\omega_m^d\}_{m=1}^M)$
3: **for** $i = 1, \cdots, J_m$ **do**
4: Draw z_i from Mult$(\cdot|\{\omega_k^z\}_{k=1}^K)$
5: Draw w_i from Mult$(\cdot|\{\omega_{kv}^w\}_{v=1}^{|\mathcal{V}|})$
6: **end for**
7: **end for**

Parameter estimation based on EM algorithm

Once we obtain the likelihood function, maximum likelihood (ML) theory is applied to estimate PLSA parameters by

$$\Theta^{\mathsf{ML}} = \arg\max_{\Theta} p(\mathcal{D}|\Theta). \tag{3.294}$$

Since PLSA has a latent variable z, it can be efficiently solved by using the EM algorithm, as we discussed in Section 3.4. The EM algorithm is introduced to find ML

https://doi.org/10.1017/CBO9781107295360.004 Published online by Cambridge University Press

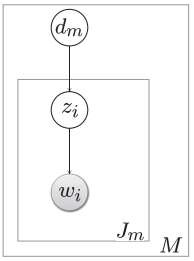

Figure 3.10 Graphical model of probabilistic latent semantic analysis.

estimates Θ^{ML}. In the E-step of the EM algorithm, we calculate an auxiliary function of new parameters Θ' given the current parameters Θ. The function is obtained by replacing HMM latent variables S and V, and speech feature observation $\mathbf{O}$ with PLSA latent variable Z and document observation $\mathcal{D}$, giving

$$\begin{aligned} Q^{\mathrm{ML}}(\Theta'|\Theta) &= \mathbb{E}_{(Z)}[\log p(\mathcal{D}, Z|\Theta')|\mathcal{D}, \Theta] \\ &= \sum_{Z} p(Z|\mathcal{D}, \Theta) \log p(\mathcal{D}, Z|\Theta'). \end{aligned} \tag{3.295}$$

By substituting Eq. (3.293) into Eq. (3.295), the auxiliary function can be rewritten as

$$\begin{aligned} Q^{\mathrm{ML}}(\Theta'|\Theta) &= \sum_{Z} p(Z|\mathcal{D}, \Theta) \log \left(\prod_{m=1}^{M} \prod_{i=1}^{J_m} p'(w_i|z_i) p'(z_i|d_m) p'(d_m) \right) \\ &= \sum_{Z} p(Z|\mathcal{D}, \Theta) \sum_{m=1}^{M} \sum_{i=1}^{J_m} \log p'(w_i|z_i) p'(z_i|d_m) p'(d_m), \end{aligned} \tag{3.296}$$

where $p'(\cdot)$ means a multinomial distribution with parameter Θ'. By using a similar trick to that used in Eq. (3.102), we can change the order of the summations, and by executing the summation over $Z_{\backslash z_i}$, Eq. (3.296) can be further rewritten as

$$\begin{aligned} Q^{\mathrm{ML}}(\Theta'|\Theta) &= \sum_{m=1}^{M} \sum_{i=1}^{J_m} \sum_{Z} p(Z|\mathcal{D}, \Theta) \log \left(p'(w_i|z_i) p'(z_i|d_m) p'(d_m) \right) \\ &= \sum_{m=1}^{M} \sum_{i=1}^{J_m} \sum_{z_i} p(z_i|\mathcal{D}, \Theta) \log \left(p'(w_i|z_i) p'(z_i|d_m) p'(d_m) \right) \\ &= \sum_{m=1}^{M} J_m \log p'(d_m) + \sum_{m=1}^{M} \sum_{i=1}^{J_m} \sum_{z_i} p(z_i|\mathcal{D}, \Theta) \log \left(p'(w_i|z_i) p'(z_i|d_m) \right). \end{aligned} \tag{3.297}$$

PLSA assumes that the above probabilistic distributions $p'(w_i|z_i)$ and $p'(z_i|d_m)$ are represented by using the multinomial distribution, which is defined in Appendix C.2 as:

https://doi.org/10.1017/CBO9781107295360.004 Published online by Cambridge University Press

$$\mathrm{Mult}(x_j|\{\omega_j\}_{j=1}^J) \triangleq \omega_j. \tag{3.298}$$

Therefore, the topic-dependent unigram distribution $p'(w_i = w_{(v)}|z_i = k)$ and topic proportion distribution $p'(z_i = k|d_m)$ are parameterized as:

$$\begin{aligned} p'(w_i = w_{(v)}|z_i = k) &= \omega'_{vk} \\ p'(z_i = k|d_m) &= \omega'_{km}. \end{aligned} \tag{3.299}$$

Thus, the parameter Θ is represented as the following set of these multinomial distributions:

$$\Theta' \triangleq \{\omega'_{vk}, \omega'_{km}|v = 1, \cdots, |\mathcal{V}|, k = 1, \cdots, K, m = 1, \cdots, M\}. \tag{3.300}$$

Now, we focus on the posterior distribution of latent variable $p(z_i|\mathcal{D}, \Theta)$. From the graphical model, z_i has the following conditional independence property:

$$\begin{aligned} p(z_i = k|\mathcal{D}, \Theta) &= p(z_i = k|w_i = w_{(v)}, d_m, \Theta) \\ &= p(k|w_{(v)}, d_m, \Theta). \end{aligned} \tag{3.301}$$

