Chapter 3

Variational Bayesian Hidden Markov Models

3.1 Introduction

Hidden Markov models (HMMs) are widely used in a variety of fields for modelling time series data, with applications including speech recognition, natural language processing, protein sequence modelling and genetic alignment, general data compression, information retrieval, motion video analysis and object/people tracking, and financial time series prediction. The core theory of HMMs was developed principally by Baum and colleagues (Baum and Petrie, 1966; Baum et al., 1970), with initial applications to elementary speech processing, integrating with linguistic models, and making use of insertion and deletion states for variable length sequences (Bahl and Jelinek, 1975). The popularity of HMMs soared the following decade, giving rise to a variety of elaborations, reviewed in Juang and Rabiner (1991). More recently, the realisation that HMMs can be expressed as Bayesian networks (Smyth et al., 1997) has given rise to more complex and interesting models, for example, factorial HMMs (Ghahramani and Jordan, 1997), tree-structured HMMs (Jordan et al., 1997), and switching state-space models (Ghahramani and Hinton, 2000). An introduction to HMM modelling in terms of graphical models can be found in Ghahramani (2001).

This chapter is arranged as follows. In section 3.2 we briefly review the learning and inference algorithms for the standard HMM, including ML and MAP estimation. In section 3.3 we show how an exact Bayesian treatment of HMMs is intractable, and then in section 3.4 follow MacKay (1997) and derive an approximation to a Bayesian implementation using a variational lower bound on the marginal likelihood of the observations. In section 3.5 we present the results of synthetic experiments in which VB is shown to avoid overfitting unlike ML. We also compare ML, MAP and VB algorithms' ability to learn HMMs on a simple benchmark problem of

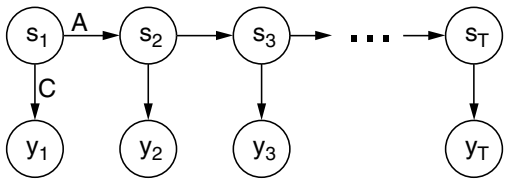


Figure 3.1: Graphical model representation of a hidden Markov model. The hidden variables $\mathbf{s}_t$ transition with probabilities specified in the rows of A, and at each time step emit an observation symbol $\mathbf{y}_t$ according to the probabilities in the rows of C.

discriminating between forwards and backwards English sentences. We present conclusions in section 3.6.

Whilst this chapter is not intended to be a novel contribution in terms of the variational Bayesian HMM, which was originally derived in the unpublished technical report of MacKay (1997), it has nevertheless been included for completeness to provide an immediate and straightforward example of the theory presented in chapter 2. Moreover, the wide applicability of HMMs makes the derivations and experiments in this chapter of potential general interest.

3.2 Inference and learning for maximum likelihood HMMs

We briefly review the learning and inference procedures for hidden Markov models (HMMs), adopting a similar notation to Rabiner and Juang (1986). An HMM models a sequence of p-valued discrete observations (symbols) $\mathbf{y}_{1:T} = \{y_1, \ldots, y_T\}$ by assuming that the observation at time t, y_t, was produced by a k-valued discrete hidden state s_t, and that the sequence of hidden states $\mathbf{s}_{1:T} = \{s_1, \ldots, s_T\}$ was generated by a first-order Markov process. That is to say the complete-data likelihood of a sequence of length T is given by:

$$p(\mathbf{s}_{1:T}, \mathbf{y}_{1:T}) = p(s_1)p(y_1 \mid s_1) \prod_{t=2}^{T} p(s_t \mid s_{t-1})p(y_t \mid s_t) \,. \tag{3.1}$$

where $p(s_1)$ is the prior probability of the first hidden state, $p(s_t \mid s_{t-1})$ denotes the probability of *transitioning* from state s_{t-1} to state s_t (out of a possible k states), and $p(y_t \mid s_t)$ are the *emission* probabilities for each of p symbols at each state. In this simple HMM, all the parameters are assumed stationary, and we assume a fixed finite number of hidden states and number of observation symbols. The joint probability (3.1) is depicted as a graphical model in figure 3.1. For simplicity we first examine just a single sequence of observations, and derive learning and inference procedures for this case; it is straightforward to extend the results to multiple i.i.d. sequences.

The probability of the observations $\mathbf{y}_{1:T}$ results from summing over all possible hidden state sequences,

$$p(\mathbf{y}_{1:T}) = \sum_{\mathbf{s}_{1:T}} p(\mathbf{s}_{1:T}, \mathbf{y}_{1:T}) \,. \tag{3.2}$$

The set of parameters for the initial state prior, transition, and emission probabilities are represented by the parameter $\boldsymbol{\theta}$:

$$\boldsymbol{\theta} = (A, C, \boldsymbol{\pi}) \tag{3.3}$$

$$A = \{a_{jj'}\} : a_{jj'} = p(s_t = j' \,|\, s_{t-1} = j) \qquad \text{state transition matrix } (k \times k) \tag{3.4}$$

$$C = \{c_{jm}\} : c_{jm} = p(y_t = m \,|\, s_t = j) \qquad \text{symbol emission matrix } (k \times p) \tag{3.5}$$

$$\boldsymbol{\pi} = \{\pi_j\} : \pi_j = p(s_1 = j) \qquad \text{initial hidden state prior } (k \times 1) \tag{3.6}$$

obeying the normalisation constraints:

$$A = \{a_{jj'}\} : \sum_{j'=1}^{k} a_{jj'} = 1 \quad \forall j \tag{3.7}$$

$$C = \{c_{jm}\} : \sum_{m=1}^{p} c_{jm} = 1 \quad \forall j \tag{3.8}$$

$$\boldsymbol{\pi} = \{\pi_j\} : \sum_{j=1}^{k} \pi_j = 1 \,. \tag{3.9}$$

For mathematical convenience we represent the state of the hidden variables using k-dimensional binary column vectors. For example, if s_t is in state j, then $\mathbf{s}_t$ is a vector of zeros with '1' in the jth entry. We use a similar notation for the observations y_t. The Kronecker-δ function is used to query the state, such that $\mathbf{s}_{t,j} = \delta(s_t, j)$ returns 1 if s_t is in state j, and zero otherwise.

Using the vectorial form of the hidden and observed variables, the initial hidden state, transition, and emission probabilities can be written as

$$p(s_1 \,|\, \boldsymbol{\pi}) = \prod_{j=1}^{k} \pi_j^{\mathbf{s}_{1,j}} \tag{3.10}$$

$$p(s_t \,|\, s_{t-1}, A) = \prod_{j=1}^{k} \prod_{j'=1}^{k} a_{jj'}^{\mathbf{s}_{t,j'}\mathbf{s}_{t-1,j}} \tag{3.11}$$

$$p(y_t \,|\, s_t, C) = \prod_{j=1}^{k} \prod_{m=1}^{p} c_{jm}^{\mathbf{s}_{t,j}\mathbf{y}_{t,m}} \tag{3.12}$$

and the log complete-data likelihood from (3.1) becomes:

$$\ln p(\mathbf{s}_{1:T}, \mathbf{y}_{1:T} \,|\, \boldsymbol{\theta}) = \sum_{j=1}^{k} \mathbf{s}_{1,j} \ln \pi_j + \sum_{t=2}^{T} \sum_{j=1}^{k} \sum_{j'=1}^{k} \mathbf{s}_{t-1,j} \ln a_{jj'} \mathbf{s}_{t,j'} + \sum_{t=1}^{T} \sum_{j=1}^{k} \sum_{m=1}^{p} \mathbf{s}_{t,j} \ln c_{jm} \mathbf{y}_{t,m} \tag{3.13}$$

$$= \mathbf{s}_1^\top \ln \boldsymbol{\pi} + \sum_{t=2}^{T} \mathbf{s}_{t-1}^\top \ln A \, \mathbf{s}_t + \sum_{t=1}^{T} \mathbf{s}_t^\top \ln C \, \mathbf{y}_t \,, \tag{3.14}$$

where the logarithms of the vector $\boldsymbol{\pi}$ and matrices A and C are taken element-wise. We are now in a position to derive the EM algorithm for ML parameter estimation for HMMs.

M step

Learning the maximum likelihood parameters of the model entails finding those settings of A, C and $\boldsymbol{\pi}$ which maximise the probability of the observed data (3.2). In chapter 2 we showed that the M step, as given by equation (2.31), is

$$\textbf{M step:} \qquad \boldsymbol{\theta}^{(t+1)} \leftarrow \arg\max_{\boldsymbol{\theta}} \sum_{\mathbf{s}_{1:T}} p(\mathbf{s}_{1:T} \,|\, \mathbf{y}_{1:T}, \boldsymbol{\theta}^{(t)}) \ln p(\mathbf{s}_{1:T}, \mathbf{y}_{1:T} \,|\, \boldsymbol{\theta}) \,, \tag{3.15}$$

where the superscript notation $^{(t)}$ denotes iteration number. Note in particular that the log likelihood in equation (3.14) is a sum of separate contributions involving $\boldsymbol{\pi}$, A and C, and summing over the hidden state sequences does not couple the parameters. Therefore we can individually optimise each parameter of the HMM:

$$\boldsymbol{\pi} \;:\; \pi_j \leftarrow \langle \mathbf{s}_{1,j} \rangle \tag{3.16}$$

$$A \;:\; a_{jj'} \leftarrow \frac{\sum_{t=2}^{T} \langle \mathbf{s}_{t-1,j} \mathbf{s}_{t,j'} \rangle}{\sum_{t=2}^{T} \langle \mathbf{s}_{t-1,j} \rangle} \tag{3.17}$$

$$C \;:\; c_{jm} \leftarrow \frac{\sum_{t=1}^{T} \langle \mathbf{s}_{t,j} \mathbf{y}_{t,m} \rangle}{\sum_{t=1}^{T} \langle \mathbf{s}_{t,j} \rangle} \tag{3.18}$$

where the angled brackets $\langle \cdot \rangle$ denote expectation with respect to the posterior distribution over the hidden state sequence, $p(\mathbf{s}_{1:T} \,|\, \mathbf{y}_{1:T}, \boldsymbol{\theta}^{(t)})$, as calculated from the E step.

