

Project Dissertation
M.Phil in Computer Speech, Text and Internet Technology

SAMPLING METHODS FOR INSTANTANEOUS SPEAKER
ADAPTATION

Supervisor:
Mark Gales

Matt Shannon
Trinity College, Cambridge
sms46@cam.ac.uk

19 June 2008

Declaration of originality

I, Sean Matthew Shannon, of Trinity College, being a candidate for M.Phil in Computer Speech, Text and Internet Technology, hereby declare that this dissertation and the work described in it are my own work, unaided except as may be specified below, and that the dissertation does not contain material that has already been used to any substantial extent for a comparable purpose.

Sean Matthew Shannon
19 June 2008

Acknowledgements

I would like to thank my supervisor, Mark Gales, for many stimulating and exciting discussions in the course of this project. I would also like to thank C.K. Raut for invaluable and patient advice; and Kai Yu, who's work on Variational Bayes was of continual help in thinking about the Bayesian approach to adaptation, and who kindly allowed me access to his Variational Bayes experimental data.

Word count

This document contains roughly 14,600 words, including footnotes, appendices and bibliography, which is within the prescribed limit of 15,000 words.

Abstract

Linear mean-based adaptation schemes, in which an affine linear transformation is applied to the mean of all Gaussian components in an LVCSR system, have been very successful as an approach to speaker adaptation. In particular, *Maximum Likelihood Linear Regression (MLLR)* offers a simple EM approach to finding the maximum likelihood (ML) transform. However, robustness issues arise when using the ML transform to adapt to a new speaker on very little data, due to the fact that the transform posterior is diffuse – intuitively, we don’t have a good ‘fix’ on the speaker yet – and so this distribution is badly approximated by picking the ML transform. Thus recognition using the ML transform can deviate significantly from the true *transform marginalized* recognition process, leading to poor performance.

This project takes a Bayesian approach to the adaptation process, within the linear mean-based framework. The integrals associated with this approach are intractable in general, so in this project approximate schemes are explored.

A new variational lower bound, and a new high-level Gibbs sampling scheme are developed, and compared to existing methods on a single utterance Conversational Telephone Speech task.

Contents

Contents	i
1 Introduction	1
1.1 Introduction to Bayesian adaptation	1
2 Background	3
2.1 Graphical models	3
2.2 Hidden Markov Models (HMMs)	4
2.3 Models for speech and N -best rescoring	6
2.4 HMMs for speech in the N -best rescoring framework	7
2.5 A note on WER	7
2.6 Gibbs sampling	8
2.7 Energy	9
3 Adaptation	11
3.1 Linear mean-based adaptation	11
3.2 MLLR	11
3.3 MAPLR	12
3.4 Bayesian approach	12
3.5 Transform distributions	14
3.6 Multiple baseclasses	18
4 Lower bound approximations	19
4.1 An introduction to variational lower bounds	19
4.2 Variational lower bounds for adaptation	20
4.3 Variational Bayes	20
4.4 Transform marginalized lower bound	21
5 Sampling methods	27
5.1 Overview	27
5.2 How to use samples	27
5.3 High-level Gibbs sampling	28
5.4 Acoustic deweighting	30
6 Experiments	35
6.1 Experimental set-up	35
6.2 Reduced- N experiments	36
6.3 Transform marginalized lower bound	36
6.4 High-level Gibbs sampling	36

7	Results	39
7.1	Reduced- N experiments	39
7.2	Transform marginalized lower bound	40
7.3	High-level Gibbs sampling	40
8	Conclusion	47
A	Sampling from a Gaussian distribution	49

Chapter 1

Introduction

1.1 Introduction to Bayesian adaptation

This is an introduction to the problem we tackle in this project. It assumes some background in speech, on the level of the CSTIT M.Phil course. For an introduction to the *tools* we use see §2.

The scenario we consider is as follows. We have a single utterance from a new, unseen, speaker, and an N -best list of hypotheses about what word sequence they said, produced for example by a Viterbi decoder. We assume that this speaker's HMM is transformed by some transform W from the generic, canonical model.

One approach to doing recognition in this scenario is pick your best estimate of the transform, then do recognition taking this transform as given. This is what MLLR and its close cousin MAPLR do, and this seems sensible enough at first glance.

However, viewing this set-up in a Bayesian way, we regard the transform W as just another random variable, similar to the state sequence. Just as we sum or marginalize over the state sequence when working out an utterance probability using Forwards-Backwards, so we want to marginalize over the transform when doing adaptation. We have a prior over transforms – some transforms are viewed as inherently more likely than others – just as we have a prior over state sequences (given by the transition probabilities) or word sequences (given by the language model) in the standard framework.

After the system hears an utterance there is now a *posterior* over transforms – a distribution representing which transform the system thinks the speaker is using. When doing Bayesian recognition, we consider all these possible transforms, exactly in proportion to how likely they are according to the transform posterior. In this view, MLLR and MAPLR approximate the entire posterior distribution by a point estimate, and use that as the basis for recognition.

When is the Bayesian approach likely to give very different results to MLLR or MAPLR? Precisely when this transform posterior is *diffuse* – not heavily spiked in one place. Since we expect that the more data we get the more we learn about the speaker's transform, and so the more heavily spiked the transform posterior gets, the situation in which we're most likely to notice a difference between Bayesian and non-Bayesian methods is when doing

adaptation on very little data. Hence we consider the case of a single utterance (on average, about 3 seconds of speech).

However, even for simple adaptation frameworks like linear mean-based adaptation, the integral required to marginalize over the transform is in general intractable. Therefore we consider approximations to the full Bayesian integral.

Our approach can be divided into two very different styles of approximating scheme:

- lower bound approximations
- sampling methods

These have a very different flavour: lower bound approaches guarantee something about the integral they approximate, but there is typically a limit to how good the approximation can be made; sampling approaches have no guarantee at any stage, but asymptotically reach the true distribution they are trying to approximate. Thus the two approaches are complementary.

The emphasis of the project is on recognition results, rather than computational efficiency.

Chapter 2

Background

2.1 Graphical models

We will use *graphical models* as a concise and visual way to express and verify conditional independence relations. A (directed) graphical model is a directed acyclic graph whose nodes are random variables, where the connections in the graph indicate how we can write the joint distribution in terms of conditionals. To take a concrete example, suppose we have the following joint distribution over 4 random variables:

$$P(x, y, z, w) = P(w|z)P(z|x, y)P(x)P(y)$$

This decomposition is expressed as the graphical model in Figure 2.1¹. By convention we shade a node to indicate its random variable is observed, for example W here.

The general rule is that a graphical model over a given set of random variables corresponds to a decomposition of the joint distribution of those random variables – each node X gives rise to exactly one term $P(X|\text{pa}(X))$ in the decomposition, where we condition on the parents $\text{pa}(X)$ of the node in the graphical model.

We shall largely just be using the obvious properties of graphical models, and a detailed understanding of them is not necessary here. However, there is one trick that is useful enough to be worth mentioning here. The *Markov blanket* of a node is the set of the node's

¹Many thanks to Henrik Bengtsson, whose latex package is used to generate the graphical models shown here.

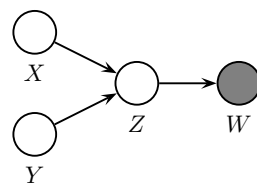


Figure 2.1: An example graphical model

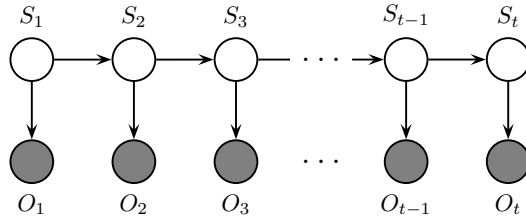


Figure 2.2: Graphical model for general HMM

parents, children, and children’s parents. For example, in Figure 2.1 the Markov blanket of X is Y and Z . The result we will use is that a node is conditionally independent of everything else in the graphical model, given its Markov blanket. For instance, X and W are conditionally independent given Y and Z . And W and (X, Y) are conditionally independent given Z .

2.2 Hidden Markov Models (HMMs)

2.2.1 Introduction

We will briefly summarize the theory of HMMs that will be relevant in what follows.

An HMM consists of a *hidden state space* (with an special ‘initial’ state identified), a transition probability a_{qr} for each pair of states q and r , an *output feature space*, and a probability distribution $b_q(v)$ over the output feature space (v being an element of this feature space) for each state q . This defines a generative model over state space sequences (S_t) and output feature sequences (O_t) , by first using the transition probabilities to make (S_t) into a Markov chain:

$$P(s_{1:T}) \triangleq a_{\text{initial}, s_1} a_{s_1 s_2} a_{s_2 s_3} \cdots a_{s_{T-1} s_T}$$

and then define O_t to be conditionally independent of everything else given S_t , with distribution $b_{s_t}(o_t)$. Thus the joint probability of a state sequence (s_t) and output sequence (o_t) is:

$$P(s_{1:T}, o_{1:T}) = a_{\text{initial}, s_1} a_{s_1 s_2} a_{s_2 s_3} \cdots a_{s_{T-1} s_T} \\ \cdot b_{s_1}(o_1) b_{s_2}(o_2) \cdots b_{s_T}(o_T)$$

For use in automatic speech recognition (ASR), the output feature space is frequently a 39-dimensional real vector space, and we can use a whole gamut of methods, which we will not discuss here, to go from the original waveform to a sequence of elements in this feature space, where successive elements of the sequence represent the waveform at successive times, typically 10ms apart.

A graphical model for this general HMM is shown in Figure 2.2.

We typically restrict the probability distributions $b_q(v)$ to be a *Gaussian mixture model (GMM)*, that is a weighted sum of Gaussian distributions:

$$b_q(v) = \sum_{m=1}^M c_m \mathcal{N}(v; \mu_m, \Sigma_m)$$

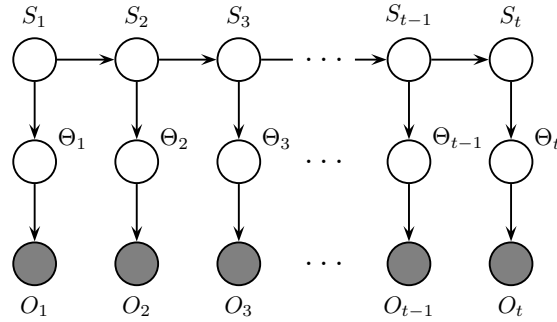


Figure 2.3: Graphical model for mixture-model HMM

where m indexes the *components* of the Gaussian mixture, c_m is the *mixture weight* of that component ($\sum_m c_m = 1$), μ_m and Σ_m are the mean and covariance matrix of the Gaussian for that component, and M is some pre-specified number of components, typically 16. We could equally specify each Gaussian component in terms of its mean μ_m and *precision matrix* P_m , where $P_m = \Sigma_m^{-1}$.

We can therefore view the generative model as having an extra layer, the *component sequence* (Θ_t). Here Θ_t is conditionally independent of everything else given S_t , with distribution given by the mixture weights (c_{θ_t}), and O_t is now conditionally independent of everything else given Θ_t , with distribution given by the Gaussian for component θ_t . A graphical model showing these conditional independencies is shown in Figure 2.3. Note that conceptually we are viewing each component as belonging to only one state. This is valid even if in practice the component distributions are shared.

Since we will be talking about sequences a lot, we will often write them without subscripts. For example, we write o to mean an *entire* output sequence, θ to mean an entire component sequence, and s to mean an entire state sequence. We can therefore write:

$$P(o|\theta) = \prod_t \mathcal{N}(o_t; \mu_{\theta_t}, \Sigma_{\theta_t}) \quad (2.1)$$

In practice, it is useful to have *non-emitting* states which have no output distribution, only transition probabilities to other states. When working with HMMs containing non-emitting states, we usually specify that there should be no *null loops*, that is no paths through state space with non-zero probability which pass only through non-emitting states and return where they started. Given this restriction, it is usually fairly easy, if slightly fiddly, to figure out how to go from the all-emitting case to the case where we have non-emitting nodes. For the sake of simplicity of notation and clarity of exposition, we will only give formulae for the all-emitting case here, the only exception being that we sometimes consider non-emitting initial (every path must start here at time 0) and final states (every path must end here at time $T + 1$).