The second probability means that the probability of topic k is given by the word $w_{(v)}$ with d_m and Θ, and it is iid for the position i in d_m. Then, we can rewrite Eq. (3.297) as the following equation:

$$\begin{aligned} &Q^{\mathrm{ML}}(\Theta'|\Theta) \\ &= \sum_{m=1}^{M} J_m \log p'(d_m) + \sum_{m=1}^{M} \sum_{v=1}^{|\mathcal{V}|} c(w_{(v)}, d_m) \sum_{k=1}^{K} p(k|w_{(v)}, d_m, \Theta) \log \left(\omega'_{vk}\omega'_{km}\right), \end{aligned} \tag{3.302}$$

where $c(w_{(v)}, d_m)$ is a co-occurrence count of word $w_{(v)}$ in document d_m. Note that the summation over words in a document $\sum_{i=1}^{J_m}$ is replaced with the summation over distinct words within a dictionary $\sum_{v=1}^{|\mathcal{V}|}$. Equation (3.302) is factorized into the topic-dependent unigram parameter ω'_{vk} and the topic proportion parameter ω'_{km} as follows:

$$\begin{aligned} Q^{\mathrm{ML}}(\Theta'|\Theta) &= \sum_{m=1}^{M} J_m \log p'(d_m) \\ &\quad + \underbrace{\sum_{m=1}^{M} \sum_{v=1}^{|\mathcal{V}|} c(w_{(v)}, d_m) \sum_{k=1}^{K} p(k|w_{(v)}, d_m, \Theta) \log \left(\omega'_{vk}\right)}_{\triangleq Q^{\mathrm{ML}}(\{\omega'_{vk}\}|\{\omega_{vk}\})} \\ &\quad + \underbrace{\sum_{m=1}^{M} \sum_{v=1}^{|\mathcal{V}|} c(w_{(v)}, d_m) \sum_{k=1}^{K} p(k|w_{(v)}, d_m, \Theta) \log \left(\omega'_{km}\right)}_{\triangleq Q^{\mathrm{ML}}(\{\omega'_{km}\}|\{\omega_{km}\})} \\ &\triangleq \sum_{m=1}^{M} J_m \log p'(d_m) + Q^{\mathrm{ML}}(\{\omega'_{vk}\}|\{\omega_{vk}\}) + Q^{\mathrm{ML}}(\{\omega'_{km}\}|\{\omega_{km}\}). \end{aligned} \tag{3.303}$$

https://doi.org/10.1017/CBO9781107295360.004 Published online by Cambridge University Press

Two multinomial parameters ω_{vk} and ω_{km} should meet the constraints for probability parameters,

$$\sum_{v=1}^{|\mathcal{V}|} \omega_{vk} = 1$$
$$\sum_{k=1}^{K} \omega_{km} = 1. \tag{3.304}$$

Thus, Eq. (3.302) provides the auxiliary function of a PLSA with the parameter constraint Eq. (3.304). Therefore, similarly to Section 3.4, we can iteratively estimate parameters based on the EM algorithm.

M-step

In the M-step, we maximize the extended auxiliary function with respect to new parameters Θ' under constraints of Eq. (3.304). Again, as we discussed in Section 3.4.3, this constrained optimization problem is solved through introducing a Lagrange multiplier η and establishing the extended auxiliary function of $Q^{\mathrm{ML}}(\{\omega'_{vk}\}|\{\omega_{vk}\}$ in Eq. (3.303) as

$$\begin{aligned}&\bar{Q}(\{\omega'_{vk}\}|\{\omega_{vk}\}) \\ &= \sum_{m=1}^{M}\sum_{v=1}^{|\mathcal{V}|} c(w_{(v)}, d_m) \sum_{k=1}^{K} p(k|w_{(v)}, d_m, \Theta) \log\left(\omega'_{vk}\right) + \eta\left(\sum_{v=1}^{|\mathcal{V}|}\omega'_{vk} - 1\right).\end{aligned} \tag{3.305}$$

By differentiating Eq. (3.305) with respect to ω'_{vk} and setting it to zero, we obtain

$$\begin{aligned}\frac{\partial \bar{Q}(\{\omega'_{vk}\}|\{\omega_{vk}\})}{\partial \omega'_{vk}} &= \sum_{m=1}^{M} c(w_{(v)}, d_m) p(k|w_{(v)}, d_m, \Theta)\frac{1}{\omega'_{vk}} + \eta = 0 \\ \Rightarrow \omega'_{vk} &= -\frac{1}{\eta}\sum_{m=1}^{M} c(w_{(v)}, d_m) p(k|w_{(v)}, d_m, \Theta).\end{aligned} \tag{3.306}$$

Again, substituting this equation into constraint Eq. (3.304), we obtain the value of Lagrange multiplier η,

$$\begin{aligned}\sum_{v=1}^{|\mathcal{V}|}\omega'_{vk} &= -\frac{1}{\eta}\sum_{v=1}^{|\mathcal{V}|}\sum_{m=1}^{M} c(w_{(v)}, d_m) p(k|w_{(v)}, d_m, \Theta) = 1 \\ \Rightarrow \eta &= -\sum_{v=1}^{|\mathcal{V}|}\sum_{m=1}^{M} c(w_{(v)}, d_m) p(k|w_{(v)}, d_m, \Theta).\end{aligned} \tag{3.307}$$