E step: forward-backward algorithm

The E step is carried out using a dynamic programming trick which utilises the conditional independence of future hidden states from past hidden states given the setting of the current

hidden state. We define $\alpha_t(\mathbf{s}_t)$ to be the posterior over the hidden state s_t given the observed sequence up to and including time t:

$$\alpha_t(\mathbf{s}_t) \equiv p(\mathbf{s}_t \,|\, \mathbf{y}_{1:t}) \,, \tag{3.19}$$

and form the forward recursion from $t = 1, \ldots, T$:

$$\alpha_t(\mathbf{s}_t) = \frac{1}{p(\mathbf{y}_t \,|\, \mathbf{y}_{1:t-1})} \sum_{\mathbf{s}_{t-1}} p(\mathbf{s}_{t-1} \,|\, \mathbf{y}_{1:t-1}) p(\mathbf{s}_t \,|\, \mathbf{s}_{t-1}) p(\mathbf{y}_t \,|\, \mathbf{s}_t) \tag{3.20}$$

$$= \frac{1}{\zeta_t(\mathbf{y}_t)} \left[\sum_{\mathbf{s}_{t-1}} \alpha_{t-1}(\mathbf{s}_{t-1}) p(\mathbf{s}_t \,|\, \mathbf{s}_{t-1}) \right] p(\mathbf{y}_t \,|\, \mathbf{s}_t) \,, \tag{3.21}$$

where in the first time step $p(\mathbf{s}_t \,|\, \mathbf{s}_{t-1})$ is replaced with the prior $p(\mathbf{s}_1 \,|\, \boldsymbol{\pi})$, and for $t = 1$ we require the convention $\alpha_0(\mathbf{s}_0) = 1$. Here, $\zeta_t(\mathbf{y}_t)$ is a normalisation constant, a function of $\mathbf{y}_t$, given by

$$\zeta_t(\mathbf{y}_t) \equiv p(\mathbf{y}_t \,|\, \mathbf{y}_{1:t-1}) \,. \tag{3.22}$$

Note that as a by-product of computing these normalisation constants we can compute the probability of the sequence:

$$p(\mathbf{y}_{1:T}) = p(y_1) p(y_2 \,|\, y_1) \ldots p(y_T \,|\, y_{1:T-1}) = \prod_{t=1}^{T} p(y_t \,|\, y_{1:t-1}) = \prod_{t=1}^{T} \zeta_t(\mathbf{y}_t) = \mathcal{Z}(\mathbf{y}_{1:T}) \,. \tag{3.23}$$

Obtaining these normalisation constants using a forward pass is simply equivalent to integrating out the hidden states one after the other in the forward ordering, as can be seen by writing the incomplete-data likelihood in the following way:

$$p(\mathbf{y}_{1:T}) = \sum_{\mathbf{s}_{1:T}} p(\mathbf{s}_{1:T}, \mathbf{y}_{1:T}) \tag{3.24}$$

$$= \sum_{\mathbf{s}_1} \cdots \sum_{\mathbf{s}_T} p(\mathbf{s}_1) p(\mathbf{y}_1 \,|\, \mathbf{s}_1) \prod_{t=2}^{T} p(\mathbf{s}_t \,|\, \mathbf{s}_{t-1}) p(\mathbf{y}_t \,|\, \mathbf{s}_t) \tag{3.25}$$

$$= \sum_{\mathbf{s}_1} p(\mathbf{s}_1) p(\mathbf{y}_1 \,|\, \mathbf{s}_1) \cdots \sum_{\mathbf{s}_T} p(\mathbf{s}_T \,|\, \mathbf{s}_{T-1}) p(\mathbf{y}_T \,|\, \mathbf{s}_T) \,. \tag{3.26}$$

Similarly to the forward recursion, the backward recursion is carried out from $t = T, \ldots, 1$:

$$\beta_t(\mathbf{s}_t) \equiv p(\mathbf{y}_{(t+1):T} \,|\, \mathbf{s}_t) \tag{3.27}$$

$$= \sum_{\mathbf{s}_{t+1}} p(\mathbf{y}_{t+2:T} \,|\, \mathbf{s}_{t+1}) p(\mathbf{s}_{t+1} \,|\, \mathbf{s}_t) p(\mathbf{y}_{t+1} \,|\, \mathbf{s}_{t+1}) \tag{3.28}$$

$$= \sum_{\mathbf{s}_{t+1}} \beta_{t+1}(\mathbf{s}_{t+1}) p(\mathbf{s}_{t+1} \,|\, \mathbf{s}_t) p(\mathbf{y}_{t+1} \,|\, \mathbf{s}_{t+1}) \,, \tag{3.29}$$

with the end condition $\beta_T(\mathbf{s}_T) = 1$, as there is no future observed data beyond $t = T$.

The forward and backward recursions can be executed in parallel as neither depends on the results of the other. The quantities $\{\alpha_t\}_{t=1}^T$ and $\{\beta_t\}_{t=1}^T$ are now combined to obtain the single and pairwise state marginals:

$$p(\mathbf{s}_t \,|\, \mathbf{y}_{1:T}) \propto p(\mathbf{s}_t \,|\, \mathbf{y}_{1:t}) p(\mathbf{y}_{t+1:T} \,|\, \mathbf{s}_t) \tag{3.30}$$

$$= \alpha_t(\mathbf{s}_t)\beta_t(\mathbf{s}_t)\,, \qquad t = 1, \ldots, T \tag{3.31}$$

and

$$p(\mathbf{s}_{t-1}, \mathbf{s}_t \,|\, \mathbf{y}_{1:T}) \propto p(\mathbf{s}_{t-1} \,|\, \mathbf{y}_{1:t-1}) p(\mathbf{s}_t \,|\, \mathbf{s}_{t-1}) p(\mathbf{y}_t \,|\, \mathbf{s}_t) p(\mathbf{y}_{t+1:T} \,|\, \mathbf{s}_t) \tag{3.32}$$

$$= \alpha_{t-1}(\mathbf{s}_{t-1}) p(\mathbf{s}_t \,|\, \mathbf{s}_{t-1}) p(\mathbf{y}_t \,|\, \mathbf{s}_t) \beta_t(\mathbf{s}_t)\,, \qquad t = 2, \ldots, T \tag{3.33}$$

which give the expectations required for the M steps (3.16-3.18),

$$\langle \mathbf{s}_{t,j} \rangle = \frac{\alpha_{t,j}\beta_{t,j}}{\sum_{j'=1}^k \alpha_{t,j'}\beta_{t,j'}} \tag{3.34}$$

$$\langle \mathbf{s}_{t-1,j}\mathbf{s}_{t,j'} \rangle = \frac{\alpha_{t-1,j} a_{jj'} p(\mathbf{y}_t \,|\, \mathbf{s}_{t,j'}) \beta_{t,j'}}{\sum_{j=1}^k \sum_{j'=1}^k \alpha_{t-1,j} a_{jj'} p(\mathbf{y}_t \,|\, \mathbf{s}_{t,j'}) \beta_{t,j'}}\,. \tag{3.35}$$

The E and M steps described above form the iterations for the celebrated Baum-Welch algorithm (Baum et al., 1970). From the analysis in chapter 2, we can prove that each iteration of EM is guaranteed to increase, or leave unchanged, the log likelihood of the parameters, and converge to a local maximum.

When learning an HMM from multiple i.i.d. sequences $\{\mathbf{y}_{i,1:T_i}\}_{i=1}^n$ which are not necessarily constrained to have the same lengths $\{T_i\}_{i=1}^n$, the E and M steps remain largely the same. The E step is performed for each sequence separately using the forward-backward algorithm, and the M step then uses statistics pooled from all the sequences to estimate the mostly likely parameters.

HMMs as described above can be generalised in many ways. Often observed data are recorded as real-valued sequences and can be modelled by replacing the emission process $p(y_t \,|\, s_t)$ with a Gaussian or mixture of Gaussians distribution: each sequence of the HMM can now be thought of as defining a sequence of data drawn from a mixture model whose hidden state labels for the mixture components are no longer i.i.d., but evolve with Markov dynamics. Note that inference in such models remains possible using the forward and backward recursions, with only a change to the emission probabilities $p(y_t \,|\, s_t)$; furthermore, the M steps for learning the parameters $\boldsymbol{\pi}$ and A for the hidden state transitions remain identical.