2.2.2 The Forwards-Backwards algorithm

There exists an efficient method to compute certain marginal and conditional distributions in an HMM. Define:

$$\begin{aligned}\alpha_q(t) &\triangleq P(o_{1:t}, S_t = q) \\ \beta_q(t) &\triangleq P(o_{t+1:T} | S_t = q)\end{aligned}$$

Then we have the following recursions:

$$\alpha_q(t) = \sum_p \alpha_p(t-1) a_{pq} b_q(o_t) \quad (2.2)$$

$$\beta_q(t) = \sum_r a_{qr} b_r(o_{t+1}) \beta_r(t+1) \quad (2.3)$$

We can therefore efficiently compute α and β .

We can then compute a number of quantities of interest. Firstly, note that

$$\beta_{\text{initial}}(0) = P(o_{1:T}) = P(o) \quad (2.4)$$

which is the probability of output marginalized over all possible state sequences. Also

$$\alpha_q(t) \beta_q(t) = P(o, S_t = q)$$

so we can compute the *state occupancies*:

$$\begin{aligned}\gamma_q(t) &\triangleq P(S_t = q | o) \\ &= \frac{P(o, S_t = q)}{P(o)} \\ &= \frac{\alpha_q(t) \beta_q(t)}{P(o)}\end{aligned} \quad (2.5)$$

From here we can easily compute the *component occupancies*:

$$\begin{aligned}\gamma_m(t) &\triangleq P(\Theta_t = m | o) \\ &= P(\Theta_t = m | s_t, o) P(s_t | o) \\ &= P(\Theta_t = m | s_t, o_t) \gamma_q(t) \\ &= \frac{P(o_t | \theta_t) P(\theta_t | s_t)}{P(o_t | s_t)} \gamma_q(t)\end{aligned} \quad (2.6)$$

where we've used conditional independence relations to simplify the expression.

2.3 Models for speech and N -best rescoring

When discussing speech, we have the output O , which is in some way representative of the acoustic speech waveform, and the sequence of words H that the speaker said. We decompose the joint probability of a given output and word sequence as follows:

$$P(o, h) = P(o|h)P(h)$$

where we refer to the $P(o|h)$ term as the *acoustic model* and the $P(h)$ as the *language model*. Within this framework, we might view recognition as the process of computing the most likely word sequence, given some output:

$$\arg \max_h P(h|o) = \arg \max_h P(o|h)P(h) \quad (2.7)$$

To do recognition, we could combine the acoustic and language models into one big model, and somehow consider all possible word sequences in an efficient way. This is what is done for Viterbi decoding. Alternatively, we could restrict the set of word sequences we consider to be some small set, and compute the above argmax only over this small set.

This approach, where we have a restricted list of N hypotheses about what the word sequence might have been and find the most likely hypothesis within the list, can be called *N -best rescoring*. It has the advantage that we can more easily incorporate more complicated language and acoustic models, and is the approach we will use in this project. Note that for this approach to be successful we have to choose our small set of possibilities wisely, and usually they are generated using something like Viterbi decoding.

2.4 HMMs for speech in the N -best rescoring framework

We haven't yet specified how to use HMMs for speech. In our N -best rescoring framework (§2.3), we just have to specify the acoustic model $P(o|h)$ for each word sequence h . We will do this by defining an HMM for each word sequence.

In fact, our hypotheses will not be exactly word sequences, but 'word sequences with pronunciations'. We consider a pronunciation of a word to be a string of phones from some finite phone set. In addition to the language model, we have a *pronunciation dictionary* which specifies the possible pronunciations of a word and a probability for each of these. Combined, the language model and pronunciation dictionary thus give the probability of any hypothesis $P(h)$.

Our building blocks for constructing the acoustic model HMM for the hypothesis h are a set of triphone models. A triphone is a triple of previous phone, current phone, and next phone, and our models are small HMMs with a predefined topology and non-emitting start and exit states. To construct the HMM for the hypothesis h , we simply take the string of triphones corresponding to h , and string the associated triphone models together in the same order, connecting the exit state of one model to the start state of the next. This concatenated HMM *defines* our acoustic model $P(o|h)$ for a given hypothesis h .

We note in passing that these are exactly the HMMs used to train the parameters of the triphone models using *embedded training*.

2.5 A note on WER

Our standard metric for evaluating speech recognition systems is *word error rate (WER)*. Thus surely our goal is not to compute the *most likely* hypothesis according to the posterior $P(h|o)$, but to pick the hypothesis that minimizes expected WER. These are not the same thing. For example, if we had 3 hypotheses all close to each other in terms of edit distance

with posterior $P(h|o) = \frac{2}{9}$, and 1 hypothesis far from all of them with posterior $P(h|o) = \frac{3}{9}$, then according to the ‘most likely hypothesis’ criterion we’d choose the 1. However, to minimize expected WER we would be much better off picking one of the group of 3.

We *will* assume for the rest of this project that we are just trying to pick the most likely hypothesis. However, we note here that our high-level Gibbs sampling procedure in fact estimates the entire posterior distribution $P(h|o)$ over the N -best list. This opens up the fascinating possibility of directly minimizing expected WER, or any other metric we choose, using a simple form of *Minimum Bayes Risk* analysis.

2.6 Gibbs sampling

Gibbs sampling is a method to sample from a joint distribution over several random variables by sampling repeatedly from their conditional distributions. For the sake of explanation, consider the concrete case of three random variables (X, Y, Z) with joint distribution $P(x, y, z)$. To do Gibbs sampling we somehow choose an initial value (x_0, y_0, z_0) , then repeatedly sample from the conditional distributions $P(x|y, z)$, $P(y|z, x)$ and $P(z|x, y)$ – that is, we update the value of X with a sample from $P(x|y, z)$, leaving y and z fixed, then update the value of Y with a sample from $P(y|z, x)$, then update Z , and repeat. In fact, we are free to choose the order in which we update the variables to be any fixed (or random, as long as it doesn’t depend on the samples) order we like.

For typical distributions and update sequence choices, this will converge to the true joint distribution as we iterate. In fact, the procedure can be seen as defining a Markov chain on the state space of the random variables (x, y, z) [2]. The true joint distribution is an invariant distribution of this chain, so ‘typical’ above means precisely that required for ergodicity of this Markov chain.

There are several things we must consider when actually running Gibbs sampling. In order to sample from the correct distribution, we must first do many iterations of (called *burn-in*) to ensure we’ve waited long enough that the distribution we’re sampling from has converged to the true one. After this, no matter how many iterations we wait between taking successive samples, they will still be samples from the correct distribution. However, it is customary to wait for enough iterations that we can be reasonably sure that successive samples are independent. This is because when using these samples to estimate something (typically via a simple Monte Carlo approximation of an integral), we get an *unbiased* estimate even if the samples are strongly correlated, but it is very hard to estimate the variance, and so assess how much confidence we have in our estimate. With independent samples, we can make definite claims on the variance. The time we wait between taking successive samples after convergence is generally much less than the burn-in time allowed to reach convergence.

We typically run the Gibbs sampling procedure more than once from the beginning, including burn-in. There is a trade-off between the amount of time we devote to each of these *runs* and the number of runs. Experts seem to favour the ground somewhere in the middle of this trade-off, allowing each run to generate lots of samples but also doing several runs [2].

Gibbs sampling does have weaknesses, the most potent being that if some of the random variables are strongly correlated, it effectively explores this part of the state space by a random walk [2]. For example, this is the case for a 2-dimensional Gaussian with a covariance

matrix like:

$$\Sigma = \begin{pmatrix} 1.0 & 0.998 \\ 0.998 & 1.0 \end{pmatrix}$$

(example taken from [2]). Another example is where we have two or more isolated peaks where the line between two peaks isn't aligned with any co-ordinate axis, e.g. a peak at $(0, 0)$ and a peak at $(1, 1)$. In cases like these convergence can be *very* slow.

2.7 Energy

Any discrete or continuous probability distribution can be written in the form

$$P(x) = \frac{1}{Z} \exp(-E(x))$$

where we refer to $E(x)$ as the *energy*, by an analogy with statistical physics. (We allow $E(x) = +\infty$ so P can be zero). Conversely, any function $E(x)$ defines a valid probability distribution by the above relation. Note that adding a constant to the energy just changes the normalizing constant Z , and leaves the normalized distribution $P(x)$ unchanged – that is, the energy for a given distribution is only unique up to an additive constant.

For a Gaussian $X \sim \mathcal{N}(\mu, \Sigma)$ with precision matrix $P = \Sigma^{-1}$ the energy is

$$E(x) = \frac{1}{2} \sum_{i,j} P^{ij} x^i x^j - \sum_i b^i x^i + \text{const} \quad (2.8)$$

where

$$b^i = \sum_j P^{ij} \mu^j \quad (2.9)$$

Given b , we can recover the mean μ as:

$$\mu^i = \sum_j \Sigma^{ij} b^j \quad (2.10)$$

Thus we can either consider a given Gaussian distribution as defined by its precision matrix and mean (P, μ) , or as defined by (P, b) . The latter form will be encountered frequently in what follows.

Chapter 3

Adaptation

3.1 Linear mean-based adaptation

We can use the above HMMs for speaker independent (SI) recognition. However, we obtain gains by adapting our model to new speakers and acoustic conditions. Such *adaptation* can take many forms, but here we will just consider the case where we transform the Gaussian component means by an affine transform:

$$\mu \mapsto A\mu + b$$

where A is a matrix / linear map and b is a constant *bias* vector. For simplicity of notation we combine A and b into one matrix, writing $W = \begin{pmatrix} A & b \end{pmatrix}$. Each speaker has their own transform associated with them, but which transform a given speaker is using is of course unknown. Recognition now becomes estimating the word sequence, taking all the possible transforms the speaker could be using into account.

We will call this form of adaptation *Linear mean-based adaptation*.

3.2 MLLR

The conventional approach to linear mean-based adaptation is *Maximum Likelihood Linear Regression (MLLR)*. Given a hypothesis h , we compute the maximum likelihood transform:

$$w_{\text{ML}} \triangleq \arg \max_w P(o|w, h)$$

Within the N -best rescoring framework, we will consider the case where we estimate a separate transform $w_{\text{ML}}^{(h)}$ for each hypothesis (so-called *N -best supervision*), then use $P(o|w_{\text{ML}}^{(h)}, h)$ as some sort of approximation to $P(o|h)$ when computing (2.7).

We compute the ML transform using a form of EM. Consider the auxiliary function:

$$\mathbb{E}_Q [\log P(o|\theta, w)] \tag{3.1}$$

where $Q(\theta)$ is a distribution over component sequences. It turns out we can express this in terms of w and three *accumulators* g , b and c , which can in turn be expressed in a closed

form in terms of the component occupancies $\gamma_m(t)$ under Q (§3.5.2). We then maximize with respect to w . If we choose $Q(\theta) = P(\theta|o, w_{\text{old}}, h)$, where w_{old} is our previous estimate of the transform, then these component occupancies are exactly those computed by the Forwards-Backwards algorithm (§2.2.2). This procedure is guaranteed not to decrease the likelihood $P(o|\theta, w)$, and we can use it iteratively to refine our estimate of w_{ML} .

3.3 MAPLR

Maximum A Posteriori Linear Regression (MAPLR) additionally incorporates the transform prior $P(w)$. We now choose the maximum a posteriori transform

$$w_{\text{MAP}} \triangleq \arg \max_w P(w|o, h) = \arg \max_w P(o|w, h)P(w)$$

and use $P(o|w_{\text{MAP}}^{(h)}, h)$ as our approximation to $P(o|h)$ when computing (2.7).

We can compute the MAP transform similarly using a form of EM. We now use the auxiliary function:

$$\mathbb{E}_Q [\log P(o, w|\theta)] = \mathbb{E}_Q [\log P(o|w, \theta)] + P(w) \quad (3.2)$$

where we choose $Q(\theta) = P(\theta|o, w_{\text{old}}, h)$ as for MLLR. In fact, the maths is very similar to that for MLLR. We can again express the auxiliary function in terms of w and three accumulators, \bar{g} , \bar{b} and \bar{c} , and these are just g , b and c with a ‘bias’ term added due to the prior (§3.5.3).