Consequently, we find the following ML estimate of ω'_{vk} (i.e., the topic-dependent unigram proportion probability) for a PLSA topic model:

$$p^{\mathrm{ML}'}(w_i = w_{(v)}|z_i = k) = \omega'_{vk} = \frac{\sum_{m=1}^{M} c(w_{(v)}, d_m) p(k|w_{(v)}, d_m, \Theta)}{\sum_{v=1}^{|\mathcal{V}|}\sum_{m=1}^{M} c(w_{(v)}, d_m) p(k|w_{(v)}, d_m, \Theta)}. \tag{3.308}$$

https://doi.org/10.1017/CBO9781107295360.004 Published online by Cambridge University Press

Similarly, the parameter of topic proportion probability ω'_{km} is estimated based on the M-step as

$$p^{\mathrm{ML}'}(z_i = k|d_m) = \omega'_{km} = \frac{\sum_{v=1}^{|\mathcal{V}|} c(w_{(v)}, d_m) p(k|w_{(v)}, d_m, \Theta)}{\sum_{k'=1}^{K} \sum_{v=1}^{|\mathcal{V}|} c(w_{(v)}, d_m) p(k'|w_{(v)}, d_m, \Theta)} = \frac{\sum_{v=1}^{|\mathcal{V}|} c(w_{(v)}, d_m) p(k|w_{(v)}, d_m, \Theta)}{\sum_{v=1}^{|\mathcal{V}|} c(w_{(v)}, d_m)}. \tag{3.309}$$

Thus, we obtain the ML solutions of the PLSA parameters based on the EM algorithm.

E-step

The E-step needs to consider the following posterior distribution of the latent topic k with the previously estimated PLSA parameter Θ:

$$p(k|w_{(v)}, d_m, \Theta). \tag{3.310}$$

By using the sum and product rules and the conditional independence (Eq. (3.291)), the posterior distribution is rewritten as

$$p(k|w_{(v)}, d_m, \Theta) = \frac{p(w_{(v)}, k|d_m, \Theta)}{\sum_{k'=1}^{K} p(w_{(v)}, k'|d_m, \Theta)} = \frac{p(w_{(v)}|k, \Theta) p(k|d_m, \Theta)}{\sum_{k'=1}^{K} p(w_{(v)}|k', \Theta) p(k'|d_m, \Theta)}. \tag{3.311}$$

By using the multinomial parameters ω_{vk} and ω_{km}, this can be represented by

$$p(k|w_{(v)}, d_m, \Theta) = \frac{\omega_{vk}\omega_{km}}{\sum_{k'=1}^{K} \omega_{vk'}\omega_{k'm}}. \tag{3.312}$$

Thus, the posterior distribution can be computed by using the previously estimated PLSA parameter Θ. We summarize the parameter estimation algorithm of the PLSA in Algorithm 7. In a practical implementation, storing the above posterior distribution $p(k|w_{(v)}, d_m, \Theta)$ explicitly needs a huge size of memory (that corresponds to store $M \times |\mathcal{V}| \times K$ data), and it is impossible to do that for large-scale text data. Instead, the PLSA parameter update is performed by using a matrix multiplication based operation. Actually, this PLSA parameter update corresponds to the multiplicative matrix update of non-negative matrix factorization (NMF) (Lee & Seung 1999) based on the KL divergence cost function (Gaussier & Goutte 2005).

PLSA is another well-known generative model in addition to the n-gram model that generates words with exponential family distributions (multinomial distribution). Since the generative model is represented with exponential family distributions, PLSA can be extended by using Bayesian approaches, as we discussed in Section 2.1.3. This extension is called the latent Dirichlet allocation (Blei *et al*. 2003), where we use a Dirichlet distribution as a conjugate prior, which is discussed in Section 7.6. We also show that PLSA parameters can be estimated efficiently by using the EM algorithm. However, PLSA does not consider the short-span language property, unlike an n-gram model, and

https://doi.org/10.1017/CBO9781107295360.004 Published online by Cambridge University Press

the next section discusses how to involve the PLSA and n-gram for the use of a language model.

Algorithm 7 EM algorithm for PLSA

Require: $\Theta \leftarrow \Theta^{\text{init}}$, $\{c(w_{(v)}, d_m)|v = 1, \cdots, |\mathcal{V}|, m = 1, \cdots, M\}$

repeat
  for $m = 1, \cdots, M$ **do**
    for $v = 1, \cdots, |\mathcal{V}|$ **do**
      for $k = 1, \cdots, K$ **do**
        Compute the posterior probability $p(k|w_{(v)}, d_m, \Theta)$
      end for
    end for
  end for
  for $v = 1, \cdots, |\mathcal{V}|$ **do**
    for $k = 1, \cdots, K$ **do**
      Compute the topic-dependent unigram probability
      $p^{\text{ML}'}(w_i = w_{(v)}|z_i = k) = \omega'_{vk}$
    end for
  end for
  for $m = 1, \cdots, M$ **do**
    for $k = 1, \cdots, K$ **do**
      Compute the topic proportion probability
      $p^{\text{ML}'}(z_i = k|d_m) = \omega'_{km}$
    end for
  end for
  $\Theta \leftarrow \Theta'$
until Convergence