Exactly analogous inference algorithms exist for the Linear Dynamical Systems (LDS) model, except that both the hidden state transition and emission processes are continuous (referred to

as dynamics and output processes, respectively). In the rest of this chapter we will see how a variational Bayesian treatment of HMMs results in a straightforwardly modified Baum-Welch algorithm, and as such it is a useful pedagogical example of the VB theorems given in chapter 2. On the other hand, for the LDS models the modified VB algorithms become substantially harder to derive and implement — these are the subject of chapter 5.

3.3 Bayesian HMMs

As has already been discussed in chapters 1 and 2, the maximum likelihood approach to learning models from data does not take into account model complexity, and so is susceptible to overfitting the data. More complex models can usually give ever-increasing likelihoods to the data. For a hidden Markov model, the complexity is related to several aspects: the number of hidden states k in the model, the degree of connectivity in the hidden state transition matrix A, and the distribution of probabilities to the symbols by each hidden state, as specified in the emission matrix, C. More generally the complexity is related to the richness of possible data sets that the model can produce. There are $k(k-1)$ parameters in the transition matrix, and $k(p-1)$ in the emission matrix, and so if there are many different observed symbols or if we expect to require more than a few hidden states then, aside from inference becoming very costly, the number of parameters to be fit may begin to overwhelm the amount of data available. Traditionally, in order to avoid overfitting, researchers have limited the complexity of their models in line with the amount of data they have available, and have also used sophisticated modifications to the basic HMM to reduce the number of free parameters. Such modifications include: parameter-tying, enforcing sparsity constraints (for example limiting the number of candidates a state can transition to or symbols it can emit), or constraining the form of the hidden state transitions (for example employing a strict left-to-right ordering of the hidden states).

A common technique for removing excessive parameters from a model is to regularise them using a prior, and then to maximise the a posteriori probability of the parameters (MAP). We will see below that it is possible to apply this type of regularisation to the multinomial parameters of the transition and emission probabilities using certain Dirichlet priors. However we would still expect the results of MAP optimisation to be susceptible to overfitting given that it searches for the maximum of the posterior density as opposed to integrating over the posterior distribution. Cross-validation is another method often employed to minimise the amount of overfitting, by repeatedly training subsets of the available data and evaluating the error on the remaining data. Whilst cross-validation is quite successful in practice, it has the drawback that it requires many sessions of training and so is computationally expensive, and often needs large amounts of data to obtain low-variance estimates of the expected test errors. Moreover, it is cumbersome to cross-validate over the many different ways in which model complexity could vary.

The Bayesian approach to learning treats the model parameters as unknown quantities and, prior to observing the data, assigns a set of beliefs over these quantities in the form of prior distributions. In the light of data, Bayes' rule can be used to infer the posterior distribution over the parameters. In this way the parameters of the model are treated as hidden variables and are integrated out to form the marginal likelihood:

$$p(\mathbf{y}_{1:T}) = \int d\boldsymbol{\theta}\; p(\boldsymbol{\theta}) p(\mathbf{y}_{1:T} \,|\, \boldsymbol{\theta}) \qquad \text{where } \boldsymbol{\theta} = (\boldsymbol{\pi}, A, C)\;. \tag{3.36}$$

This Bayesian integration embodies the principle of Occam's razor since it automatically penalises those models with more parameters (see section 1.2.1; also see MacKay, 1992). A natural choice for parameter priors over $\boldsymbol{\pi}$, the rows of A, and the rows of C are Dirichlet distributions. Whilst there are many possible choices, Dirichlet distributions have the advantage that they are conjugate to the complete-data likelihood terms given in equations (3.1) (and with foresight we know that these forms will yield tractable variational Bayesian algorithms):

$$p(\boldsymbol{\theta}) = p(\boldsymbol{\pi}) p(A) p(C) \tag{3.37}$$

$$p(\boldsymbol{\pi}) = \mathrm{Dir}(\{\pi_1, \ldots, \pi_k\} \,|\, \mathbf{u}^{(\pi)})) \tag{3.38}$$

$$p(A) = \prod_{j=1}^{k} \mathrm{Dir}(\{a_{j1}, \ldots, a_{jk}\} \,|\, \mathbf{u}^{(A)}) \tag{3.39}$$

$$p(C) = \prod_{j=1}^{k} \mathrm{Dir}(\{c_{j1}, \ldots, c_{jp}\} \,|\, \mathbf{u}^{(C)})\;. \tag{3.40}$$

Here, for each matrix the same single hyperparameter vector is used for every row. This hyperparameter sharing can be motivated because the hidden states are identical a priori. The form of the Dirichlet prior, using $p(\boldsymbol{\pi})$ as an example, is

$$p(\boldsymbol{\pi}) = \frac{\Gamma(u_0^{(\pi)})}{\prod_{j=1}^{k} \Gamma(u_j^{(\pi)})} \prod_{j=1}^{k} \pi_j^{u_j^{(\pi)} - 1}\;, \qquad u_j^{(\pi)} > 0,\; \forall\, j\;, \tag{3.41}$$

where $u_0^{(\pi)} = \sum_{j=1}^{k} u_j^{(\pi)}$ is the *strength* of the prior, and the positivity constraint on the hyperparameters is required for the prior to be proper. Conjugate priors have the intuitive interpretation of providing hypothetical observations to augment those provided by the data (see section 1.2.2). If these priors are used in a *maximum a posteriori* (MAP) estimation algorithm for HMMs, the priors add imaginary counts to the M steps. Taking the update for A as an example, equation (3.17) is modified to

$$A \;:\; a_{jj'} \leftarrow \frac{(u_{j'}^{(A)} - 1) + \sum_{t=2}^{T} \langle \mathbf{s}_{t-1,j} \mathbf{s}_{t,j'} \rangle}{\sum_{j'=1}^{k} (u_{j'}^{(A)} - 1) + \sum_{t=2}^{T} \langle \mathbf{s}_{t-1,j} \rangle}\;. \tag{3.42}$$

Researchers tend to limit themselves to hyperparameters $u_j \geq 1$ such that this MAP estimate is guaranteed to yield positive probabilities. However there are compelling reasons for having hy-

perparameters $u_j \leq 1$ (as discussed in MacKay and Peto, 1995; MacKay, 1998), and these arise naturally as described below. It should be emphasised that the MAP solution is not invariant to reparameterisations, and so (3.42) is just one possible result. For example, reparameterisation into the softmax basis yields a MAP estimate without the '-1' terms, which also coincides with the predictive distribution obtained from integrating over the posterior. The experiments carried out in this chapter for MAP learning do so in this basis.

We choose to use symmetric Dirichlet priors, with a fixed strength f, i.e.

$$\mathbf{u}^{(A)} = \left[\frac{f^{(A)}}{k}, \ldots, \frac{f^{(A)}}{k}\right]^\top , \qquad s.t. \sum_{j=1}^{k} u_j^{(A)} = f^{(A)} , \tag{3.43}$$

and similarly so for $\mathbf{u}^{(C)}$ and $\mathbf{u}^{(\pi)}$. A fixed strength is chosen because we do not want the amount of imaginary data to increase with the complexity of the model. This relates to a key issue in Bayesian prior specification regarding the *scaling* of model priors. Imagine an un-scaled prior over each row of A with hyperparameter $\left[f^{(A)}, \ldots, f^{(A)}\right]^\top$, where the division by k has been omitted. With a fixed strength prior, the contribution to the posterior distributions over the parameters from the prior diminishes with increasing data, whereas with the un-scaled prior the contribution increases linearly with the number of hidden states and can become greater than the amount of observed data for sufficiently large k. This means that for sufficiently complex models the modification terms in (3.42) would obfuscate the data entirely. This is clearly undesirable, and so the $\frac{1}{k}$ scaling of the hyperparameter entries is used. Note that this scaling will result in hyperparameters ≤ 1 for sufficiently large k.

The marginal probability of a sequence of observations is given by

$$p(\mathbf{y}_{1:T}) = \int d\boldsymbol{\pi}\; p(\boldsymbol{\pi}) \int dA\; p(A) \int dC\; p(C) \sum_{\mathbf{s}_{1:T}} p(\mathbf{s}_{1:T}, \mathbf{y}_{1:T} \,|\, \boldsymbol{\pi}, A, C) , \tag{3.44}$$

where the dependence on the hyperparameters is implicitly assumed as they are fixed beforehand. Unfortunately, we can no longer use the dynamic programming trick of the forward-backward algorithm, as the hidden states are now coupled by the integration over the parameters. Intuitively this means that, because the parameters of the model have become uncertain quantities, the future hidden states $\mathbf{s}_{(t+1):T}$ are no longer independent of past hidden states $\mathbf{s}_{1:(t-1)}$ given the current state $\mathbf{s}_t$. The summation and integration operations in (3.44) can be interchanged, but there are still an intractable number of possible sequences to sum over, a number exponential in the length of the sequence. This intractability becomes even worse with multiple sequences, as hidden states of different sequences also become dependent in the posterior.

It is true that for any *given* setting of the parameters, the likelihood calculation is possible, as is finding the distribution over possible hidden state sequences using the forward-backward algorithm; but since the parameters are continuous this insight is not useful for calculating

(3.44). It is also true that for any *given* trajectory representing a single hidden state sequence, we can treat the hidden variables as observed and analytically integrate out the parameters to obtain the marginal likelihood; but since the number of such trajectories is exponential in the sequence length (k^T), this approach is also ruled out.