3.4 Bayesian approach

The Bayesian view of this adaptation process is that the transform the speaker is using is again a random variable, just like the state sequence or word sequence. Thus it has a prior associated with it, and the correct way to work out the probability of a given output is to *marginalize* over all possible transforms, in the same way that we marginalize over all possible state sequences when computing the probability of a given output for an unadapted HMM using the Forwards-Backwards algorithm.

A graphical model for this adaptation HMM is shown in Figure 3.1.

Taking everything so far together, for a given utterance we have:

- an N -best list of hypotheses, and a prior probability $P(h)$ over these hypotheses given by the language model
- a transform for the speaker and a prior $P(w)$ over possible transforms
- an acoustic model HMM of the form in Figure 3.1, given the transform w and the hypothesis h

Since the component sequence Θ in our acoustic model HMM doesn’t depend on the transform (we only transform the Gaussian component means), we can write the relationship between O , W , Θ and H as the graphical model in Figure 3.2. Here O denotes the entire output *sequence* as discussed previously, and similarly for Θ . Equivalently, we can

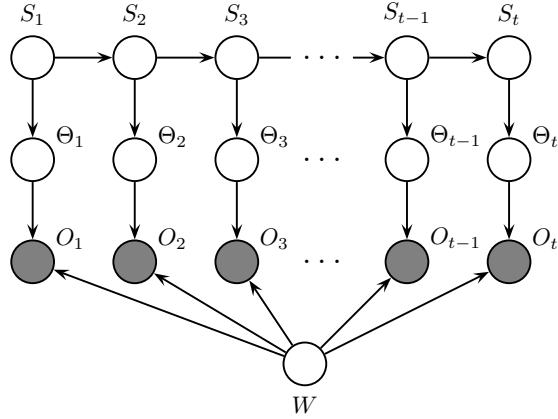


Figure 3.1: Graphical model for adaptation HMM

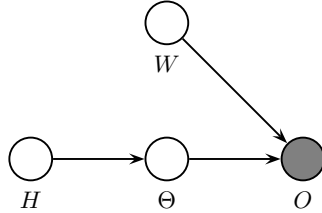


Figure 3.2: High-level graphical model relevant to adaptation

decompose the joint distribution as:

$$P(o, w, \theta, h) = P(o|\theta, w)P(w)P(\theta|h)P(h)$$

Therefore, to do adaptation in a Bayesian way, our goal is to *marginalize* over w and θ :

$$P(o, h) = \sum_{\theta} \int P(o|\theta, w)P(w)P(\theta|h)P(h) dw \quad (3.3)$$

picking the hypothesis h that maximizes this expression. In particular, it would be sufficient to be able to marginalize $P(o, \theta, w|h)$ over w and θ :

$$P(o|h) = \sum_{\theta} \int P(o|\theta, w)P(w)P(\theta|h) dw \quad (3.4)$$

since within the N -best framework we can just consider all possible hypotheses.

Note that we are only considering Bayesian *adaptation* in this project – we use models *trained* in a conventional maximum likelihood style.

3.4.1 Conjugate transform prior

In general, a class of prior probability distributions ($P_\alpha(\lambda)$) is said to be *conjugate* to a class of likelihood distributions ($P_\beta(x|\lambda)$) if for any members of the prior and likelihood class, the posterior $P(\lambda|x)$ belongs to the prior class. This often aids both analytic and experimental work – the former since the conjugate prior classes often have nice properties, and the latter since, if the prior class is described by a finite set of hyperparameters, we only need to keep track of these hyperparameters rather than an entire distribution when doing inference.

For us, the likelihood function in question is $P(o|w, \theta)$ as a function of w , which has an (unnormalized) Gaussian shape (§3.5.2). In fact, assuming our original models have diagonal covariance matrices, this likelihood can be factored over the rows of the transform. Following [3], we therefore pick the intuitively simplest prior class which is conjugate to this likelihood – another Gaussian for each row of the transform, with the distribution for each row independent.

In fact, much of what follows relies on choosing a prior of something like this form. This is necessary for Variational Bayes and our new transform marginalized bound, and our specific high-level Gibbs sampling procedure also relies on it. We suspect that each of these methods could at least be extended to work for priors which are finite mixtures of distributions of the above form, but have not investigated this in detail.

3.4.2 Estimating the transform prior

Since we’re only considering Bayesian *adaptation*, and not Bayesian *training*, estimating the transform prior doesn’t really fit into remit of this project. However, we have to use something, and there isn’t a well established literature as there is for training the HMMs themselves. We therefore use a procedure known as *Empirical Bayes*, which is just a name for using the maximum likelihood hyperparameters. In our case, these hyperparameters are the mean and covariance of the prior distribution for each row of the transform, and [3] shows how to do this as part of adaptive training.

3.5 Transform distributions

In this section we develop some of the maths for the form of adaptation used in this project, namely where we transform the Gaussian component means with an affine transform:

$$\mu \mapsto A\mu + b$$

Writing $W = \begin{pmatrix} A & b \end{pmatrix}$ and $\xi = \begin{pmatrix} \mu \\ 1 \end{pmatrix}$ the transformed mean is $\tilde{\mu} = W\xi$.

We’ll start by considering a general EM-like expression, and specialize this in several ways to be useful for both the lower bound and the sampling aspects of this project.

3.5.1 EM-like auxiliary function

Let Q be a probability distribution over the component sequence Θ . Consider an EM-like auxiliary function

$$\begin{aligned} \mathbb{E}_Q [-\log P(o, \theta, w|h)] &= \mathbb{E}_Q [-\log P(o|\theta, w)] \\ &\quad + \mathbb{E}_Q [-\log P(\theta|h)] \\ &\quad + [-\log P(w)] \end{aligned} \quad (3.5)$$

This is exactly the auxiliary function considered for MAPLR (§3.3), except for the second term, which we conventionally drop since it doesn't depend on w .

3.5.2 Transform likelihood

Expanding the first term to its individual Gaussian components and substituting in the transformed mean, we get

$$\mathbb{E}_Q [-\log P(o|\theta, w)] = \frac{1}{2} \sum_{i,j,k,l} g^{ijkl} w^{ik} w^{jl} - \sum_{i,k} b^{ik} w^{ik} + \frac{1}{2} c$$

where

$$\begin{aligned} g^{ijkl} &= \sum_m \gamma_m P_m^{ij} \xi_m^k \xi_m^l \\ b^{ik} &= \sum_m \sum_j \gamma_m P_m^{ij} \hat{\mu}_m^j \xi_m^k \\ c &= \sum_m \sum_{i,j} \gamma_m P_m^{ij} \hat{S}_m^{ij} + \sum_m \gamma_m C_m \\ \gamma_m(t) &= Q(\{\Theta_t = m\}) \\ \gamma_m &= \sum_t \gamma_m(t) \\ C_m &= n \log 2\pi - \log |P_m| \\ \hat{\mu}_m^i &= \frac{\sum_t \gamma_m(t) o_t^i}{\sum_t \gamma_m(t)} \\ \hat{S}_m^{ij} &= \frac{\sum_t \gamma_m(t) o_t^i o_t^j}{\sum_t \gamma_m(t)} \end{aligned}$$

Comparing the above with (2.8), we see that this is the energy of a Gaussian distribution over the transform components w^{ij} (considering the set of matrix components to be dimensions for the multivariate Gaussian).

The above expression is exactly the auxiliary function in (3.1) that we use when doing EM for MLLR. For MLLR, we take the maximum of this Gaussian distribution, which is at its mean. This can be computed from g^{ijkl} and b^{ik} above, as in [1].

If the original precision matrices are diagonal

$$P_m^{ij} = \tau_m^i \delta^{ij}$$

then we get the following simplification:

$$\mathbb{E}_Q[-\log P(o|\theta, w)] = \sum_i \left(\frac{1}{2} \sum_{k,l} g^{ikl} w^{ik} w^{il} - \sum_k b^{ik} w^{ik} \right) + \frac{1}{2}c \quad (3.6)$$

where

$$g^{ikl} = \sum_m \gamma_m \tau_m^i \xi_m^k \xi_m^l \quad (3.7a)$$

$$b^{ik} = \sum_m \gamma_m \tau_m^i \hat{\mu}_m^i \xi_m^k \quad (3.7b)$$

$$c = \sum_m \sum_i \gamma_m \tau_m^i \hat{S}_m^{ii} + \sum_m \gamma_m C_m \quad (3.7c)$$

which is the energy of a Gaussian distribution for each row (w^i), with the rows independent.

In the rest of this project we will only consider this case, namely where the original precision matrices (equivalently, covariance matrices) are diagonal.

3.5.3 Incorporating the transform prior

As discussed in §3.4.1 we use a transform prior $P(w)$ which is conjugate to the likelihood $P(o|w, \theta)$, namely another Gaussian distribution for each row with rows independent:

$$-\log P(w) = \sum_i \left(\frac{1}{2} \sum_{k,l} P_{\text{prior}}^{ikl} w^{ik} w^{il} - \sum_{k,l} P_{\text{prior}}^{ikl} \mu_{\text{prior}}^{il} w^{ik} \right) + \frac{1}{2}c_{\text{prior}}$$

where μ_{prior}^i and P_{prior}^i are the mean and precision matrix for the i^{th} row.

Combining the first and third term of (3.5), we get:

$$\mathbb{E}_Q[-\log P(o, w|\theta)] = \sum_i \left(\frac{1}{2} \sum_{k,l} \bar{g}^{ikl} w^{ik} w^{il} - \sum_k \bar{b}^{ik} w^{ik} \right) + \frac{1}{2}\bar{c} \quad (3.8)$$

where

$$\bar{g}^{ikl} = g^{ikl} + P_{\text{prior}}^{ikl} \quad (3.9a)$$

$$\bar{b}^{ik} = b^{ik} + \sum_l P_{\text{prior}}^{ikl} \mu_{\text{prior}}^{il} \quad (3.9b)$$

$$\bar{c} = c + c_{\text{prior}} \quad (3.9c)$$

which is again the energy of a Gaussian distribution for each row (w^i), with the rows independent. Using (2.10), we can write the mean vector of the distribution for the i^{th} row, $\bar{\mu}^i$, in terms of the corresponding covariance matrix $\bar{\Sigma}^i$, and \bar{b}^i :

$$\bar{\mu}^{ij} = \sum_k \bar{\Sigma}^{ijk} \bar{b}^{ik} \quad (3.10)$$

The covariance matrix $\bar{\Sigma}^i$ for the i^{th} row is just the matrix inverse of the corresponding precision matrix \bar{g}^i .

The above expression is exactly the auxiliary function in (3.2) that we use for EM for MAPLR. For MAPLR, we take the maximum of this Gaussian distribution, which is at its mean $\bar{\mu}^{ij}$.

3.5.4 Single component forms

We will be sampling from the above distributions, or maximizing them for MLLR and MAPLR. To do this directly we have to compute the mean and covariance, which involves inverting a roughly $n \times n$ matrix for each row of the transform. (For sampling we'd also need to compute the Cholesky decomposition of each matrix, though this may be a byproduct of matrix inversion as is the case in HTK). This isn't horrendous, but it is interesting to consider the case of updating each transform component w^{ij} separately. The marginals will also be Gaussian, and from (3.8) we can calculate the mean μ and precision τ of the corresponding Gaussian distribution for w^{ij} :

$$\begin{aligned}\tau &= \bar{g}^{ijj} \\ \mu &= \frac{1}{\bar{g}^{ijj}} \left(\bar{b}^{ij} - \sum_{k \neq j} \bar{g}^{ijk} w^{ik} \right)\end{aligned}$$

which is analogous to [1, equation (14)] (and corrected).

3.5.5 Fixed component sequence

If our distribution Q over component sequences is just one fixed component sequence θ^* :

$$Q(\theta) = \delta(\theta, \theta^*)$$

then

$$\gamma_m(t) = \delta_{m, \theta^*(t)}$$

and so

$$g^{ikl} = \sum_t \tau_{\theta^*(t)}^i \xi_{\theta^*(t)}^k \xi_{\theta^*(t)}^l \quad (3.11a)$$

$$b^{ik} = \sum_t \tau_{\theta^*(t)}^i o_t^i \xi_{\theta^*(t)}^k \quad (3.11b)$$

$$c = \sum_t \sum_i \tau_{\theta^*(t)}^i (o_t^i)^2 + \sum_t C_{\theta^*(t)} \quad (3.11c)$$

In this case (3.8) becomes $-\log P(o, w|\theta^*)$, so the Gaussian distribution over w in (3.8) is $P(o, w|\theta^*)$ up to a constant, or $P(w|o, \theta^*)$ exactly.