3.7.4 PLSA language model

In Gildea & Hofmann (1999), Akita & Kawahara (2004) and Mrva & Woodland (2004), a PLSA framework was applied to estimate the n-gram model, which is seen as a combination of large-span and short-span language models. That is, we consider the short-span language model (n-gram)

$$p(w_i|w_{i-n+1}^{i-1}) \tag{3.313}$$

and long-span language model

$$p(w_i|\mathcal{D}) \approx \sum_{k=1}^{K} p(w_i|k)p(k|\mathcal{D}) \tag{3.314}$$

to obtain the probability of w_i. As an alternative, a long-span language model $p(w_i|\mathcal{D})$, a cache-based language model (Kuhn & De Mori 1990), or a trigger-based language

https://doi.org/10.1017/CBO9781107295360.004 Published online by Cambridge University Press

model (Lau, Rosenfeld & Roukos 1993) are used as conventional approaches. As an example of the document $\mathcal{D}$, we consider the history of all words from the beginning w_1^{i-1}. Given word sequence w_1^{i-1}, latent topic probabilities of the history $p(k|w_1^{i-1})$ are inferred by using a PLSA model and incorporated into an n-gram language model.

To obtain the latent topic probabilities $p(k|w_1^{i-1})$, maximum likelihood PLSA parameters $\Theta = \{p(w_{(v)}|k), p(k|d_m)\} = \{\omega_{vk}, \omega_{km} | v = 1, \cdots, |\mathcal{V}|, k = 1, \cdots, K, m = 1, \cdots, M\}$ are estimated in advance from a large amount of training documents in the training step, as we discussed in the previous section. Then, given Θ as an initial parameter set of Algorithm 7, we estimate $p(k|w_1^{i-1})$ in the test step. However, to avoid over-training as we only use w_i^{i-1} and to avoid a high computational cost in the test step, we only update $p^{\mathrm{ML}'}(z_i = k|d_m) = \omega'_{km}$ with fewer iterations than the full estimation of the PLSA parameter Θ in the training step. In addition, we use the following *online EM* algorithm (Neal & Hinton 1998) to update topic posterior probabilities to make the estimation in an on-line manner for every word position i:

$$\begin{aligned} p(k|w_1^{i-1}) &= \frac{1}{i+1} \frac{p(w_{i-1} = v|k) p(k|w_1^{i-2})}{\sum_{k'=1}^{K} p(w_{i-1} = v|k') p(k'|w_1^{i-2})} + \frac{i}{i+1} p(k|w_1^{i-2}) \\ &= \frac{1}{i+1} \frac{\omega_{vk} p(k|w_1^{i-2})}{\sum_{k'=1}^{K} \omega_{vk'} p(k'|w_1^{i-2})} + \frac{i}{i+1} p(k|w_1^{i-2}). \end{aligned} \tag{3.315}$$

Here, $p(k|w_1^{i-2})$ in the first and second terms in the right-hand-side is obtained by the previous estimation, and the first term is computed on-the-fly based on a pre-computation of $p(w_{i-1}|k)$, which is obtained from ω_{vk} in Eq. (3.308). The factors $\frac{1}{i+1}$ and $\frac{i}{i+1}$ denote a linear interpolation ratio, and the contribution of the second term becomes larger when the length of the word sequence i is larger. These factors are derived as a specific solution of the on-line EM algorithm, and Mrva & Woodland (2004) suggest modifying these factors for practical use.

In an initialization stage (i.e., $w_1^{i=0} = \emptyset$ in Eq. (3.184)), the initial topic posterior probability could be obtained from a prior topic probability in the training stage as follows:

$$\begin{aligned} p(k|w_1^{i=0}) &= p(k) \\ &= \sum_{m=1}^{M} p(k|d_m) p(d_m) \\ &= \frac{\sum_{m=1}^{M} \sum_{v=1}^{|\mathcal{V}|} c(w_{(v)}, d_m) p(k|d_m)}{\sum_{m=1}^{M} \sum_{v=1}^{|\mathcal{V}|} c(w_{(v)}, d_m)} \\ &= \frac{\sum_{m=1}^{M} \sum_{v=1}^{|\mathcal{V}|} c(w_{(v)}, d_m) \omega_{km}}{\sum_{m=1}^{M} \sum_{v=1}^{|\mathcal{V}|} c(w_{(v)}, d_m)}, \end{aligned} \tag{3.316}$$

where $p(d_m)$ is estimated from the maximum likelihood equation based on the word co-occurrence counts $c(d_m) = \sum_{v=1}^{|\mathcal{V}|} c(w_{(v)}, d_m)$, i.e.,

$$p(d_m) = \frac{c(d_m)}{\sum_{m=1}^{M} c(d_m)}. \tag{3.317}$$

https://doi.org/10.1017/CBO9781107295360.004 Published online by Cambridge University Press

Once we obtain the topic proportion probability $p(k|w_1^{i-1})$, the PLSA based n-gram probability is finally calculated by

$$p^{\text{PLSA}}(w_i = v|w_1^{i-1}) = \sum_{k=1}^{K} p(w_i = v|k)p(k|w_1^{i-1}) = \sum_{k=1}^{K} \omega_{vk} p(k|w_1^{i-1}). \tag{3.318}$$

Instead of hard-clustering computation in class-based n-gram in Eq. (3.198), PLSA n-gram in Eq. (3.318) performs a so-called *soft-clustering* computation by marginalizing the topic index k in a Bayesian sense. This model is known as a large-span language model because long-distance topics $k = 1, \cdots, K$ and history words w_1^{i-1} within the n-gram window as well as outside the n-gram window are all taken into account for word prediction. However, since the computation can be performed by considering the history words w_1^{i-1}, it is not pre-computed, unlike n-gram models, which makes the implementation of the PLSA language model harder than n-gram models in ASR.