These considerations form the basis of a very simple and elegant algorithm due to Stolcke and Omohundro (1993) for estimating the marginal likelihood of an HMM. In that work, the posterior distribution over hidden state trajectories is approximated with the most likely sequence, obtained using a Viterbi algorithm for discrete HMMs (Viterbi, 1967). This single sequence (let us assume it is unique) is then treated as observed data, which causes the parameter posteriors to be Dirichlet, which are then easily integrated over to form an estimate of the marginal likelihood. The MAP parameter setting (the mode of the Dirichlet posterior) is then used to infer the most probable hidden state trajectory to iterate the process. Whilst the reported results are impressive, substituting MAP estimates for both parameters and hidden states seems safe only if: there is plenty of data to determine the parameters (i.e. many long sequences); and the individual sequences are long enough to reduce any ambiguity amongst the hidden state trajectories.

Markov chain Monte Carlo (MCMC) methods can be used to approximate the posterior distribution over parameters (Robert et al., 1993), but in general it is hard to assess the convergence and reliability of the estimates required for learning. An analytically-based approach is to approximate the posterior distribution over the parameters with a Gaussian, which usually allows the integral to become tractable. Unfortunately the Laplace approximation is not well-suited to bounded or constrained parameters (e.g. sum-to-one constraints), and computation of the likelihood Hessian can be computationally expensive. In MacKay (1998) an argument for transforming the Dirichlet prior into the softmax basis is presented, although to the best of our knowledge this approach is not widely used for HMMs.

3.4 Variational Bayesian formulation

In this section we derive the variational Bayesian implementation of HMMs, first presented in MacKay (1997). We show that by making only the approximation that the posterior over hidden variables and parameters factorises, an approximate posterior distribution over hidden state trajectories can be inferred under an *ensemble* of model parameters, and how an approximate posterior distribution over parameters can be analytically obtained from the sufficient statistics of the hidden state.

3.4.1 Derivation of the VBEM optimisation procedure

Our choice of priors $p(\boldsymbol{\theta})$ and the complete-data likelihood $p(\mathbf{s}_{1:T}, \mathbf{y}_{1:T} \,|\, \boldsymbol{\theta})$ for HMMs satisfy conditions (2.80) and (2.88) respectively, for membership of the conjugate-exponential (CE) family. Therefore it is possible to apply the results of theorem 2.2 directly to obtain the VBM and VBE steps. The derivation is given here step by step, and the ideas of chapter 2 brought in gradually. We begin with the log marginal likelihood for an HMM (3.36), and lower bound it by introducing any distribution over the parameters and hidden variables $q(\boldsymbol{\pi}, A, C, \mathbf{s}_{1:T})$:

$$\ln p(\mathbf{y}_{1:T}) = \ln \int d\boldsymbol{\pi} \int dA \int dC \sum_{\mathbf{s}_{1:T}} p(\boldsymbol{\pi}, A, C) p(\mathbf{y}_{1:T}, \mathbf{s}_{1:T} \,|\, \boldsymbol{\pi}, A, C) \tag{3.45}$$

$$\geq \int d\boldsymbol{\pi} \int dA \int dC \sum_{\mathbf{s}_{1:T}} q(\boldsymbol{\pi}, A, C, \mathbf{s}_{1:T}) \ln \frac{p(\boldsymbol{\pi}, A, C) p(\mathbf{y}_{1:T}, \mathbf{s}_{1:T} \,|\, \boldsymbol{\pi}, A, C)}{q(\boldsymbol{\pi}, A, C, \mathbf{s}_{1:T})} \,. \tag{3.46}$$

This inequality is tight when $q(\boldsymbol{\pi}, A, C, \mathbf{s}_{1:T})$ is set to the exact posterior over hidden variables and parameters $p(\boldsymbol{\pi}, A, C, \mathbf{s}_{1:T} \,|\, \mathbf{y}_{1:T})$, but it is intractable to compute this distribution. We make progress by assuming that the posterior is factorised:

$$p(\boldsymbol{\pi}, A, C, \mathbf{s}_{1:T} \,|\, \mathbf{y}_{1:T}) \approx q(\boldsymbol{\pi}, A, C) q(\mathbf{s}_{1:T}) \tag{3.47}$$

which gives a lower bound of the form

$$\ln p(\mathbf{y}_{1:T}) \geq \int d\boldsymbol{\pi} \int dA \int dC \sum_{\mathbf{s}_{1:T}} q(\boldsymbol{\pi}, A, C, \mathbf{s}_{1:T}) \ln \frac{p(\boldsymbol{\pi}, A, C) p(\mathbf{y}_{1:T}, \mathbf{s}_{1:T} \,|\, \boldsymbol{\pi}, A, C)}{q(\boldsymbol{\pi}, A, C, \mathbf{s}_{1:T})} \tag{3.48}$$

$$= \int d\boldsymbol{\pi} \int dA \int dC \; q(\boldsymbol{\pi}, A, C) \Bigg[\ln \frac{p(\boldsymbol{\pi}, A, C)}{q(\boldsymbol{\pi}, A, C)} + \sum_{\mathbf{s}_{1:T}} q(\mathbf{s}_{1:T}) \ln \frac{p(\mathbf{y}_{1:T}, \mathbf{s}_{1:T} \,|\, \boldsymbol{\pi}, A, C)}{q(\mathbf{s}_{1:T})} \Bigg] \tag{3.49}$$

$$= \mathcal{F}(q(\boldsymbol{\pi}, A, C), q(\mathbf{s}_{1:T})) \,, \tag{3.50}$$

where the dependence on $\mathbf{y}_{1:T}$ is taken to be implicit. On taking functional derivatives of $\mathcal{F}$ with respect to $q(\boldsymbol{\pi}, A, C)$ we obtain

$$\ln q(\boldsymbol{\pi}, A, C) = \ln p(\boldsymbol{\pi}, A, C) \langle \ln p(\mathbf{y}_{1:T}, \mathbf{s}_{1:T} \,|\, \boldsymbol{\pi}, A, C) \rangle_{q(\mathbf{s}_{1:T})} + c \tag{3.51}$$

$$\begin{aligned} &= \ln p(\boldsymbol{\pi}) + \ln p(A) + \ln p(C) \\ &\quad + \langle \ln p(\mathbf{s}_1 \,|\, \boldsymbol{\pi}) \rangle_{q(\mathbf{s}_1)} + \langle \ln p(\mathbf{s}_{2:T} \,|\, \mathbf{s}_1, A) \rangle_{q(\mathbf{s}_{1:T})} \\ &\quad + \langle \ln p(\mathbf{y}_{1:T} \,|\, \mathbf{s}_{1:T}, C) \rangle_{q(\mathbf{s}_{1:T})} + c \,, \end{aligned} \tag{3.52}$$

where c is a normalisation constant. Given that the prior over the parameters (3.37) factorises, and the log complete-data likelihood (3.14) is a sum of terms involving each of $\boldsymbol{\pi}$, A, and C, the variational posterior over the parameters can be factorised *without further approximation* into:

$$q(\boldsymbol{\pi}, A, C) = q(\boldsymbol{\pi})q(A)q(C) \; . \tag{3.53}$$

Note that sometimes this independence is assumed beforehand and believed to concede accuracy, whereas we have seen that it falls out from a free-form extremisation of the posterior with respect to the entire variational posterior over the parameters $q(\boldsymbol{\pi}, A, C)$, and is therefore exact once the assumption of factorisation between hidden variables and parameters has been made.

The VBM step

The VBM step is obtained by taking functional derivatives of $\mathcal{F}$ with respect to each of these distributions and equating them to zero, to yield Dirichlet distributions:

$$q(\boldsymbol{\pi}) = \mathrm{Dir}(\{\pi_1, \ldots, \pi_k\} \,|\, \{w_1^{(\pi)}, \ldots, w_k^{(\pi)}\}) \tag{3.54}$$

$$\text{with} \quad w_j^{(\pi)} = u_j^{(\pi)} + \langle \delta(s_1, j) \rangle_{q(\mathbf{s}_{1:T})} \tag{3.55}$$

$$q(A) = \prod_{j=1}^{k} \mathrm{Dir}(\{a_{j1}, \ldots, a_{jk}\} \,|\, \{w_{j1}^{(A)}, \ldots, w_{jk}^{(A)}\}) \tag{3.56}$$

$$\text{with} \quad w_{jj'}^{(A)} = u_{j'}^{(A)} + \sum_{t=2}^{T} \langle \delta(s_{t-1}, j)\delta(s_t, j') \rangle_{q(\mathbf{s}_{1:T})} \tag{3.57}$$

$$q(C) = \prod_{j=1}^{k} \mathrm{Dir}(\{c_{j1}, \ldots, c_{jp}\} \,|\, \{w_{j1}^{(C)}, \ldots, w_{jp}^{(C)}\}) \tag{3.58}$$

$$\text{with} \quad w_{jq}^{(C)} = u_q^{(A)} + \sum_{t=1}^{T} \langle \delta(s_t, j)\delta(y_t, q) \rangle_{q(\mathbf{s}_{1:T})} \; . \tag{3.59}$$

These are straightforward applications of the result in theorem 2.2(b), which states that the variational posterior distributions have the same form as the priors with their hyperparameters augmented by sufficient statistics of the hidden state and observations.