3.5.6 Analytic marginalization over transform

Since (3.8) is the energy for a Gaussian distribution – specifically, the Gaussian distribution defined by (\bar{g}, \bar{b}) – it follows that:

$$e^{\mathbb{E}_Q[\log P(o, w|\theta)]}$$

is a constant times this Gaussian distribution. We can therefore integrate this expression over w – it’s just the constant, since the integral over the Gaussian distribution is 1. Working out this constant explicitly, and taking the negative log of both sides, we obtain:

$$-\log \left(\int e^{\mathbb{E}_Q[\log P(o,w|\theta)]} dw \right) = \frac{1}{2}\bar{c} - \frac{1}{2} \sum_{i,k,l} \bar{\Sigma}^{ikl} \bar{b}^{ik} \bar{b}^{il} - \frac{1}{2}n(n+1) \log 2\pi + \frac{1}{2} \sum_i \log |\bar{g}^i| \quad (3.12)$$

where $|\cdot|$ is the determinant, and $\bar{\Sigma}^i$ is the covariance matrix for the i^{th} row, as in §3.5.3.

In the case of a fixed component sequence θ^* , the expectation on the left hand side effectively disappears, to leave:

$$-\log P(o|\theta^*) = \frac{1}{2}\bar{c} - \frac{1}{2} \sum_{i,k,l} \bar{\Sigma}^{ikl} \bar{b}^{ik} \bar{b}^{il} - \frac{1}{2}n(n+1) \log 2\pi + \frac{1}{2} \sum_i \log |\bar{g}^i| \quad (3.13)$$

Thus we have analytically marginalized $P(o, w|\theta^*)$ over w to give $P(o|\theta^*)$.

Note that if we have computed values for \bar{g} , \bar{b} , \bar{c} and $\bar{\Sigma}$ for some θ , and we change the value of θ_t at one time t , then we can efficiently compute these values for the new θ :

- the changes to g , b and c are given by a single term in (3.11), and so is fast to compute. Since the prior constants are fixed, this also gives the changes to \bar{g} , \bar{b} and \bar{c} .
- the new value of $\bar{\Sigma}^i$, which is the matrix inverse of \bar{g}^i , will be close to the old value since \bar{g}^i is close to its old value. Thus we can use Jacobi iteration seeded by the old value of $\bar{\Sigma}^i$ to efficiently compute its new value.

This is useful for Rao Blackwellized Gibbs Sampling (§4.4.1).

3.6 Multiple baseclasses

We may choose to use multiple transforms for each speaker – we can partition the set of models into groups, and use one transform per group. These groups of models are commonly called *baseclasses*. We assume that the transforms used for each group are independent, and so use separate priors for each baseclass.

For clarity, we do not explicitly consider the multiple baseclass case elsewhere. However, we have endeavoured to check the multiple baseclass extension to our formulae is both reasonably intuitive given what’s presented here, and valid. Our experiments in fact use two baseclasses, one for silence and one for speech.

Chapter 4

Lower bound approximations

In this chapter we consider approximations based on *variational lower bounds*. We review *Variational Bayes (VB)*, the variational lower bound of choice explored in [3]. We then introduce a new lower bound, which we will refer to as the *transform marginalized* lower bound. This turns out to be closely related to Variational Bayes, though it comes from a fairly different starting point.

4.1 An introduction to variational lower bounds

In general, a *variational lower bound* for the integral

$$G = \int g(x) dx$$

is a function $f(x; \alpha)$ where α is an adjustable (*variational*) parameter, such that for any choice of α :

$$F(\alpha) = \int f(x; \alpha) dx \leq \int g(x) dx = G$$

If it happens that for our chosen α , $f(x; \alpha) = g(x)$, then the lower bound becomes an equality $F(\alpha) = G$. Thus it makes sense to *define* the tightness of the above bound as a way to measure how good an approximation $f(x; \alpha)$ is to $g(x)$ – the tighter the bound the better we consider the approximation.

A typical procedure is to choose the value of α such that the bound is as tight as possible, then compute $F(\alpha)$, and use that as an approximation to G in subsequent calculations. In other words, we use the $f(x; \alpha)$ that is the ‘best’ approximation to $g(x)$, where best is defined by the above measure of similarity.

The class of approximating functions $f(x; \alpha)$ is generally in some sense simpler than the true $g(x)$ (otherwise we could just compute G directly). So typically the bound isn’t attained, even for the best α .

4.2 Variational lower bounds for adaptation

In the adaptation setting the game is to approximate the tricky $P(o|h)$ term appearing in (2.7). This is our G in §4.1.

Once we've computed our approximate values of $P(o|h)$, we substitute them into (2.7) to compute an approximate hypothesis posterior, and take the argmax. Thus we don't care that the actual values of $P(o|h)$ are accurate, only that the rank ordering of hypotheses under our approximate posterior is the same as the true rank ordering, or indeed just that the argmax of the approximation is the true argmax.

4.3 Variational Bayes

Consider the following lower bound:

$$\log P(o|h) = \log \sum_{\theta} \int P(o, \theta, w|h) dw \quad (4.1)$$

$$= \log \sum_{\theta} \int q(\theta, w) \frac{P(o, \theta, w|h)}{q(\theta, w)} dw \quad (4.2)$$

$$\geq \sum_{\theta} \int q(\theta, w) \log \frac{P(o, \theta, w|h)}{q(\theta, w)} dw \quad (4.3)$$

$$= \sum_{\theta} \int q(\theta, w) \log \frac{P(o|\theta, w)P(\theta|h)P(w)}{q(\theta, w)} dw \quad (4.4)$$

where $q(\theta, w)$ is any valid probability distribution. The inequality is just Jensen's inequality for $x \mapsto \log(x)$, and so we have equality iff

$$q(\theta, w) = P(\theta, w|o, h)$$

Since this lower bound holds for any distribution q , we could use this as our variational parameter (α in §4.1). However, for general q the integral is intractable (indeed, we wouldn't have any need for a variational approach otherwise). *Variational Bayes* restricts the allowed q to be of the form:

$$q(\theta, w) = q_{\theta}(\theta)q_w(w)$$

where our new variational parameters q_{θ} and q_w are valid distributions. How can we find the values of q_{θ} and q_w that give the tightest bound?

4.3.1 VBEM

Finding the q_{θ} and q_w which give the tightest bound is still intractable in general. However, we can use an EM-like procedure to alternately optimize q_{θ} keeping q_w fixed and vice versa. If $q_{\theta}^{(k)}$ and $q_w^{(k)}$ are our estimates of the distributions at iteration k , then from [3]:

$$\begin{aligned} q_{\theta}^{(k)}(\theta) &\propto \exp\left(\mathbb{E}_{q_w^{(k-1)}} \log P(o, \theta, w|h)\right) \\ &\propto \exp\left(\mathbb{E}_{q_w^{(k-1)}} \log P(o, \theta|w, h)\right) \end{aligned} \quad (4.5)$$

and

$$\begin{aligned} q_w^{(k)}(w) &\propto \exp\left(\mathbb{E}_{q_\theta^{(k)}} \log P(o, \theta, w|h)\right) \\ &\propto P(w) \exp\left(\mathbb{E}_{q_\theta^{(k)}} \log P(o, \theta|w, h)\right) \end{aligned} \quad (4.6)$$

It turns out that for linear mean-based adaptation with a conjugate transform prior, these functions can be decomposed in a way that enables us to do something like standard MLLR EM. Instead of re-estimating the transform, we re-estimate the hyperparameters of the transform distribution, and we do so using modified component occupancies $\gamma_m(t)$. Finally, the sum/integral in (4.4) can be expressed in a closed form in terms of the converged transform distribution hyperparameters. See [3] for full details.

4.4 Transform marginalized lower bound

In §3.5.6, we saw that since $P(o, w|\theta)$ is Gaussian in w , we can analytically marginalize over w to obtain $P(o|\theta)$. In this section, we consider how we might use this analytic result as the basis for an approximation scheme.

4.4.1 Simple transform marginalized lower bound

First we consider a simple variational lower bound:

$$P(o|h) = \sum_{\theta} P(o|\theta)P(\theta|h) \quad (4.7)$$

$$\geq P(o|\theta^*)P(\theta^*|h) \quad (4.8)$$

where our variational parameter θ^* is any fixed component sequence.

We use (3.13) to compute $P(o|\theta^*)$. Explicitly, we compute g , b and c for the fixed component sequence θ^* using (3.11), add the prior bias terms in (3.9) to obtain \bar{g} , \bar{b} and \bar{c} , and use these values in (3.13) to obtain $-\log P(o|\theta)$.

We compute $P(\theta|h)$ trivially as the product of all the transition probabilities in the state sequence corresponding to θ , and all the mixture weights necessary to go from this state sequence to the component sequence θ .

However, it is not clear how we might analytically optimize this bound. We therefore consider approaches to choosing θ^* that we heuristically think should do well. For instance, we could choose:

- the most likely component sequence (given the hypothesis h) from an SI system
- the most likely component sequence using a SAT system with the MAPLR transform

In fact, we are free to try a number of different values of θ , and see which one gives the tightest bound. This suggests using some procedure to explore the space of all component sequences, using the SI or SAT/MAPLR sequences above to guide our search, thus restricting the search space.

Greedy search

One possibility is to do a component-wise *greedy search*. At each iteration, we choose one component θ_t to update, and evaluate (4.8) for every possible value of this component, keeping the rest of the component sequence fixed. We then choose the component that gives the maximum of these computed values, and repeat. Since we are keeping the rest of the component sequence fixed, the rest of the state sequence is also fixed, and so the neighbouring state values conspire to restrict the current state s_t to one of at most two values. For each of these possible states, θ_t can be any component for that state, so in a 16-component system we get 32 possible values of θ_t at each iteration.

We start the greedy search at, for example, the SAT/MAPLR most likely sequence. Finally, to compute the lower bound we take the maximum value reached over all iterations. We would expect this to typically converge to a local optimum near the initial component sequence.

Rao Blackwellized Gibbs Sampling

Another possibility for exploring the space of component sequences is to use the component sequences generated in the course of *Rao Blackwellized Gibbs Sampling (RBGS)*.

As we saw in §3.5.6, we can analytically integrate over w to obtain $P(o|\theta)$, and furthermore that if we change θ at only one time θ_t we can efficiently compute the updated likelihood $P(o|\theta)$, as long as we keep track of certain values. Since we can easily compute $P(\theta|h)$, we can therefore efficiently compute the old and updated values of $P(o, \theta|h)$.

This enables us to do Gibbs sampling on the component sequence, since

$$\frac{P(\Theta_t = \theta_t | o, h, \theta_{\neq t})}{P(\Theta_t = \tilde{\theta}_t | o, h, \theta_{\neq t})} = \frac{P(o, \Theta = \theta | h)}{P(o, \Theta = \tilde{\theta} | h)}$$

For RBGS, then, we proceed as for greedy search, at each iteration choosing one component θ_t to update. But now instead of choosing the value that *maximizes* (4.8), we *sample* from the (normalized) distribution defined by this equation, considering it as a function of θ_t .

Initialization and final computation of the lower bound are as for greedy search. However, since RBGS is a form of Gibbs sampling (§2.6), it will eventually explore the whole search space, assuming non-pathological conditional distributions. It thus theoretically avoids becoming stuck in local optima as greedy search does. However, the time to explore a significant portion of the search space may be still be very large.

Problems with the simple approach

We might expect the lower bound in (4.8) to be pretty bad. We are approximating a sum by one term of that sum, and this will be poor unless one term dominates – that is, one component sequence dominates. Intuitively this seems unlikely. For any given state sequence, we have *very* many plausible component sequences, and our experience with standard Forwards-Backwards suggests that there are many plausible state sequences. Indeed, if it was often the case that one component sequence significantly dominated all

others, then there would be little advantage in doing EM over so-called *hard EM* or *Viterbi re-estimation*, where we use a single fixed component sequence rather than a distribution, when doing standard model parameter re-estimation.