Smoothing with n-grams

PLSA n-grams could be further improved by combining with ML n-grams based on additional linear interpolation with factor λ:

$$\hat{p}(w_i|w_1^{i-1}) = \lambda p^{\text{ML}}(w_i|w_{i-n+1}^{i-1}) + (1-\lambda)p^{\text{PLSA}}(w_i|w_1^{i-1}). \tag{3.319}$$

The interpolation parameter λ could be found by maximizing likelihood or tuned by maximizing the ASR performance of some validation data.

We can also use a unigram rescaling technique, which is approximated based on the conditional independence assumption of w_1^{i-1} and w_{i-n+1}^{i-1}:

$$\hat{p}(w_i|w_1^{i-1}) \approx p^{\text{ML}}(w_i|w_{i-n+1}^{i-1}) \frac{p^{\text{PLSA}}(w_i|w_1^{i-1})}{p^{\text{ML}}(w_i)}. \tag{3.320}$$

Note that this representation does not hold the sum-to-one condition, and it must be normalized.

A more precise unigram rescaling technique can be performed by using the dynamic unigram marginal (also known as the minimum discrimination information (MDI) adaptation), which can consider backoff probabilities in an n-gram probability (Kneser, Peters & Klakow 1997, Niesler & Willett 2002, Tam & Schultz 2006). First, we define the following unigram rescaling factor π for word w_i:

$$\pi(w_i, w_1^{i-1}) \triangleq \left(\frac{p^{\text{PLSA}}(w_i|w_1^{i-1})}{p^{\text{ML}}(w_i)} \right)^{\rho}. \tag{3.321}$$

https://doi.org/10.1017/CBO9781107295360.004 Published online by Cambridge University Press

Then the new n-gram probability with a history of n-gram word sequence w_{i-n+1}^{i-1} is represented as follows:

$$p(w_i|w_{i-n+1}^{i-1}, w_1^{i-1}) = \begin{cases} \frac{\pi(w_i, w_1^{i-1})}{C_0(w_{i-n+1}^{i-1}, w_1^{i-1})} p(w_i|w_{i-n+1}^{i-1}) & \text{if} \quad c(w_{i-n+1}^{i}) > 0 \\ \frac{1}{C_1(w_{i-n+1}^{i-1}, w_1^{i-1})} p(w_i|w_{i-n+2}^{i-1}, w_1^{i-1}) & \text{otherwise,} \end{cases} \tag{3.322}$$

where $p(w_i|w_{i-n+1}^{i-1})$ is an n-gram probability obtained in advance, and C_0 and C_1 are normalization constants, which can be computed by:

$$C_0(w_{i-n+1}^{i-1}, w_1^{i-1}) = \frac{\sum_{\{w_i|c(w_{i-n+1}^{i})>0\}} \pi(w_i, w_1^{i-1}) p(w_i|w_{i-n+1}^{i-1})}{\sum_{\{w_i|c(w_{i-n+1}^{i})>0\}} p(w_i|w_{i-n+1}^{i-1})}, \tag{3.323}$$

and

$$C_1(w_{i-n+1}^{i-1}, w_1^{i-1}) = \frac{1 - \sum_{\{w_i|c(w_{i-n+1}^{i})>0\}} p(w_i|w_{i-n+2}^{i-1}, w_1^{i-1})}{1 - \sum_{\{w_i|c(w_{i-n+1}^{i})>0\}} p(w_i|w_{i-n+1}^{i-1})}. \tag{3.324}$$

Then, $p(w_i|w_{i-n+2}^{i-1})$ is also iteratively calculated by $\pi(w_i, w_1^{i-1})$, $n-1$-gram probability $p(w_i|w_{i-n+2}^{i-1})$, and $p(w_i|w_{i-n+3}^{i-1})$. Thus, the unigram rescaled language model is obtained by modifying the backoff coefficients.

In this chapter, we address several language model methods tackling the issues of small sample size and long-distance information. Different language model smoothing methods including interpolation smoothing, backoff smoothing and unigram smoothing are summarized. The Bayesian language modeling based on MAP estimation and VB learning is addressed in Section 4.7 and Section 7.7, respectively.