The VBE step

Taking derivatives of $\mathcal{F}$ (3.49) with respect to the variational posterior over the hidden state $q(\mathbf{s}_{1:T})$ yields:

$$\ln q(\mathbf{s}_{1:T}) = \langle \ln p(\mathbf{s}_{1:T}, \mathbf{y}_{1:T} \,|\, \boldsymbol{\pi}, A, C) \rangle_{q(\boldsymbol{\pi})q(A)q(C)} - \ln \tilde{\mathcal{Z}}(\mathbf{y}_{1:T}) \; , \tag{3.60}$$

where $\tilde{\mathcal{Z}}(\mathbf{y}_{1:T})$ is an important normalisation constant that we will return to shortly. Substituting in the complete-data likelihood from (3.14) yields

$$\ln q(\mathbf{s}_{1:T}) = \left\langle \mathbf{s}_1^\top \ln \boldsymbol{\pi} + \sum_{t=2}^{T} \mathbf{s}_{t-1}^\top \ln A \, \mathbf{s}_t + \sum_{t=1}^{T} \mathbf{s}_t^\top \ln C \, \mathbf{y}_t \right\rangle_{q(\boldsymbol{\pi})q(A)q(C)} - \ln \tilde{\mathcal{Z}}(\mathbf{y}_{1:T}) \quad (3.61)$$

$$= \mathbf{s}_1^\top \langle \ln \boldsymbol{\pi} \rangle_{q(\boldsymbol{\pi})} + \sum_{t=2}^{T} \mathbf{s}_{t-1}^\top \langle \ln A \rangle_{q(A)} \, \mathbf{s}_t + \sum_{t=1}^{T} \mathbf{s}_t^\top \langle \ln C \rangle_{q(C)} \, \mathbf{y}_t - \ln \tilde{\mathcal{Z}}(\mathbf{y}_{1:T}) \, . \quad (3.62)$$

Note that (3.62) appears identical to the complete-data likelihood of (3.14) except that expectations are now taken of the logarithm of the parameters. Relating this to the result in corollary 2.2, the natural parameter vector $\boldsymbol{\phi}(\boldsymbol{\theta})$ is given by

$$\boldsymbol{\theta} = (\boldsymbol{\pi} \, , A \, , C) \quad (3.63)$$

$$\boldsymbol{\phi}(\boldsymbol{\theta}) = (\ln \boldsymbol{\pi} \, , \ln A \, , \ln C) \, , \quad (3.64)$$

and the expected natural parameter vector $\overline{\boldsymbol{\phi}}$ is given by

$$\overline{\boldsymbol{\phi}} \equiv \langle \boldsymbol{\phi}(\boldsymbol{\theta}) \rangle_{q(\boldsymbol{\theta})} = (\langle \ln \boldsymbol{\pi} \rangle_{q(\boldsymbol{\pi})}, \; \langle \ln A \rangle_{q(A)}, \; \langle \ln C \rangle_{q(C)}) \, . \quad (3.65)$$

Corollary 2.2 suggests that we can use a modified parameter, $\tilde{\boldsymbol{\theta}}$, in the same inference algorithm (forward-backward) in the VBE step. The modified parameter $\tilde{\boldsymbol{\theta}}$ satisfies $\overline{\boldsymbol{\phi}} = \boldsymbol{\phi}(\tilde{\boldsymbol{\theta}}) = \langle \boldsymbol{\phi}(\boldsymbol{\theta}) \rangle_{q(\boldsymbol{\theta})}$, and is obtained simply by using the inverse of the $\boldsymbol{\phi}$ operator:

$$\tilde{\boldsymbol{\theta}} = \boldsymbol{\phi}^{-1}(\langle \boldsymbol{\phi}(\boldsymbol{\theta}) \rangle_{q(\boldsymbol{\theta})}) = (\exp\langle \ln \boldsymbol{\pi} \rangle_{q(\boldsymbol{\pi})} \, , \exp\langle \ln A \rangle_{q(A)} \, , \exp\langle \ln C \rangle_{q(C)}) \quad (3.66)$$

$$= (\tilde{\boldsymbol{\pi}} \, , \tilde{A} \, , \tilde{C}) \, . \quad (3.67)$$

Note that the natural parameter mapping $\boldsymbol{\phi}$ operates separately on each of the parameters in the vector $\boldsymbol{\theta}$, which makes the inversion of the mapping $\boldsymbol{\phi}^{-1}$ straightforward. This is a consequence of these parameters being uncoupled in the complete-data likelihood. For other CE models, the inversion of the natural parameter mapping may not be as simple, since having uncoupled parameters is not necessarily a condition for CE family membership. In fact, in chapter 5 we encounter such a scenario for Linear Dynamical Systems.

It remains for us to calculate the expectations of the logarithm of the parameters under the Dirichlet distributions. We use the result that

$$\int d\boldsymbol{\pi} \; \mathrm{Dir}(\boldsymbol{\pi} \,|\, \mathbf{u}) \ln \pi_j = \psi(u_j) - \psi(\sum_{j=1}^{k} u_j) \, , \quad (3.68)$$

where ψ is the *digamma* function (see appendices A and C.1 for details). This yields

$$\tilde{\boldsymbol{\pi}} = \{\tilde{\pi}_j\} = \exp\left[\psi(w_j^{(\pi)}) - \psi(\sum_{j=1}^{k} w_j^{(\pi)})\right] \;:\; \sum_{j=1}^{k} \tilde{\pi}_j \leq 1 \tag{3.69}$$

$$\tilde{A} = \{\tilde{a}_{jj'}\} = \exp\left[\psi(w_{jj'}^{(A)}) - \psi(\sum_{j'=1}^{k} w_{jj'}^{(A)})\right] \;:\; \sum_{j'=1}^{k} \tilde{a}_{jj'} \leq 1 \quad \forall j \tag{3.70}$$

$$\tilde{C} = \{\tilde{c}_{jm}\} = \exp\left[\psi(w_{jm}^{(C)}) - \psi(\sum_{m=1}^{p} w_{jm}^{(C)})\right] \;:\; \sum_{m=1}^{p} \tilde{c}_{jm} \leq 1 \quad \forall j \;. \tag{3.71}$$

Note that taking geometric averages has resulted in sub-normalised probabilities. We may still use the forward-backward algorithm with these sub-normalised parameters, but should bear in mind that the normalisation constants (scaling factors) change. The forward pass (3.21) becomes

$$\alpha_t(\mathbf{s}_t) = \frac{1}{\tilde{\zeta}_t(\mathbf{y}_t)} \left[\sum_{\mathbf{s}_{t-1}} \alpha_{t-1}(\mathbf{s}_{t-1}) \tilde{p}(\mathbf{s}_t \,|\, \mathbf{s}_{t-1})\right] \tilde{p}(\mathbf{y}_t \,|\, \mathbf{s}_t) \;, \tag{3.72}$$

where $\tilde{p}(\mathbf{s}_t \,|\, \mathbf{s}_{t-1})$ and $\tilde{p}(\mathbf{y}_t \,|\, \mathbf{s}_t)$ are new subnormalised probability distributions according to the parameters $\tilde{A}, \tilde{C}$, respectively. Since $\alpha_t(\mathbf{s}_t)$ is the posterior probability of $\mathbf{s}_t$ given data $\mathbf{y}_{1:t}$, it must sum to one. This implies that, for any *particular* time step, the normalisation $\tilde{\zeta}_t(\mathbf{y}_t)$ must be smaller than if we had used normalised parameters. Similarly the backward pass becomes

$$\beta_t(\mathbf{s}_t) = \sum_{\mathbf{s}_{t+1}} \beta_{t+1}(\mathbf{s}_{t+1}) \tilde{p}(\mathbf{s}_{t+1} \,|\, \mathbf{s}_t) \tilde{p}(\mathbf{y}_{t+1} \,|\, \mathbf{s}_{t+1}) \;. \tag{3.73}$$

Computation of the lower bound $\mathcal{F}$

Recall from (3.22) that the product of the normalisation constants corresponds to the probability of the sequence. Here the product of normalisation constants corresponds to a different quantity:

$$\prod_{t=1}^{T} \tilde{\zeta}_t(\mathbf{y}_t) = \tilde{\mathcal{Z}}(\mathbf{y}_{1:T}) \tag{3.74}$$

which is the normalisation constant given in (3.60). Thus the modified forward-backward algorithm recursively computes the normalisation constant by integrating out each s_t in $q(\mathbf{s}_{1:T})$, as opposed to $p(\mathbf{s}_{1:T} \,|\, \mathbf{y}_{1:T})$. We now show how $\tilde{\mathcal{Z}}(\mathbf{y}_{1:T})$ is useful for computing the lower bound, just as $\mathcal{Z}(\mathbf{y}_{1:T})$ was useful for computing the likelihood in the ML system.