In practice, we found that the bounds obtained for the simple transform marginalized lower bound were many, many orders of magnitude worse than the VB lower bounds. For this reason we decided not to investigate the RBGS or greedy search enhancements further. However, with a tweak we can in fact produce bounds competitive with VB.

4.4.2 Extended transform marginalized lower bound

For the simple transform marginalized case, we computed \bar{g} , \bar{b} and \bar{c} based on a *single* component sequence θ^* , and used these values in (3.13). A natural question is what happens if we instead use values of \bar{g} , \bar{b} and \bar{c} based on a *distribution* over component sequences $Q(\theta)$. In this case we are computing (3.12). Can we somehow use this new value to obtain a lower bound on $P(o|h)$?

Indeed we can:

$$\begin{aligned}
\log P(o, w|h) &= \log \sum_{\theta} P(o, w, \theta|h) & (4.9) \\
&= \log \sum_{\theta} Q(\theta) \frac{P(o, w, \theta|h)}{Q(\theta)} \\
&\geq \sum_{\theta} Q(\theta) \log \frac{P(o, w, \theta|h)}{Q(\theta)} \\
&= \sum_{\theta} Q(\theta) [\log P(o, w|\theta) + \log P(\theta|h) - \log Q(\theta)] \\
&= \mathbb{E}_Q [\log P(o, w|\theta)] - \text{KL}(Q(\theta)||P(\theta|h)) & (4.10)
\end{aligned}$$

where we've used Jensen's inequality for $x \mapsto \log(x)$ again, and so have equality iff $Q(\theta) = P(\theta|o, w, h)$, which can't generally be satisfied for all w . Therefore:

$$P(o, w|h) \geq e^{\mathbb{E}_Q [\log P(o, w|\theta)]} e^{-\text{KL}(Q(\theta)||P(\theta|h))}$$

so

$$\begin{aligned}
P(o|h) &= \int P(o, w|h) dw \\
&\geq e^{-\text{KL}(Q(\theta)||P(\theta|h))} \int e^{\mathbb{E}_Q [\log P(o, w|\theta)]} dw
\end{aligned}$$

and so

$$\log P(o|h) \geq \log \left(\int e^{\mathbb{E}_Q [\log P(o, w|\theta)]} dw \right) - \text{KL}(Q(\theta)||P(\theta|h)) \quad (4.11)$$

The first term is just minus left hand side of (3.12), and so we know how to compute it given \bar{g} , \bar{b} and \bar{c} .

Note that in the case that $Q(\theta) = \delta(\theta, \theta^*)$, the first term in (4.11) becomes $\log P(o|\theta^*)$, and the second term becomes $\log P(\theta^*|h)$. Thus this extended lower bound is a generalization of the simple lower bound above.

Note also that although we also used Jensen’s inequality in the derivation of (4.4), and so in the derivation of Variational Bayes, we’ve done something very different here. Firstly, in the VB case we considered variational distributions over θ and w , whereas now we consider variational distributions over θ only, having already marginalized over w . Furthermore, whereas for VB we use Jensen once and that’s the only inequality, here we’ve used Jensen for *every* value of w , then exponentiated both sides, then integrated. These are clearly very different operations. Nevertheless, our bound will turn out to be closely related to VB.

We now restrict to the case that $Q(\theta) = P(\theta|o, w^*, h)$ for some transform w^* . This is precisely the distribution computed when doing standard EM for model parameter re-estimation with a fixed transform w^* . We can use (3.9) to compute \bar{g} , \bar{b} and \bar{c} used in the first term of (4.11). How can we compute the KL divergence term?

In fact, for this special class of Q we have equality in (4.10) for $w = w^*$, and so we have:

$$\begin{aligned} -\text{KL}(Q(\theta)||P(\theta|h)) &= \log P(o, w^*|h) - \mathbb{E}_Q[\log P(o, w^*|\theta)] \\ &= \log P(o|w^*, h) - \mathbb{E}_Q[\log P(o|w^*, \theta)] \end{aligned}$$

where $\log P(o|w^*, h)$ is just the log likelihood of the utterance as computed by $\beta_{\text{initial}}(0)$ when doing Forwards-Backwards.

Thus our bound (4.11) becomes:

$$\log P(o|h) \geq \log P(o|w^*, h) + \log \left(\int e^{\mathbb{E}_Q[\log P(o, w|\theta)]} dw \right) - \mathbb{E}_Q[\log P(o|w^*, \theta)] \quad (4.12)$$

We have already noted that (3.12) expresses the second term in terms of \bar{g} , \bar{b} , \bar{c} , and looking at (3.6) we see it expresses the third term in terms of g , b and c . Combining these two explicitly, and using (3.10), we get:

$$\begin{aligned} \log \left(\int e^{\mathbb{E}_Q[\log P(o, w|\theta)]} dw \right) - \mathbb{E}_Q[\log P(o|w^*, \theta)] &= \\ \log P(w^*) & \\ + \frac{1}{2} \sum_{ikl} \bar{g}^{ikl} (w_*^{ik} - \bar{\mu}^{ik}) (w_*^{il} - \bar{\mu}^{il}) & \quad (4.13) \\ + \frac{1}{2} \left(n(n+1) \log 2\pi - \sum_i \log |\bar{g}^i| \right) & \end{aligned}$$

where we’ve written w^* as w_* to make the superscripts clearer.

Note that this extended bound is a variational lower bound, since the lower bound (4.12) holds for any choice of Q such that $Q(\theta) = P(\theta|o, w^*, h)$. Our variational parameter is therefore w^* . As in the case of the simple bound, it is not clear how to directly optimize w^* . We take the approach of choosing w^* to be the MAPLR transform.

Computing the extended bound

The procedure for computing our bound (4.12) is therefore to compute the sum of the second and third terms using (4.13), then add this to the likelihood $\log P(o|w^*, h)$ computed by Forwards-Backwards.

This computation can in fact be easily embedded in the EM procedure for MAPLR (§3.3). When doing MAPLR, we already compute $\log P(o|w^*, h)$ in the course of doing Forwards-Backwards, and we already compute \bar{g} , \bar{b} and \bar{c} for the transform re-estimation step. Thus all that is required to compute our bound is to add a bit of code that computes (4.13) as we are doing transform re-estimation. In fact, we are even already computing $\bar{\mu}^i$, since MAPLR chooses w as the argmax of (3.8), which is precisely the mean of the Gaussian distribution, $\bar{\mu}^i$.

4.4.3 Relationship of transform marginalized bounds to VB

We now consider the relationship of both the simple and extended transform marginalized bounds to explicitly variational approaches like VB. Indeed, since the simple bound is a special case of the extended one, it suffices to only consider the extended bound.

An interesting question is whether the extended transform marginalized bound, which had a somewhat complicated derivation, can be more directly expressed as a variational lower bound of the form (4.4) for some appropriate choice of $q(\theta, w)$. It turns out that it can. Set

$$\begin{aligned} q_w(w) &\propto e^{\mathbb{E}_Q[\log P(o, \theta, w|h)]} \\ q_\theta(\theta) &= Q(\theta) \\ q(\theta, w) &= q_\theta(\theta)q_w(w) \end{aligned}$$

Then substituting this Q into (4.4) and simplifying, it can be shown that:

$$\begin{aligned} \log P(o|h) &\geq \sum_{\theta} \int q(\theta, w) \log \frac{P(o|\theta, w)P(\theta|h)P(w)}{q(\theta, w)} dw \\ &= \log \left(\int e^{\mathbb{E}_Q[\log P(o, w|\theta)]} dw \right) - \text{KL}(Q(\theta)||P(\theta|h)) \end{aligned}$$

which is precisely our lower bound (4.11).

In fact, since the form of variational distribution $q(\theta, w)$ was separable as it is for VB, the transform marginalized lower bound is a *special case* of VB – VB allows q_θ and q_w to be arbitrary distributions.

The connection with VB is even closer. The astute reader may have noticed that our choice of q_w above is, as it happens, exactly that used in (4.6) to do VBEM. Thus our bound is precisely the bound you would get using VBEM, if you started with q_θ set to Q , performed one step of VBEM transform re-estimation using (4.6), and used this to compute the VB bound.

Therefore our method can be seen as performing a version of VBEM, where instead of *optimizing* q_θ , we just set $q_\theta(\theta) = P(\theta|o, w^*, h)$, where w^* is the MAPLR transform.

The advantages of our method over VBEM, then, are that it is very easy to implement once we can do MAPLR, and requires less storage, since we only have to store a transform, not the hyperparameters for a *distribution* over transforms as at each step of VBEM. However, VBEM should always result in tighter bounds.

It is perhaps surprising that the extended transform marginalized lower bound and VB turn out to be so closely related, given the differences in their respective derivations.

Chapter 5

Sampling methods

5.1 Overview

To recap, our aim is, for each utterance, to estimate the most likely hypothesis given the observations:

$$\arg \max_h P(h|o)$$

We do this by building up an approximation of the true hypothesis posterior $P(h|o)$, relying on the approximation being good enough that the most likely hypothesis under the approximation will often coincide with the most likely hypothesis under the true distribution.

We will show below how to use sampling methods to generate samples from the overall joint posterior $P(w, \theta, h|o)$, but first we'll discuss how to use these samples to build up our approximation of $P(h|o)$. In general, taking a sample from a joint distribution $P(x, y)$ and throwing away one variable, y , is easily seen to be equivalent to taking a sample from the marginalized distribution $P(x)$. Thus sampling from the overall joint posterior automatically gives us samples from $P(w|o)$ and $P(h|o)$. How might we use these?

5.2 How to use samples

We could build up an approximation to the distribution $P(h|o)$ by sampling from it repeatedly, and counting the frequencies of the different hypotheses. We are effectively building up a histogram. Intuitively this leads to quite a blocky distribution at low sample counts.

An alternative is to use the following expression:

$$P(h|o) = \int P(h|w, o)P(w|o)dw$$

By taking a series of independent samples $(w_i)_1^n$ from $P(w|o)$, we can produce a Monte Carlo approximation

$$P(h|o) \approx \frac{1}{n} \sum_i P(h|w_i, o)$$

to this integral. It turns out that the required value $P(h|w_i, o)$ is easily computable by performing the Forwards-Backwards algorithm, and in fact is already computed for our

sampling procedure below. This process can be seen as adding a *distribution* for each sample, whereas before we were adding a discrete delta function for each sample. We therefore expect it to converge faster (in fewer samples), and this is the form used in our experiments.

Note that in both cases the accuracy of the approximation only matters to the extent that its *most likely hypothesis* is often the true one. Thus we might expect that even approximations quite far from the true distribution would still be useful.

5.3 High-level Gibbs sampling

In this section we show how to draw samples from $P(w, \theta, h|o)$. To do this, we will use a high-level Gibbs sampling procedure. See §2.6 for a brief introduction to Gibbs sampling.

We will treat the hypothesis H and the component sequence within that hypothesis Θ as one random variable (H, Θ) , and the transform W as the other. This view of H and Θ as a pair is in fact very natural, if we consider its value as picking a single path through the set of all possible component sequences for any possible word sequences (indeed, this is the network on which we do Viterbi). We call our procedure a *high-level* Gibbs procedure since we are sampling big objects like the matrix W , not its components.

For our method, then, we need to be able to sample from the conditionals $P(h, \theta|o, w)$ and $P(w|o, \theta, h)$. Since

$$P(h, \theta|o, w) = P(h|o, w)P(\theta|h, o, w)$$

we can sample from $P(h, \theta|o, w)$ by sampling from $P(h|o, w)$, then using this sample as our value h and sampling from $P(\theta|h, o, w)$.

Thus our scheme for generating a sample from the overall joint distribution is a 3-way sampling procedure:

1. sample the hypothesis from $P(h|o, w)$
2. sample the component sequence from $P(\theta|h, o, w)$
3. sample the transform from $P(w|o, \theta)$
4. (repeat)

In the following sections we show how to sample efficiently from each of these distributions.

5.3.1 Sampling the hypothesis

We want to sample from

$$P(h|o, w) \propto P(o|w, h)P(h)$$

For any h , we can compute $P(o|w, h)$ using the Forwards-Backwards algorithm – it is just the overall ‘utterance likelihood’ $\beta_{\text{initial}}(0)$, where w is incorporated as a transform of the model parameters (an input transform in HTK). The term $P(h)$ is just the probability assigned to the hypothesis by the language model.