3.8 Revisit of automatic speech recognition with Bayesian manner

In Chapter 2, we introduce the basic mathematical tools in the Bayesian approach, including the sum and product rules, and the conditional independence, and provide simple ways to obtain the posterior distributions. In addition, throughout the discussions from Section 3.1 to Section 3.7, we also introduce statistical models of speech and language, which are mainly used for ASR. Based on these mathematical tools and statistical models, this section revisits methods to formulate the whole ASR framework in a Bayesian manner, as consistently as possible, by only using the sum rule, product rule, and conditional independence, as discussed in Section 2.1. In particular, we focus on how to train our ASR model based on the Bayesian manner. The aim of this section is to show the limitation of simply using basic Bayesian tools for ASR, which leads to the requirement of approximated Bayesian inference techniques in the following chapters.

3.8.1 Training and test (unseen) data for ASR

To deal with the ASR problem within a machine learning framework, we need to consider two sets of data: *training* data and *test* data. The training data usually consist of

https://doi.org/10.1017/CBO9781107295360.004 Published online by Cambridge University Press

a sequence of speech feature vectors ($\mathbf{O}$) and the corresponding label sequence (W) for all utterances. The test data consist of only a sequence of speech feature vectors ($\mathbf{O}'$) for all utterances, and there is a candidate label sequence (W') among all possible word sequences. The ASR problem is to correctly estimate the corresponding label sequence ($\hat{W}'$). Therefore, we summarize these four variables as:

- $\mathbf{O}$: speech feature sequence for training (observed);
- W: word sequence for training (observed);
- $\mathbf{O}'$: speech feature sequence for test (observed);
- W': a candidate of word sequences for test (not observed).

Since we have various possible candidates for W', it is natural to consider the following conditional probabilistic distribution of W', given the other observed variables, as:

$$p(W'|\mathbf{O}', \mathbf{O}, W). \tag{3.325}$$

Therefore, this section starts with the formulation based on this conditional probability.

Note that if we want to estimate the correct word sequence, it is performed by using the well-known MAP decision rule, as we discussed in Eq. (3.2):

$$\hat{W}' = d^{\mathrm{MAP}}(\mathbf{O}') = \arg\max_{W'} p(W'|\mathbf{O}', \mathbf{O}, W). \tag{3.326}$$

Therefore, the following section focuses on $p(W'|\mathbf{O}', \mathbf{O}, W)$ in more detail.

3.8.2 Bayesian manner

Recalling the discussion in Section 2.1, the Bayesian manner deals with these variables as the arguments of probability distributions as follows:

$$\begin{aligned} \mathbf{O} &\rightarrow p(\mathbf{O}) \\ W &\rightarrow p(W) \\ \mathbf{O}' &\rightarrow p(\mathbf{O}') \\ W' &\rightarrow p(W'). \end{aligned} \tag{3.327}$$

We should also recall that these probabilistic variables (x: continuous variable and n: discrete variable) have the following mathematical properties:

$$p(n) \geq 0, \tag{3.328}$$

$$\sum_{n} p(n) = 1, \tag{3.329}$$

$$p(x) \geq 0, \tag{3.330}$$

$$\int p(x)dx = 1. \tag{3.331}$$

These properties yield various benefits for probabilistic pattern recognition problems. Based on this probabilistic variable treatment, we can apply the sum, product rules for all the variables, and we can also use the Bayes theorem for Eq. (3.325).

https://doi.org/10.1017/CBO9781107295360.004 Published online by Cambridge University Press

MAP decision rule

By replacing a with word sequence W' and b with speech feature vector $\mathbf{O}'$ in Bayes theorem Eq. (2.7) in Section 2.1.1, we can derive the following noisy channel model of speech recognition based on the well-known MAP decision rule (Eq. (3.2)):

$$\hat{W}' = d^{\text{MAP}}(\mathbf{O}') = \arg\max_{W'} p(W'|\mathbf{O}'), \tag{3.332}$$

$$= \arg\max_{W'} \frac{p(\mathbf{O}'|W')p(W')}{p(\mathbf{O}')}, \tag{3.333}$$

$$= \arg\max_{W'} p(\mathbf{O}'|W')p(W'). \tag{3.334}$$

Here, the denominator of Eq. (3.333) is disregarded with the arg max operation, since $p(\mathbf{O}')$ does not depend on W'.

The product rule (Bayes theorem) based on the MAP decision theory decomposes the posterior distribution $p(W'|\mathbf{O}')$ into the two generative models of $\mathbf{O}'$ and W', i.e., acoustic model $p(\mathbf{O}'|W')$ and language model $p(W')$. Solving Eq. (3.332) by directly obtaining $p(W'|\mathbf{O}')$ is called the *discriminative approach*. The approach tries to infer what a speaker wants to say in his/her brain from the observation $\mathbf{O}'$, as shown in Figure 3.11.

On the other hand, the *generative approach* obtains $p(W'|\mathbf{O}')$ indirectly via the generative models $p(\mathbf{O}'|W')$ and $p(W')$ based on Eq. (3.333). Therefore, the approach tries to imitate the generation process of $\mathbf{O}'$ given W' in the acoustic model and W itself in the language model, as shown in Figure 3.12. Since the generation process comes from a physical phenomenon or linguistic phenomena, we can involve various knowledge of the phenomena (e.g., articulatory models in speech production and grammatical or semantic models in linguistics) in principle. The rest of this section further discusses the Bayesian formulation along with the generative approach.