Using (3.49) the lower bound can be written as

$$\begin{aligned}\mathcal{F}(q(\boldsymbol{\pi}, A, C), q(\mathbf{s}_{1:T})) = &\int d\boldsymbol{\pi}\; q(\boldsymbol{\pi}) \ln \frac{p(\boldsymbol{\pi})}{q(\boldsymbol{\pi})} + \int dA\; q(A) \ln \frac{p(A)}{q(A)} + \int dC\; q(C) \ln \frac{p(C)}{q(C)} \\ &+ H(q(\mathbf{s}_{1:T})) \\ &+ \langle \ln p(\mathbf{s}_{1:T}, \mathbf{y}_{1:T} \,|\, \boldsymbol{\pi}, A, C) \rangle_{q(\boldsymbol{\pi})q(A)q(C)q(\mathbf{s}_{1:T})} \,, \end{aligned} \tag{3.75}$$

where $H(q(\mathbf{s}_{1:T}))$ is the entropy of the variational posterior distribution over hidden state sequences. Straight after a VBE step, the form of the hidden state posterior $q(\mathbf{s}_{1:T})$ is given by (3.60), and the entropy can be written:

$$H(q(\mathbf{s}_{1:T})) = -\sum_{\mathbf{s}_{1:T}} q(\mathbf{s}_{1:T}) \ln q(\mathbf{s}_{1:T}) \tag{3.76}$$

$$= -\sum_{\mathbf{s}_{1:T}} q(\mathbf{s}_{1:T}) \left[\langle \ln p(\mathbf{s}_{1:T}, \mathbf{y}_{1:T} \,|\, \boldsymbol{\pi}, A, C) \rangle_{q(\boldsymbol{\pi})q(A)q(C)} - \ln \tilde{\mathcal{Z}}(\mathbf{y}_{1:T}) \right] \tag{3.77}$$

$$= -\sum_{\mathbf{s}_{1:T}} q(\mathbf{s}_{1:T}) \langle \ln p(\mathbf{s}_{1:T}, \mathbf{y}_{1:T} \,|\, \boldsymbol{\pi}, A, C) \rangle_{q(\boldsymbol{\pi})q(A)q(C)} + \ln \tilde{\mathcal{Z}}(\mathbf{y}_{1:T}) \,. \tag{3.78}$$

Substituting this into (3.75) cancels the expected log complete-data likelihood terms, giving

$$\begin{aligned}\mathcal{F}(q(\boldsymbol{\pi}, A, C), q(\mathbf{s}_{1:T})) = &\int d\boldsymbol{\pi}\; q(\boldsymbol{\pi}) \ln \frac{p(\boldsymbol{\pi})}{q(\boldsymbol{\pi})} + \int dA\; q(A) \ln \frac{p(A)}{q(A)} + \int dC\; q(C) \ln \frac{p(C)}{q(C)} \\ &+ \ln \tilde{\mathcal{Z}}(\mathbf{y}_{1:T}) \end{aligned} \tag{3.79}$$

Therefore computing $\mathcal{F}$ for variational Bayesian HMMs consists of evaluating KL divergences between variational posterior and prior Dirichlet distributions for each row of $\boldsymbol{\pi}$, A, C (see appendix A), and collecting the modified normalisation constants $\{\tilde{\zeta}_t(y_t)\}_{t=1}^T$. In essence we have by-passed the difficulty of trying to compute the entropy of the hidden state by recursively computing it with the VBE step's forward pass. Note that this calculation is then only valid straight after the VBE step.

VB learning with multiple i.i.d. sequences is conceptually straightforward and very similar to that described above for ML learning. For the sake of brevity the reader is referred to the chapter on Linear Dynamical Systems, specifically section 5.3.8 and equation (5.152), from which the implementational details for variational Bayesian HMMs can readily be inferred.

Optimising the hyperparameters of the model is straightforward. Since the hyperparameters appear in $\mathcal{F}$ only in the KL divergence terms, maximising the marginal likelihood amounts to minimising the KL divergence between each parameter's variational posterior distribution and its prior distribution. We did not optimise the hyperparameters in the experiments, but instead examined several different settings.

3.4.2 Predictive probability of the VB model

In the Bayesian scheme, the predictive probability of a test sequence $\mathbf{y}' = \mathbf{y}'_{1:T'}$, given a set of training cases denoted by $\mathbf{y} = \{\mathbf{y}_{i,1:T_i}\}_{i=1}^{n}$, is obtained by averaging the predictions of the HMM with respect to the posterior distributions over its parameters $\boldsymbol{\theta} = \{\boldsymbol{\pi}, A, C\}$:

$$p(\mathbf{y}' \,|\, \mathbf{y}) = \int d\boldsymbol{\theta}\; p(\boldsymbol{\theta} \,|\, \mathbf{y}) p(\mathbf{y}' \,|\, \boldsymbol{\theta}) \;. \tag{3.80}$$

Unfortunately, for the very same reasons that the marginal likelihood of equation (3.44) is intractable, so is the predictive probability. There are several possible methods for approximating the predictive probability. One such method is to sample parameters from the posterior distribution and construct a Monte Carlo estimate. Should it not be possible to sample directly from the posterior, then importance sampling or its variants can be used. This process can be made more efficient by employing Markov chain Monte Carlo and related methods. Alternatively, the posterior distribution can be approximated with some form which when combined with the likelihood term becomes amenable to integration analytically; it is unclear which analytical forms might yield good approximations.

An alternative is to approximate the posterior distribution with the variational posterior distribution resulting from the VB optimisation:

$$p(\mathbf{y}' \,|\, \mathbf{y}) \approx \int d\boldsymbol{\theta}\; q(\boldsymbol{\theta}) p(\mathbf{y}' \,|\, \boldsymbol{\theta}) \;. \tag{3.81}$$

The variational posterior is a product of Dirichlet distributions, which is in the same form as the prior, and so we have not gained a great deal because we know this integral is intractable. However we can perform two lower bounds on this quantity to obtain:

$$\begin{aligned} p(\mathbf{y}' \,|\, \mathbf{y}) &\approx \int d\boldsymbol{\theta}\; q(\boldsymbol{\theta}) p(\mathbf{y}' \,|\, \boldsymbol{\theta}) && (3.82)\\ &\geq \exp \int d\boldsymbol{\theta}\; q(\boldsymbol{\theta}) \ln \sum_{\mathbf{s}'_{1:T'}} p(\mathbf{s}'_{1:T}, \mathbf{y}'_{1:T} \,|\, \boldsymbol{\theta}) && (3.83)\\ &\geq \exp \int d\boldsymbol{\theta}\; q(\boldsymbol{\theta}) \sum_{\mathbf{s}'_{1:T'}} q(\mathbf{s}'_{1:T'}) \ln \frac{p(\mathbf{s}'_{1:T}, \mathbf{y}'_{1:T} \,|\, \boldsymbol{\theta})}{q(\mathbf{s}'_{1:T'})} \;. && (3.84) \end{aligned}$$

Equation 3.84 is just the last term in the expression for the lower bound of the marginal likelihood of a training sequence given by (3.49), but with the test sequence in place of the training sequence. This insight provides us with the following method to evaluate the approximation. One simply carries out a VBE step on the test sequence, starting from the result of the last VBM step on the training set, and gathers the normalisation constants $\{\tilde{\mathcal{Z}}'_t\}_{t=1}^{T'_i}$ and takes the product of these. Whilst this is a very straightforward method, it should be remembered that it is only a bound on an approximation.

A different way to obtain the predictive probability is to assume that the model at the mean (or mode) of the variational posterior, with parameter $\boldsymbol{\theta}_{\text{MVB}}$, is representative of the distribution as a whole. The likelihood of the test sequence is then computed under the single model with those parameters, which is tractable:

$$p(\mathbf{y}' \,|\, \mathbf{y})_{\text{MVB}} = \sum_{\mathbf{s}'_{1:T}} p(\mathbf{s}'_{1:T}, \mathbf{y}'_{1:T} \,|\, \boldsymbol{\theta}_{\text{MVB}}) \,. \tag{3.85}$$

This approach is suggested as further work in MacKay (1997), and is discussed in the experiments described below.

3.5 Experiments

In this section we perform two experiments, the first on synthetic data to demonstrate the ability of the variational Bayesian algorithm to avoid overfitting, and the second on a toy data set to compare ML, MAP and VB algorithm performance at discriminating between forwards and backwards English character sequences.

3.5.1 Synthetic: discovering model structure

For this experiment we trained ML and VB hidden Markov models on examples of three types of sequences with a three-symbol alphabet $\{a, b, c\}$. Using standard regular expression notation, the first type of sequence was a substring of the regular grammar $(abc)^*$, the second a substring of $(acb)^*$, and the third from $(a^*b^*)^*$ where a and b symbols are emitted stochastically with probability $\frac{1}{2}$ each. For example, the training sequences included the following:

$$\begin{aligned}
\mathbf{y}_{1,1:T_1} &= (abcabcabcabcabcabcabcabcabcabcabc) \\
\mathbf{y}_{2,1:T_2} &= (bcabcabcabcabcabcabcabcabcabcabc) \\
&\vdots \\
\mathbf{y}_{12,1:T_{12}} &= (acbacbacbacbacbacbacb) \\
\mathbf{y}_{13,1:T_{13}} &= (acbacbacbacbacbacbacbacbacbacbacbac) \\
&\vdots \\
\mathbf{y}_{n-1,1:T_{n-1}} &= (baabaabbabaaaabbabaaabbaabbbaa) \\
\mathbf{y}_{n,1:T_n} &= (abaaabbababaabababbbbbaaabaaabba) \,.
\end{aligned}$$

In all, the training data consisted of 21 sequences of maximum length 39 symbols. Looking at these sequences, we would expect an HMM to require 3 hidden states to model $(abc)^*$, a dif-

ferent 3 hidden states to model $(acb)^*$, and a single self-transitioning hidden state stochastically emitting a and b symbols to model $(a^*b^*)^*$. This gives a total of 7 hidden states required to model the data perfectly. With this foresight we therefore chose HMMs with $k = 12$ hidden states to allow for some redundancy and room for overfitting.