Thus we can compute the above expression for all h , normalize to obtain a valid distribution, and sample from this naively – that is, picking a random number from the uniform

distribution on $[0, 1]$, and taking the first hypothesis such that the cumulative distribution at that hypothesis exceeds our uniformly chosen value.

5.3.2 Sampling the component sequence

We want to be able to sample the component sequence Θ from the distribution $P(\theta|o, w, h)$. First, let's show how to sample the state sequence S from the distribution $P(s|o, w, h)$.

Given a transform w and a hypothesis h^1 , we can compute backwards probabilities $\beta_q(t)$ as discussed in §2.2.2:

$$\begin{aligned}\beta_q(t) &\triangleq P(o_{t+1:T}|S_t = q) \\ &= \sum_r a_{qr} b_r(o_{t+1}) \beta_r(t+1)\end{aligned}$$

Manipulating the above expression, it is easy to show that

$$\begin{aligned}\hat{a}_{qr}(t) &\triangleq P(S_{t+1} = r | S_t = q, o) \\ &= \frac{a_{qr} b_r(o_{t+1}) \beta_r(t+1)}{\beta_q(t)}\end{aligned}$$

where we condition on the *entire* observation sequence o . Furthermore, we can see by a number of methods² that

$$P(s_{t+1}|s_{1:t}, o) = P(s_{t+1}|s_t, o) \quad (5.1)$$

i.e. that the left hand side is in fact independent of $s_{1:(t-1)}$. This means that, conditional on the entire observation sequence o , the state sequence s is a (time-dependent) Markov chain, with transitions probabilities given by $\hat{a}_{qr}(t)$. In particular, we can sample from the entire distribution $P(s|o)$ just by starting at the value `initial` at time 0 and sampling in sequence: first sample s_1 given $S_0 = \text{initial}$, then s_2 given this s_1 , then s_3 given s_2 , etc., at each stage sampling s_{t+1} from the distribution $\hat{a}_{s_t s_{t+1}}(t)$.

Given q , a_{qr} and so $\hat{a}_{qr}(t)$ is typically non-zero for only a few values of r (at most 2 for standard HMMs). Thus sampling at each t is reasonably efficient, and we only need to repeat this T times to get a sample for the entire state sequence.

In particular, note that we are doing a very similar computation at each time step to that which originally computed the backwards probabilities. But for each t we only need to do this computation at one state, whereas the backwards algorithm must compute values at every (active) state. Thus we can sample using the backwards probabilities for a fraction what it cost to compute them.

Given this state sequence sample we can easily sample from the component sequence, since

$$P(\theta_t | s_t, o_t, w) = \frac{P(o_t | \theta_t, w) P(\theta_t | s_t)}{P(o_t | s_t, w)}$$

¹Just for the remainder of this argument, we implicitly condition all probabilities on w and h .

²A direct one is to note that:

$$P(s_t | s_{1:(t-1)}, o) = P(s_t | o_{t:T}, s_{1:(t-1)}, o_{1:(t-1)}) = \frac{P(s_t, o_{t:T} | s_{1:(t-1)}, o_{1:(t-1)})}{P(o_{t:T} | s_{1:(t-1)}, o_{1:(t-1)})}$$

where looking at the graphical model in Figure 2.2 we can easily see that both the numerator and denominator are independent of $s_{1:(t-2)}$, depending as they do only on s_{t-1} .

and θ_t is independent of everything else (including θ at other t), since we conditioned on its Markov blanket (§2.1).

We note explicitly that this method can be extended in an intuitive way to the case where we have non-emitting nodes. Indeed, the same considerations apply as to the extension of the normal Backwards algorithm to the non-emitting case.

5.3.3 Sampling the transform

The distribution we want to sample from, $P(w|o, \theta, h)$, is Gaussian in the components of w (§3.5.5). In §A we show how to sample from a Gaussian given its mean and the Cholesky decomposition of its covariance matrix. Thus we can sample from our required distribution.

It is worth commenting on the relationship with the transform update step when doing EM for MLLR and MAPLR. We compute the same mean when doing MAPLR, and computing this mean involves inverting the covariance matrix. Depending on how this inversion is done, it may be possible to derive the Cholesky decomposition as a byproduct – for example, in HTK the inverse is computed by Singular Value Decomposition, and is trivial to go from this to the Cholesky decomposition. MLLR is very similar computationally to MAPLR, differing only by a bias term on the accumulators (§3.5.3). So computing the parameters we need to sample from our distribution is in fact very closely related to that needed for the EM step in MLLR or MAPLR.

5.4 Acoustic deweighting

The assumption in HMMs that the output vectors at successive times are independent, conditioned on the state sequence, is known to be a particularly bad approximation, and leads (in part) to the dynamic range of the acoustic model likelihood being vastly greater than the dynamic range of the language model probability. Thus the acoustic score dominates unreasonably. In practice a hack is customarily used to normalize the dynamic ranges – either we adjust the language model probabilities by raising each to the power of a *language model scale factor*, or we adjust the acoustic model score via some form of *acoustic deweighting*.

In general, we deweight probabilities by raising them to a power (less than 1). This is equivalent to multiplying the log probabilities by a constant. Note that this usually results in a distribution which is no longer normalized, so we must remember to take the normalizing constant into account if we ever want to compute actual probabilities.

5.4.1 Forms of acoustic deweighting

In the case of acoustic deweighting, we can apply deweighting at several different levels: we could deweight individual components, individual states, the acoustic model given a particular transform, or even the entire acoustic model.

Component level deweighting

Here we raise $P(o_t|\theta_t, w)$ to a power α for each t , which is the same as raising $P(o|\theta, w)$ to the power α . So whereas before we had the joint distribution

$$P(o, w, \theta, h) = P(o|w, \theta)P(\theta|h)P(w)P(h) \quad (5.2)$$

we now have the (unnormalized) joint distribution:

$$\tilde{P}(o, w, \theta, h) \propto P(o|w, \theta)^\alpha P(\theta|h)P(w)P(h) \quad (5.3)$$

We could also write this mentioning the state sequence s explicitly. Now our original joint distribution

$$P(o, w, \theta, s, h) = P(o|w, \theta)P(\theta|s)P(s|h)P(w)P(h) \quad (5.4)$$

goes to:

$$\tilde{P}(o, w, \theta, s, h) \propto P(o|w, \theta)^\alpha P(\theta|s)P(s|h)P(w)P(h) \quad (5.5)$$

Note that we leave $P(\theta|h) = P(\theta|s)P(s|h)$ intact, since we believe the mixture weights and state transition probabilities are reasonable since they're not part of the above faulty conditional independence assumption.

State level deweighting

Here we raise $P(o_t|s_t, w)$ to a power α for each t , which is the same as raising $P(o|s, w)$ to the power α . Our original joint distribution

$$P(o, w, s, h) = P(o|w, s)P(s|h)P(w)P(h) \quad (5.6)$$

goes to:

$$\tilde{P}(o, w, s, h) \propto P(o|w, s)^\alpha P(s|h)P(w)P(h) \quad (5.7)$$

Similarly, we leave $P(s|h)$ intact here.

Deweighting the acoustic model given a particular transform

Here we raise $P(o|w, h)$ to a power. Our original joint distribution

$$P(o, w, h) = P(o|w, h)P(w)P(h) \quad (5.8)$$

goes to:

$$\tilde{P}(o, w, h) \propto P(o|w, h)^\alpha P(w)P(h) \quad (5.9)$$

Deweighting the entire acoustic model

Here we raise $P(o|h)$ to a power. Our original joint distribution

$$P(o, h) = P(o|h)P(h) \quad (5.10)$$

goes to:

$$\tilde{P}(o, h) \propto P(o|h)^\alpha P(h) \quad (5.11)$$

5.4.2 Discussion

The above forms differ in the level at which we apply acoustic deweighting. For state level deweighting, we first marginalize (5.4) over the component sequence θ , then apply deweighting. For deweighting the acoustic model given a particular transform, we first marginalize (5.6) over the state sequence s (or equivalently marginalize (5.2) over the component sequence θ), then apply deweighting. For deweighting the entire acoustic model, we first marginalize (5.8) over the transform w , then apply deweighting.

Note that deweighting at higher levels ends up in some way deweighting things we believe are fine, for example the transition probabilities in the case of deweighting the acoustic model given a particular transform, or the transform prior in the case of deweighting the entire acoustic model. In this sense, perhaps the conceptually nicest form is *state level deweighting*, since we deweight precisely the distribution we don't believe in – namely $P(o|w, s)$ ³.

Even if we made the different forms of deweighting more similar by taking the above into account (for example, deweighting the transition probabilities when doing state level deweighting to match the fact this is implicitly done for higher level deweighting), the different forms of deweighting are not equivalent. Marginalizing and then raising to a power bears no simple relationship to raising to the power and then marginalizing, since in general $(\sum_i a_i)^\alpha$ bears no simple relationship to $\sum_i a_i^\alpha$.

As an aside, we note that a true Bayesian approach would be to estimate the acoustic deweighting factor as a random variable. We would have a prior over α , and the sum total of all data would give us evidence about the true value. However, this really counts as training not recognition, and in any case is beyond the scope of this project.

5.4.3 Deweighting for lower bound and sampling methods

For VB and other lower bound procedures, it makes no difference whether we use a language model scale factor or do acoustic model level deweighting, since the rank ordering is the same in both cases, and the estimation of $P(o|h)$ occurs identically in both cases. However, using a language model scale factor instead of acoustic model level deweighting makes the distribution $\tilde{P}(o, h)$ more extreme, with its dynamic range greatly increased, and its energy scaled by a large factor (e.g. 15). Intuitively, we might expect this to exacerbate problems with sampling procedures only exploring a certain portion of parameter space⁴, for example with Gibbs sampling taking a very long time to explore groups of peaks not aligned with any co-ordinate axis (§2.6). Thus acoustic deweighting is more important for our high-level Gibbs sampling method than for lower bound methods.

Unfortunately, it seems hard to include higher level deweighting into a valid probabilistic framework that we can still sample from using our high-level Gibbs sampling procedure. The problem is that it becomes hard to define helpful joint distributions over the parts of

³The factoring $P(o|w, \theta) = \prod_t P(o_t|w, \theta_t)$ expressing the conditional independence of the outputs at different times given the components is actually fine, since this just an inherent part of using GMMs to model state output distributions, which is *not* part of the above faulty conditional independence assumption.

⁴Slightly more explicitly, using a language model scale factor rather than acoustic model level deweighting is equivalent to considering $P(o, h)$ at a lower temperature. In fact, we want to find the argmax in h of this function, which is equivalent to sampling at 0 temperature. However, the success of simulated annealing as a method shows us that it is advantageous not to only explore the low temperature landscape, since typical sampling-like methods in some sense get stuck in very high local peaks.

the model we marginalize over before applying deweighting, e.g. the component sequence in state level deweighting, and the state and component sequences in acoustic model level deweighting. This is not to say it can't be done, and this will be an interesting avenue to pursue. However, for this project we restrict ourselves to component level deweighting.

With component level deweighting it turns out that the techniques we've developed so far for high-level Gibbs sampling are still valid. That is, our procedure samples from the joint distribution $\tilde{P}(o, \theta, w, h)$ in (5.3). The only changes necessary are to use the modified $P(o_t | \theta_t, w)$ values when doing Forwards-Backwards, and to multiply the value of the accumulators g , b and c by α .

We mention here that high levels of component level deweighting have a deleterious effect on computation time, since the dynamic range in tokens is reduced, and so a given pruning beam is less effective. We can legitimately reduce the pruning threshold to combat this, by setting it so that we have roughly the same number of active tokens as before.

Chapter 6

Experiments

6.1 Experimental set-up

We chose to use the experimental set-up from [3], which was kindly made available to us, as the basis for our experiments. We recapitulate the important details here.