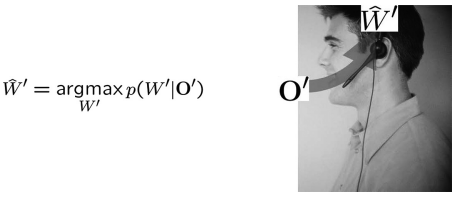

Figure 3.11 Probabilistic speech recognition: discriminative approach.

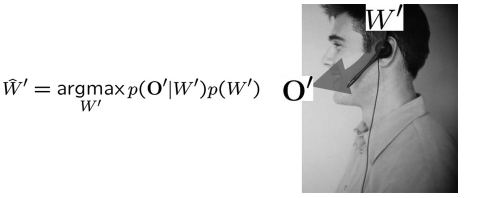

Figure 3.12 Probabilistic speech recognition: generative approach.

https://doi.org/10.1017/CBO9781107295360.004 Published online by Cambridge University Press

3.8.3 Learning generative models

Bayesian formulation reasonably extends the generative models to teach them from training data, by adding training data ($\mathbf{O}$ and W) to conditional variables in the target distribution, namely,

$$\begin{aligned} p(\mathbf{O}'|W') &\rightarrow p(\mathbf{O}'|W', \mathbf{O}, W) \\ p(W') &\rightarrow p(W'|\mathbf{O}, W). \end{aligned} \tag{3.335}$$

Usually, we also use the reasonable conditional independence assumption for the language model expressed by $p(W'|\mathbf{O}, W) \approx p(W'|W)$. Now, we focus on the acoustic model $p(\mathbf{O}'|W', \mathbf{O}, W)$, and make this abstract distribution more concrete.

3.8.4 Sum rule for model

To make the distribution more concrete, we usually provide a *model* with the distribution. For example, when we have some data to analyze, we first start to consider what kind of models we use to make the problems concrete by considering HMMs or GMMs for speech feature sequences, and n-gram for word sequences. However, we do not know whether the model provided is correct or not, and it should be tested by different model settings. Thus, the model settings can be regarded as probabilistic variables in the Bayesian formulation. In addition, the variation of setting model topologies, distribution forms of prior and posterior distributions, and hyperparameters of these distributions can also be treated as probabilistic variables, as shown in Table 3.2. We call this probabilistic variable to denote the model setting *model* variable M.

For example, once we decide to use an HMM for speech feature modeling, we have many arbitrary properties for the model topology: (i) whether we use a word unit, context-independent phoneme unit, or context-dependent phoneme unit; (ii) left to right HMM, including skip transitions, or fully connected ergodic HMM; (iii) how many shared-triphone HMM states and how many Gaussians in these states. Some of the model variables dealing with this model structure are called *model structure* variables, and so-called structure learning in the machine learning field tries to optimize the model structure using a more general framework than the Bayesian framework.

Table 3.2 Examples of model variables.

Variation	Examples
Model types	HMM, GMM, SVM, neural network, etc.
Model topologies	# HMM states, # Gaussians, # hidden layers, etc.
Priors/posteriors	Gaussian, Laplace, Gamma, Bernoulli, etc.
Hyperparameters	parameters of priors/posteriors, kernel parameters, etc.

https://doi.org/10.1017/CBO9781107295360.004 Published online by Cambridge University Press

The model variable M can be involved in our abstract distribution (Eq. (3.335)) of the acoustic model by using the sum rule (Eq. (2.3)) as follows:

$$p(\mathbf{O}'|W', \mathbf{O}, W) = \sum_{M} p(\mathbf{O}', M|W', \mathbf{O}, W). \tag{3.336}$$

As M would include continuous values (e.g., hyperparameters), the marginalization over M should involve both summation and integration. However, for simplicity, we use the summation in this formulation.

3.8.5 Sum rule for model parameters and latent variables

Once we set a model variable to a specific value, we can also provide the corresponding model parameters Θ and latent variables Z for training data and Z' for test data. For example, once we decide a model for the specific setting based on a standard HMM–GMM with a fixed model topology, these variables can also be involved in the distribution with M (Eq. (3.336)) by using the sum rule (Eq. (2.3)) as follows:

$$p(\mathbf{O}'|W', \mathbf{O}, W) = \int \sum_{M,Z,Z'} p(\mathbf{O}', M, \Theta, Z, Z'|W', \mathbf{O}, W) d\Theta. \tag{3.337}$$

However, it is really difficult to deal with this joint distribution, and we need to factorize the distribution.

3.8.6 Factorization by product rule and conditional independence

First, we factorize Eq. (3.337) by using the product rule (Eq. (2.4)), as follows:

$$\begin{aligned} p(\mathbf{O}'|W', \mathbf{O}, W) &= \int \sum_{M,Z,Z'} p(\mathbf{O}', M, \Theta, Z, Z'|W', \mathbf{O}, W) d\Theta \\ &= \int \sum_{M,Z,Z'} p(\mathbf{O}', Z'|Z, \Theta, M, W', \mathbf{O}, W) p(Z|\Theta, M, W', \mathbf{O}, W) \\ &\qquad \times p(\Theta|M, W', \mathbf{O}, W) p(M|W', \mathbf{O}, W) d\Theta. \end{aligned} \tag{3.338}$$

In this formulation, we keep the joint distribution of $\mathbf{O}'$ and Z' and do not factorize them. This is because the distribution corresponds to the complete data likelihood function, which is useful to handle latent models based on the EM algorithm, as discussed in Section 3.4.