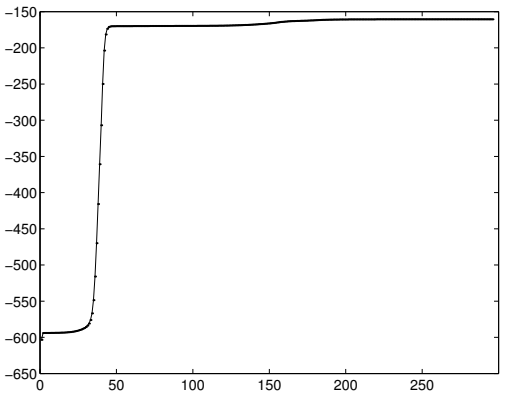
The parameters were initialised by drawing the components of the probability vectors from a uniform distribution and normalising. First the ML algorithm was run to convergence, and then the VB algorithm run *from that point* in parameter space to convergence. This was made possible by initialising each parameter's variational posterior distribution to be Dirichlet with the ML parameter as mean and a strength arbitrarily set to 10. For the MAP and VB algorithms, the prior over each parameter was a symmetric Dirichlet distribution of strength 4.

Figure 3.2 shows the profile of the likelihood of the data under the ML algorithm and the subsequent profile of the lower bound on the marginal likelihood under the VB algorithm. Note that it takes ML about 200 iterations to converge to a local optimum, and from this point it takes only roughly 25 iterations for the VB optimisation to converge — we might expect this as VB is initialised with the ML parameters, and so has less work to do.

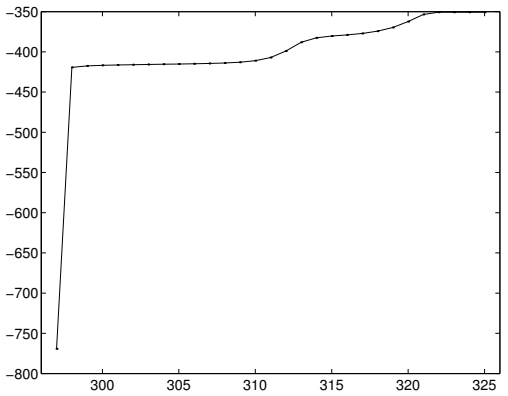
Figure 3.3 shows the recovered ML parameters and VB distributions over parameters for this problem. As explained above, we require 7 hidden states to model the data perfectly. It is clear from figure 3.3(a) that the ML model has used more hidden states than needed, that is to say it has overfit the structure of the model. Figures 3.3(b) and 3.3(c) show that the VB optimisation has removed excess transition and emission processes and, on close inspection, has recovered exactly the model that was postulated above. For example: state (4) self-transitions, and emits the symbols a and b in approximately equal proportions to generate the sequences $(a^*b^*)^*$; states (9,10,8) form a strong repeating path in the hidden state space which (almost) deterministically produce the sequences $(acb)^*$; and lastly the states (3,12,2) similarly interact to produce the sequences $(abc)^*$. A consequence of the Bayesian scheme is that *all* the entries of the transition and emission matrices are necessarily non-zero, and those states (1,5,6,7,11) that are not involved in the dynamics have uniform probability of transitioning to all others, and indeed of generating any symbol, in agreement with the symmetric prior. However these states have small probability of being used at all, as both the distribution $q(\boldsymbol{\pi})$ over the initial state parameter $\boldsymbol{\pi}$ is strongly peaked around high probabilities for the remaining states, and they have very low probability of being transitioned into by the active states.

3.5.2 Forwards-backwards English discrimination

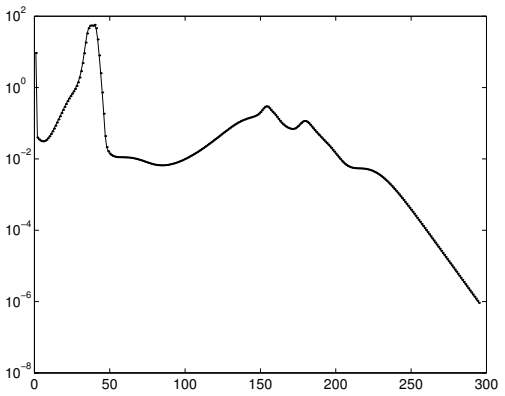
In this experiment, models learnt by ML, MAP and VB are compared on their ability to discriminate between forwards and backwards English text (this toy experiment is suggested in MacKay, 1997). A sentence is classified according to the predictive log probability under each



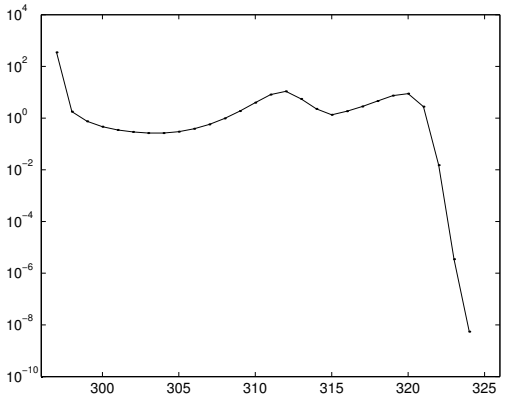
(a) ML: plot of the log likelihood of the data, $p(\mathbf{y}_{1:T} \mid \boldsymbol{\theta})$.



(b) VB: plot of the lower bound $\mathcal{F}(q(\mathbf{s}_{1:T}), q(\boldsymbol{\theta}))$.

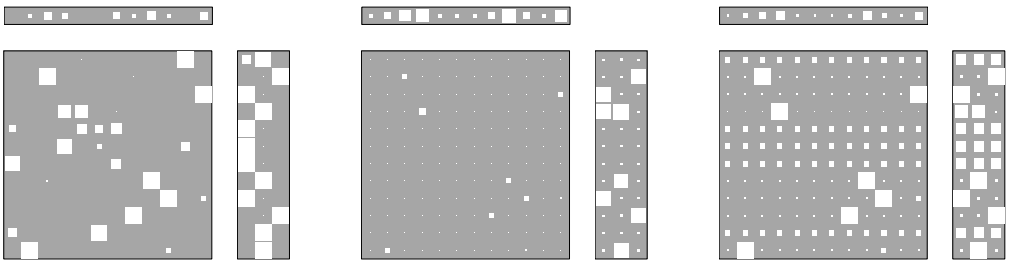


(c) ML: plot of the derivative of the log likelihood in (a).



(d) VB: plot of the derivative of the lower bound in (b).

Figure 3.2: Training ML and VB hidden Markov models on synthetic sequences drawn from $(abc)^*$, $(acb)^*$ and $(a^*b^*)^*$ grammars (see text). Subplots **(a)** & **(c)** show the evolution of the likelihood of the data in the maximum likelihood EM learning algorithm for the HMM with $k = 12$ hidden states. As can be seen in subplot (c) the algorithm converges to a local maximum after by about 296 iterations of EM. Subplots **(b)** & **(d)** plot the marginal likelihood lower bound $\mathcal{F}(q(\mathbf{s}_{1:T}), q(\boldsymbol{\theta}))$ and its derivative, as a *continuation* of learning from the point in parameter space where ML converged (see text) using the variational Bayes algorithm. The VB algorithm converges after about 29 iterations of VBEM.



(a) ML state prior $\boldsymbol{\pi}$, transition A and emission C probabilities.

(b) VB variational posterior parameters for $q(\boldsymbol{\pi})$, $q(A)$ and $q(C)$.

(c) Variational posterior mean probabilities: $\langle q(\boldsymbol{\pi})\rangle$, $\langle q(A)\rangle$ and $\langle q(C)\rangle$.

Figure 3.3: **(a)** Hinton diagrams showing the probabilities learnt by the ML model, for the initial state prior $\boldsymbol{\pi}$, transition matrix A, and emission matrix C. **(b)** Hinton diagrams for the analogous quantities $\mathbf{u}^{(\pi)}$, $\mathbf{u}^{(A)}$ and $\mathbf{u}^{(C)}$, which are the variational parameters (counts) describing the posterior distributions over the parameters $q(\boldsymbol{\pi})$, $q(A)$, and $q(C)$ respectively. **(c)** Hinton diagrams showing the mean/modal probabilities of the posteriors represented in (b), which are simply row-normalised versions of $\mathbf{u}^{(\pi)}$, $\mathbf{u}^{(A)}$ and $\mathbf{u}^{(C)}$.

of the learnt models of forwards and backwards character sequences. As discussed above in section 3.4.2, computing the predictive probability for VB is intractable, and so we approximate the VB solution with the model at the mean of the variational posterior given by equations (3.54–3.59).