The domain is conversational telephone speech. The ML-trained SAT (Speaker Adaptive Training) models are trained on about 295 hours of data from the Call Home English, Switchboard and Switchboard-Cellular corpora. The evaluation data set was taken from the NIST RT-03 Spring Evaluation, consisting of 144 speakers (77 female, 67 male) totalling about 6 hours of speech. The front-end used is as follows: the speech is converted to 12-dimensional PLP with log energy, and first, second and third derivatives; cepstral mean and variance normalization and VTLN are then applied; finally, a HLDA-estimated transform is used to reduce the dimensionality from 52 to 39. A decision tree was used to produce a state-clustered triphone model set with an average of 16 Gaussian components per state. This is the speaker independent (SI) model. From here the ML-SAT system was built using MLLR adaptive training, with separate speech and silence transforms.

Transform priors were then estimated separately for the speech and silence transforms, using empirical Bayes as discussed in §3.4.2.

The evaluation set consists of 7074 utterances with an average utterance length of 3.13 seconds. The SI model was then used to construct the N -best lists used for the experiments.

We use the SAT system for the experiments with Bayesian inference since it is slightly closer to a Bayesian approach to training than the SI system. Traditional complexity control schemes such as regression class trees are used to ensure sufficient data for robust transform estimation when doing adaptive training. These also mean that there is sufficient data for each transform that a Bayesian approach, incorporating a prior, might tend to have a posterior distribution heavily peaked at the ML estimated transform, and so the ML transform is a reasonable approximation. Indeed, performance gains are seen in [3] using the SAT system instead of the SI system for Variational Bayes.

The HMM Toolkit (HTK) was used for all experiments, and my new methods were implemented in HTK on top the standard code.

6.2 Reduced- N experiments

As stated in the introduction, this project is primarily concerned with what is possible, not with computational efficiency. It would be nice to be able to run our Gibbs sampling procedure for long enough that we don't have to worry too much about whether or not it has converged, for instance. We therefore look at how VB's performance degrades when using a reduced N -best list. The original VB approach in [3] uses $N = 150$. We try 5, 10 and 15.

6.3 Transform marginalized lower bound

We care about two things when comparing lower bounds: what the difference in tightness is between the two bounds, and what effect this has on rank ordering, and so WER.

For an individual hypothesis, it is easy to compare two lower bounds – the higher value is better. However, we want a way to see how the two bounds compare over many hypotheses. In order to get a handle on this, we decided to do the following. Firstly, we use 'log ratio' as our measure of choice for comparing the two bound, i.e. $\log(\frac{a}{b}) = \log(a) - \log(b)$, where a is the bound one method gives and b is the bound the other methods gives. This is positive or negative depending on which method gives the tighter bound. This is a sensible metric since the bounds are derived and expressed in terms of logs, and it is not uncommon for them to differ by very many orders of magnitude in linear terms. We then take the log ratio values for all hypotheses, and do a cumulative plot showing that, say, 40% of our hypotheses had a log ratio less than +200. We compare our extended transform marginalized bound to VB using such a cumulative plot.

For WER, we combine the lower bound on the acoustic model score $P(o|h)$ and the language model score $P(h)$ to get our estimate of the posterior $P(h|o)$. So that our experiments are directly comparable to the ones for VB in [3], we used the same value of 15.0 for the language model scale factor (it makes no difference to the rank ordering whether we use a language model scale factor or do acoustic model-level deweighting).

6.4 High-level Gibbs sampling

6.4.1 Acoustic deweighting

The first task in actually running our high-level Gibbs sampling is to select an appropriate component-level acoustic deweighting value. This is a delicate parameter, as too much acoustic deweighting means the acoustic model has no influence, and too little means the language model has no influence. To give us some rough idea, note that a typical language model scale factor is about 15.0, and this might suggest trying an acoustic deweighting factor of $\frac{1}{15}$.

To get a better idea, we decided to do something more involved. As we saw in §5.4, acoustic deweighting is meant to be compensating for false conditional independence assumptions in the HMMs. In other words, we don't believe the model when, without acoustic deweighting, it tells us that the one hypothesis is always orders of magnitude more likely than any of the others. One way to get a handle on acoustic deweighting, then, is to look at what happens

to the posterior $P(h|o)$ as we vary the deweighting parameter. The problem is, we want to select a value that will be sensible for all the utterances. How do we get a handle on what the acoustic deweighting value does to the posterior distributions for a *set* of utterances?

We decided to give each posterior distribution a score according to how skewed to one hypothesis it was. The measure we adopted was *entropy*. At the extremes, a completely flat distribution has maximum entropy (around 1.61 nats or 2.32 bits for 5 hypotheses), and a completely deterministic distribution has zero entropy. If we then measure the posterior's entropy for a sample of utterances, and plot a histogram showing how often the entropy lies in a given range, we can build up an idea of what the acoustic deweighting is doing to all utterances.

It is important to get a feel for how the entropy value compares with our internal notion of skewness. To this end, we take a couple of random samples from each histogram bin and display their posteriors.

On a practical note, the posterior $P(h|o)$ is estimated in exactly the same way as for our high-level Gibbs sampling. Because we are using a small set of utterances (we chose 40), we can afford to sample a lot, and so can be reasonably sure we have a good approximation.

6.4.2 High-level Gibbs sampling

For our high-level Gibbs experiment we estimate the posterior $P(h|o)$ for each utterance by taking a series of samples from the overall joint posterior, as described in §5.2.

As discussed in §2.6, we have a trade-off between the length of time we devote to an individual 'run' and the number of runs. For our high-level Gibbs sampling procedure, we chose to do 5 runs, 100 iterations total of Gibbs sampling, with the first 54 iterations as burn-in, and samples taken every 15 iterations, to give a total of 20 samples for each utterance.

We initialize by sampling from the transform prior.

Chapter 7

Results

7.1 Reduced- N experiments

The results of using a reduced size of N -best list for VB are given in Table 7.1, along with the SI system results for comparison. We see that we don't lose much in WER, and retain most of the increase of VB over the SI system, even for a 5-best list. Thus we know gains are still possible even at $N = 5$, and so it is sensible to test our new methods at this value.

We therefore adopt $N = 5$ for the rest of the experiments.

In order to be able to compare our new methods to existing methods, we summarize the results from [3] for MLLR, MAPLR and VB on a 150-best list in Table 7.2.

N	WER (%)
150	31.5
15	31.7
10	31.8
5	31.9
(SI)	(32.8)

Table 7.1: Performance degradation with reduced N -best list size for VB

method	WER (%)
SI	32.8
MLLR	35.2
MAPLR	31.8
VB	31.5

Table 7.2: Performance of existing schemes (150-best list)

7.2 Transform marginalized lower bound

Preliminary results for the simple transform marginalized bound showed bounds so loose that we decided to devote our time to the extended transform marginalized bound instead. The simple bound had a log ratio of around 12,000 compared to the extended one – a long way off the chart in Figure 7.1! As discussed in §4.4.1, we might have expected this to be the case.

The results of the cumulative log ratio plot (§6.3) for our extended transform marginalized bound compared to VB are shown in Figure 7.1. We plot curves for our bound after varying numbers of iterations of MAPLR. We may call the value at iteration 4 ‘the MAP estimate’. We see that even at what is nominally iteration 0 (the transform prior), our bound is still not wildly bad. After even one iteration, it is uniformly better than VB. However, it is misleading to suggest our method is better than VB just because it has a uniformly tighter bound in the graph – the results for VB are after only 1 iteration of VBEM, and we cannot directly compare the way we count iterations for MAPLR and VBEM. We are confident that VB will still always beat our method after more iterations, as discussed in §4.4.3. The graph does show, however, that our method, which is simpler to implement and requires less storage than VB, does not perform much worse (as a lower bound) than VB after a comparable number of iterations – indeed, here it performs uniformly better.

Might we expect the increased tightening we get above to be converted into an improvement in WER? In fact, we don’t. In [3], performing more than one iteration of VBEM was not found to improve WER. However, since we believe that VBEM will beat our bound after a comparable number of iterations, we conjecture that performing additional iterations of VBEM does tighten the *bound* significantly, roughly of the same order as the tightening for our bound in Figure 7.1. This being the case, we expect the extra tightening we get for iteration 4 of our method over 1 iteration of VBEM *won’t* convert into an improvement in WER. We can see in Table 7.3 that we don’t get any significant improvement in WER.

Given that the tightening of the bounds did not lead to a significant improvement in WER, we might ask whether it has any effect on rank ordering – does tightening the bounds just tend to tighten the bounds for all 5 hypotheses by similar amounts, thus leaving their rank ordering the same? To investigate this we looked at the number of utterances for which two methods *didn’t* select the same best hypothesis. The results are shown in Table 7.4. We can see that the extra tightening certainly does make a difference to best hypothesis selection – over a fifth of the utterances have a different best hypothesis between our bound at iteration 4 and VB. Thus it is not the case that tightening has no effect on rank ordering.

7.3 High-level Gibbs sampling

7.3.1 Acoustic deweighting

The first task was to select the component-level acoustic deweighting value. Our entropy histogram, as discussed in §6.4.1, is shown in Figure 7.2. Example posteriors chosen at random from each bin are shown in Figure 7.3, as an aid to our intuition, as discussed in §6.4.1.

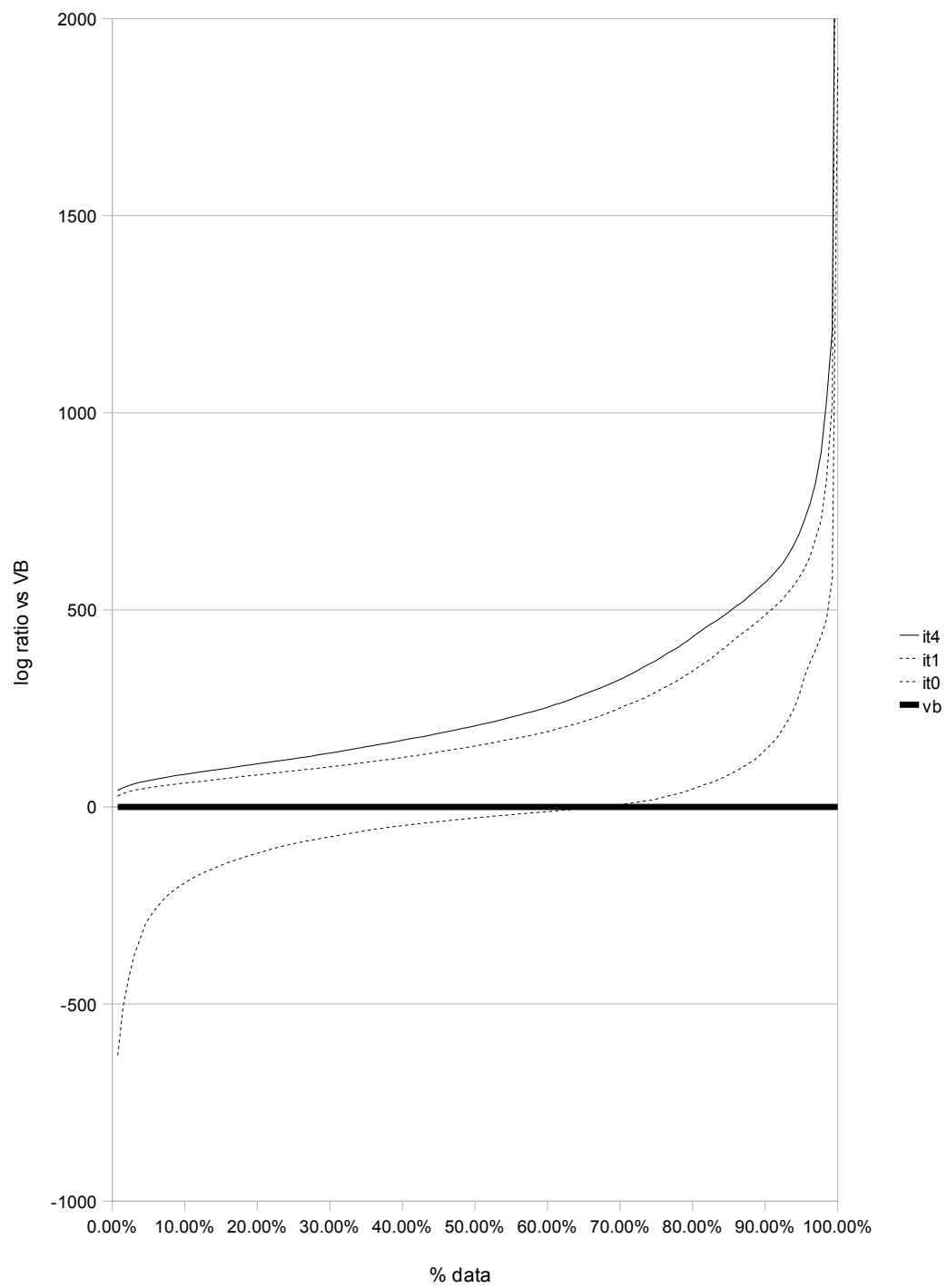


Figure 7.1: Cumulative plot comparing our transform marginalized extended bound to VB, after varying numbers of MAPLR iterations (VB 1 iteration only)

method	WER (%)
VB	31.9
ETMB (it1)	31.8
ETMB (it4)	31.8

Table 7.3: Performance of extended transform marginalized bound (ETMB) compared to VB (1 iteration only). 5-best lists were used in all cases.

methods	disagreement
VB and ETMB (it1)	17.4%
VB and ETMB (it4)	20.5%
ETMB (it1) and ETMB (it4)	9.3%

Table 7.4: Proportion of utterances for which two methods select different best hypotheses

On the basis of these results, we selected a value of 0.01 for the component-level acoustic deweighting. This can be seen to be in nice balance between the zero and full deweighting extremes, with plenty of skewed hypotheses and also plenty of flat ones.