Since the dependencies of the distributions in Eq. (3.338) are very complicated, we use the reasonable conditional independence assumptions for these distributions as follows:

$$p(\mathbf{O}', Z'|Z, \Theta, M, W', \mathbf{O}, W) \approx p(\mathbf{O}', Z'|\Theta, M, W'), \tag{3.339}$$

$$p(Z|\Theta, M, W', \mathbf{O}, W) \approx p(Z|\Theta, M, \mathbf{O}, W), \tag{3.340}$$

$$p(\Theta|M, W', \mathbf{O}, W) \approx p(\Theta|M, \mathbf{O}, W), \tag{3.341}$$

$$p(M|W', \mathbf{O}, W) \approx p(M|\mathbf{O}, W). \tag{3.342}$$

https://doi.org/10.1017/CBO9781107295360.004 Published online by Cambridge University Press

Equation (3.339) means that the test data are generated by the Θ and M and do not depend on the training data ($\mathbf{O}$ and W) and their latent variable Z, explicitly. The other assumptions just use the conditional independence of W'. By using the assumptions, we can approximate the original distribution of $p(\mathbf{O}'|W', \mathbf{O}, W)$ as follows:

$$\begin{aligned}\text{Eq. (3.338)} &\approx \int \sum_{M,Z,Z'} p(\mathbf{O}', Z'|M, \Theta, W')p(Z|\Theta, M, \mathbf{O}, W) \\ &\quad \times p(\Theta|M, \mathbf{O}, W)p(M|\mathbf{O}, W)d\Theta \\ &\approx \int \sum_{M,Z'} p(\mathbf{O}', Z'|M, \Theta, W')p(\Theta|M, \mathbf{O}, W)p(M|\mathbf{O}, W)d\Theta, \end{aligned} \tag{3.343}$$

where we use the fact that $\sum_Z p(Z|\Theta, M, \mathbf{O}, W) = 1$ since Z does not depend on the other distributions. We also introduce lexical category c, and further factorize the equation as:

$$\text{Eq. (3.343)} \approx \int \sum_{M,Z'} \prod_{c'} p(\mathbf{O}', Z'|M, \Theta, c') \prod_{c} p(\Theta|M, \mathbf{O}, c)p(M|\mathbf{O}, c)d\Theta. \tag{3.344}$$

Thus, the acoustic model can be represented by the joint distribution of $\mathbf{O}'$ and Z' (complete data likelihood) and the posterior distributions of model parameters Θ and model M. Since $p(\mathbf{O}', Z'|M, \Theta, c')$ does not depend on training data ($\mathbf{O}$ and W), which can be obtained by setting the model and its parameters, we only focus on the two posterior distributions in Eq. (3.344).

3.8.7 Posterior distributions

The posterior distributions of models and model parameters can be rewritten by the following equations by using the sum and product rules (Eqs. (2.3) and (2.4)).

- **Model parameter** Θ:

$$p(\Theta|M, \mathbf{O}) = \frac{p(\mathbf{O}|\Theta, M)p(\Theta|M)}{p(\mathbf{O}|M)}, \tag{3.345}$$

$$\propto \sum\nolimits_Z p(\mathbf{O}, Z|\Theta, M)p(\Theta|M). \tag{3.346}$$

- **Model** M:

$$p(M|\mathbf{O}) = \frac{p(\mathbf{O}|M)p(M)}{\sum_M p(\mathbf{O}|M)p(M)}, \tag{3.347}$$

$$\propto \sum\nolimits_Z \int p(\mathbf{O}, Z|\Theta, M)p(\Theta|M)d\Theta p(M). \tag{3.348}$$

Note that the posterior distributions are represented by the following two types of distributions:

- Joint distribution of training data $\mathbf{O}$ and Z:
  $p(\mathbf{O}, Z|\Theta, M)$;
- Prior distributions of Θ and M :
  $p(\Theta|M)$ and $p(M)$.

https://doi.org/10.1017/CBO9781107295360.004 Published online by Cambridge University Press

Again, the joint distribution can be obtained by setting the model and its parameters. Once we set the prior distributions $p(\Theta|M)$ and $p(M)$, we can obtain the posterior distributions $p(\Theta|M, \mathbf{O})$ and $p(M|\mathbf{O})$ by solving (3.346) and (3.348). Then, the acoustic model likelihood can be computed by using Eq. (3.344).

3.8.8 Difficulties in speech and language applications

Although the Bayesian approach is powerful, it is very difficult to realize. One of the most critical aspects is that we cannot solve the above equations. The practical Bayesian approach rests on how to find appropriate approximations:

- *Chapter 4*: *Maximum a-posteriori approximation;*
- *Chapter 5: Evidence approximation;*
- *Chapter 6: Asymptotic approximation;*
- *Chapter 7: Variational Bayes;*
- *Chapter 8: Markov chain Monte Carlo.*

In the machine learning field, other approximations are also studied actively (e.g., loopy belief propagation (Murphy *et al.* 1999), expectation propagation (Minka 2001)). This book introduces the above approximations in speech and language applications.

https://doi.org/10.1017/CBO9781107295360.004 Published online by Cambridge University Press