7.3.2 High-level Gibbs sampling

The results of our high-level Gibbs sampling experiment are shown in Table 7.5. As we can see, they are somewhat disappointing. The performance is quite a bit worse than VB – even than the SI model. Two things jump out from the table.

Firstly, the extended transform marginalized bound, which was previously competitive with VB, does equally as badly as our high-level Gibbs method for this method of acoustic deweighting. Thus an immediate implication is that we have chosen a bad value of acoustic deweighting, in spite of our pains to ensure this didn’t happen. It is also possible that the *method* of deweighting is a bad one – the ETMB and VB bounds above effectively use acoustic model-level deweighting (actually they use a language model scale factor but the effect on rank ordering is the same), whereas our high-level Gibbs sampling procedure was forced to use component-level deweighting as discussed in §5.4.3. This adds impetus to the search for a way to do acoustic model-level deweighting within a strictly probabilistic framework. Further evidence that this choice of acoustic deweighting is a bad one comes from some preliminary high-level Gibbs sampling experiments which incorporated an attempt at deweighting the acoustic model given a particular transform, before this was found to be invalid. The WER for a single sample taken after 5 iterations is 33.4%, which is better than anything above.

The second thing immediately noticeable from the results is that our estimate of the posterior from even a single sample – indeed even a single sample at only the fifth iteration – is as good in terms of WER as averaging many samples. Our iterations thus appear to be doing very little.

We investigated this further using the same technique we used above for the lower bound experiments, namely computing the proportion of utterances on which the two models disagree about the best hypothesis. We find that our ‘lead’ case (55 burn-in, 15 gap, 20

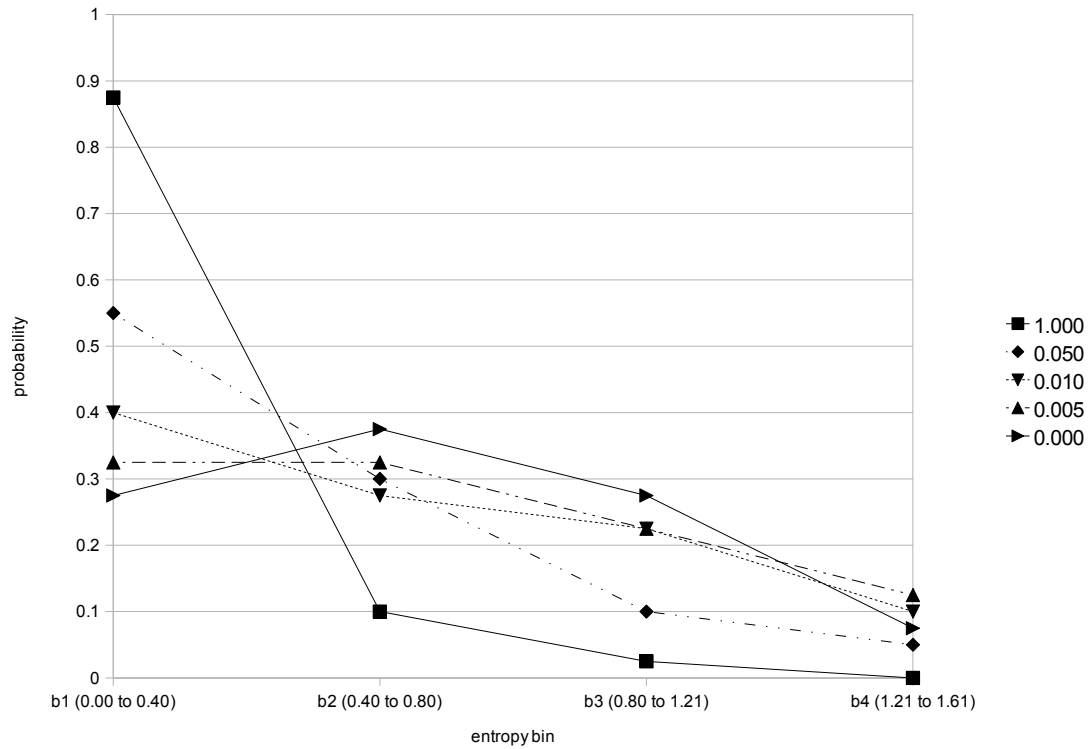
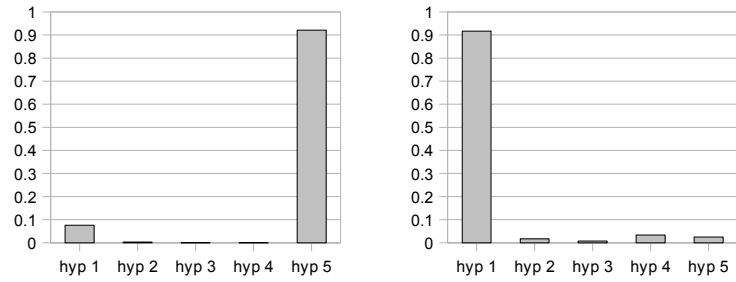


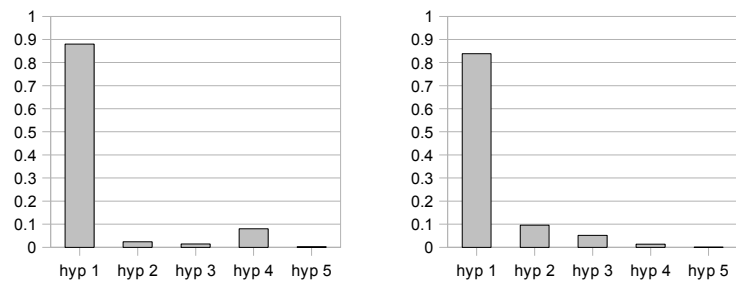
Figure 7.2: Entropy distribution

samples) differs from the single sample fifth iteration case at only 5.3% of utterances. And our lead case differs from ETMB, which is a decent indication of what the results would be for VB, at only 2.4% of utterances.

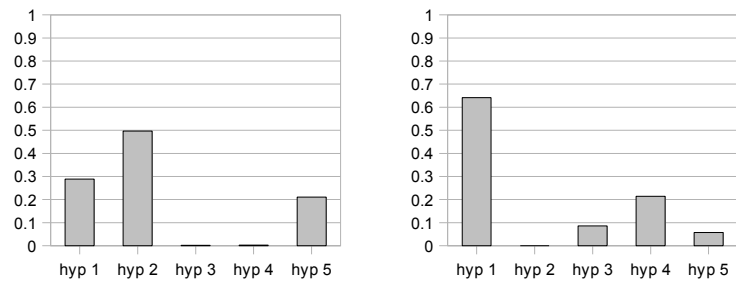
We may draw the following conclusions from this discussion: component-level acoustic deweighting with a value of 0.01 is not optimal; at this value of deweighting, high-level Gibbs sampling almost certainly converges *very* quickly to the true distribution; and at this value of deweighting, ETMB (and so, probably, VB) is a good enough approximation that the best hypotheses are very similar to the true (given the deweighting factor) best hypothesis. Unfortunately this doesn't tell us very much about how VB performs compared to the true distribution for other values or forms of acoustic deweighting.



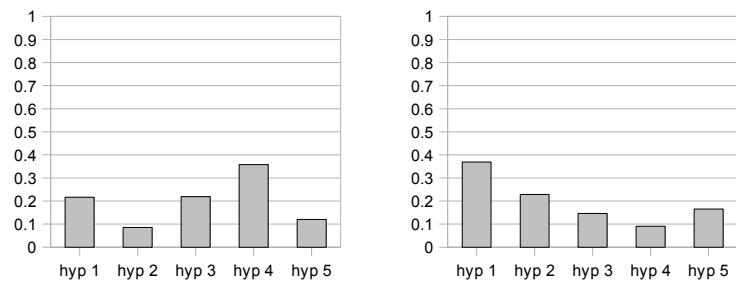
(a) bin 1 (0.0 to 0.40 nats)



(b) bin 2 (0.40 to 0.80 nats)



(c) bin 3 (0.80 to 1.21 nats)



(d) bin 4 (1.21 to 1.61 nats)

Figure 7.3: Random examples of posterior distributions $P(h|o)$ for entropy bins

parameters			WER (%)
burn-in	gap	samples	
55	15	20	34.1
55	5	50	34.2
100	-	1	34.2
5	-	1	34.2
ETMB (it4)			34.2

Table 7.5: High-level Gibbs sampling results, with extended transform marginalized bound (ETMB) shown for comparison ('gap' is the number of iterations left between successive samples after burn-in).

Chapter 8

Conclusion

We've presented new methods to do Bayesian adaptation within a linear mean-based framework.

The *extended transform marginalized bound* is a viable alternative to Variational Bayes for the scenario we consider. It is easier to implement, and requires less disk storage for the way adaptation is commonly implemented (e.g. in HTK).

The *high-level* Gibbs sampling results were inconclusive due to issues with acoustic deweighting. Further work needs to be done to establish to what extent results can be improved by choosing a judicious value for component-level deweighting, and whether acoustic model-level deweighting can be incorporated in a coherent way into the high-level Gibbs sampling procedure. Results from the case with suboptimal deweighting suggest, however, that Variational Bayes and our extended transform marginalized bound may be very good approximations to the true distribution, as far as the metric of WER is concerned.

Possible future directions would involve using single-component Gibbs sampling, possibly with overrelaxation, trying to somehow incorporate Rao Blackwellized Gibbs Sampling into our high-level Gibbs sampling procedure, and using more general MCMC methods such as Hamiltonian Monte Carlo – in particular, this would allow arbitrary transform priors.

Appendix A

Sampling from a Gaussian distribution

We want to sample from the multivariate Gaussian distribution

$$\mathcal{N}(\mu, \Sigma)$$

where μ is the mean and Σ is the covariance matrix. We assume that we can sample from the unit normal distribution in one dimension.

First, we find the *Cholesky decomposition* of Σ . Any symmetric positive-definite matrix, such as Σ , can be decomposed as:

$$\Sigma = LL^T$$

where L is a lower triangular matrix with strictly positive diagonal entries.

We then sample Z_i from $\mathcal{N}(0, 1)$, for each dimension in our space. This generates a sample Z from $\mathcal{N}(0, I)$.

Finally, we set $X = LZ + \mu$. In general, if $W \sim \mathcal{N}(0, \Sigma)$, then applying the linear map A yields $A(W) \sim \mathcal{N}(0, A\Sigma A^T)$. Thus the distribution of X is $\mathcal{N}(\mu, LIL^T) = \mathcal{N}(\mu, \Sigma)$, as we required.

References

- [1] M. J. F. Gales. Maximum likelihood linear transformations for HMM-based speech recognition. *Computer Speech and Language*, 12(2):75–98, 1998.
- [2] David J. C. MacKay. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003. Available from <http://www.inference.phy.cam.ac.uk/mackay/itila/>.
- [3] K. Yu and M. J. F. Gales. Bayesian Adaptive Inference and Adaptive Training. *IEEE Transactions on Audio, Speech and Language Processing*, 15(6):1932–1943, 2007.