We used sentences taken from Lewis Carroll's *Alice's Adventures in Wonderland*. All punctuation was removed to leave 26 letters and the blank space (that is to say $p = 27$). The training data consisted of a maximum of 32 sentences (of length between 10 and 100 characters), and the test data a fixed set of 200 sentences of unconstrained length. As an example, the first 10 training sequences are given below:

(1) 'i shall be late '

(2) 'thought alice to herself after such a fall as this i shall think nothing of tumbling down stairs '

(3) 'how brave theyll all think me at home '

(4) 'why i wouldnt say anything about it even if i fell off the top of the house '

(5) 'which was very likely true '

(6) 'down down down '

(7) 'would the fall never come to an end '

(8) 'i wonder how many miles ive fallen by this time '

(9) 'she said aloud '

(10) 'i must be getting somewhere near the centre of the earth '

ML, MAP and VB hidden Markov models were trained on varying numbers of sentences (sequences), n, varying numbers of hidden states, k, and for MAP and VB, varying prior strengths, u_0, common to all the hyperparameters $\{\mathbf{u}^{(\pi)}, \mathbf{u}^{(A)}, \mathbf{u}^{(C)}\}$. The choices were:

$$n \in \{1, 2, 3, 4, 5, 6, 8, 16, 32\}, \quad k \in \{1, 2, 4, 10, 20, 40, 60\}, \quad u_0 \in \{1, 2, 4, 8\} \; . \tag{3.86}$$

The MAP and VB algorithms were initialised at the ML estimates (as per the previous experiment), both for convenience and fairness. The experiments were repeated a total of 10 times to explore potential multiple maxima in the optimisation.

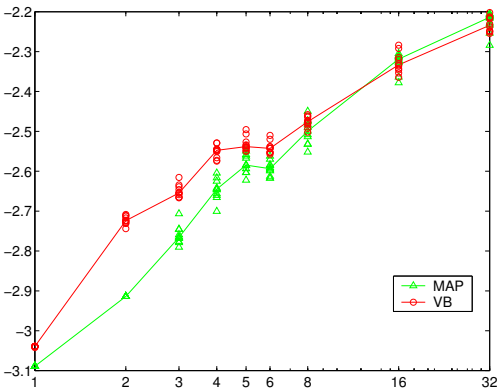
In each scenario two models were learnt, one based on forwards sentences and the other on backwards sentences, and the discrimination performance was measured by the average fraction of times the forwards and backwards models correctly classified forwards and backwards test sentences. This classification was based on the log probability of the test sequence under the forwards and backwards models learnt by each method.

Figure 3.4 presents some of the results from these experiments. Each subplot is an examination of the effect of one of the following: the size of the training set n, the number of hidden states k, or the hyperparameter setting u_0, whilst holding the other two quantities fixed. For the purposes of demonstrating the main trends, the results have been chosen around the canonical values of $n = 2$, $k = 40$, and $u_0 = 2$.

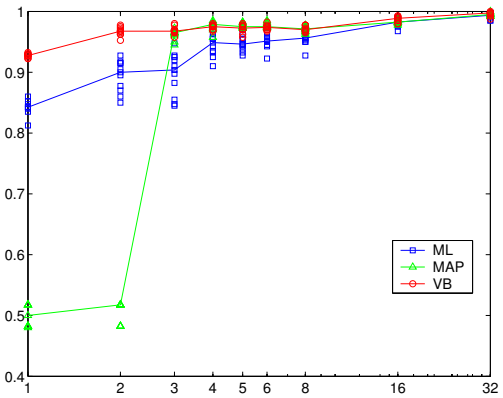
Subplots **(a,c,e)** of figure 3.4 show the average test log probability *per symbol* in the test sequence, for MAP and VB algorithms, as reported on 10 runs of each algorithm. Note that for VB the log probability is measured under the model at the mode of the VB posterior. The plotted curve is the median of these 10 runs. The test log probability for the ML method is omitted from these plots as it is well below the MAP and VB likelihoods (qualitatively speaking, it increases with n in **(a)**, it decreases with k in **(c)**, and is constant with u_0 in **(e)** as the ML algorithm ignores the prior over parameters). Most importantly, in **(a)** we see that VB outperforms MAP when the model is trained on only a few sentences, which suggests that entertaining a distribution over parameters is indeed improving performance. These log likelihoods are those of the forward sequences evaluated under the forward models; we expect these trends to be repeated for reverse sentences as well.

Subplots **(b,d,f)** of figure 3.4 show the fraction of correct classifications of forwards sentences as forwards, and backwards sentences as backwards, as a function of n, k and u_0, respectively.

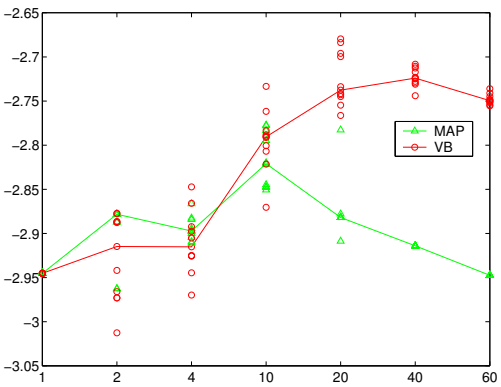
We see that for the most part VB gives higher likelihood to the test sequences than MAP, and also outperforms MAP and ML in terms of discrimination. For large amounts of training data n, VB and MAP converge to approximately the same performance in terms of test likelihood and discrimination. As the number of hidden states k increases, VB outperforms MAP considerably, although we should note that the performance of VB also seems to degrade slightly for $k >$



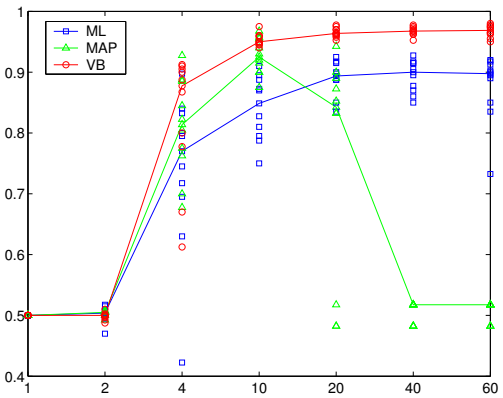
(a) Test log probability per sequence symbol: dependence on n. With $k = 40,\ u_0 = 2$.



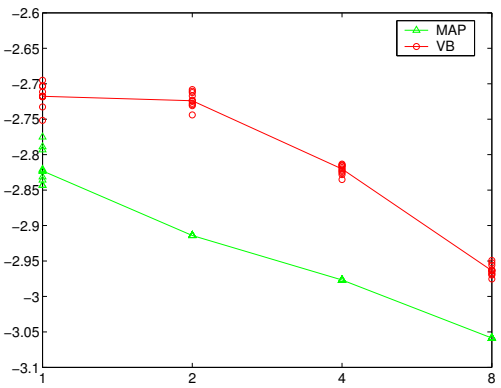
(b) Test discrimination rate dependence on n. With $k = 40,\ u_0 = 2$.



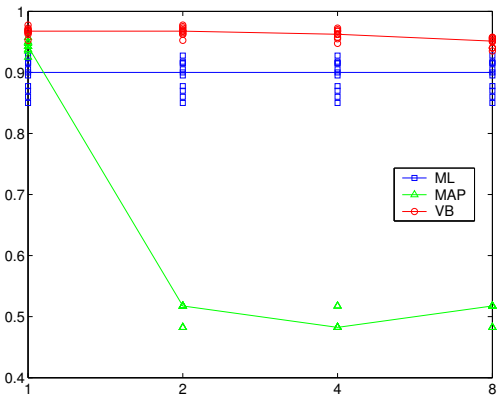
(c) Test log probability per sequence symbol: dependence on k. With $n = 2,\ u_0 = 2$.



(d) Test discrimination rate dependence on k. With $n = 2,\ u_0 = 2$.



(e) Test log probability per sequence symbol: dependence on u_0. With $n = 2,\ k = 40$.



(f) Test discrimination rate dependence on u_0. With $n = 2,\ k = 40$.

Figure 3.4: Variations in performance in terms of test data log predictive probability and discrimination rates of ML, MAP, and VB algorithms for training hidden Markov models. Note that the reported predictive probabilities are per test sequence symbol. Refer to text for details.

20. This decrease in performance with high k corresponds to a solution with the transition matrix containing approximately equal probabilities in all entries, which shows that MAP is over-regularising the parameters, and that VB does so also but not so severely. As the strength of the hyperparameter u_0 increases, we see that both the MAP and VB test log likelihoods decrease, suggesting that $u_0 \leq 2$ is suitable. Indeed at $u_0 = 2$, the MAP algorithm suffers considerably in terms of discrimination performance, despite the VB algorithm maintaining high success rates.

There were some other general trends which were not reported in these plots. For example, in **(b)** the onset of the rise in discrimination performance of MAP away from .5 occurs further to the right as the strength u_0 is increased. That is to say the over-regularising problem is worse with a stronger prior, which makes sense. Similarly, on increasing u_0, the point at which MAP begins to decrease in **(c,d)** moves to the left. We should note also that on increasing u_0, the test log probability for VB **(c)** begins to decrease earlier in terms of k.

The test sentences on which the algorithms tend to make mistakes are the shorter, and more reversible sentences, as to be expected. Some examples are: 'alas ', 'pat ', 'oh ', and 'oh dear '.

3.6 Discussion

In this chapter we have presented the ML, MAP and VB methods for learning HMMs from data. The ML method suffers because it does not take into account model complexity and so can overfit the data. The MAP method performs poorly both from over-regularisation and also because it entertains a single point-parameter model instead of integrating over an ensemble. We have seen that the VB algorithm outperforms both ML and MAP with respect to the likelihood of test sequences and in discrimination tasks between forwards and reverse English sentences. Note however, that a fairer comparison of MAP with VB would include allowing each method to use cross-validation to find the best setting of their hyperparameters. This is fairer because the effective value of u_0 used in the MAP algorithm changes depending on the basis used for the optimisation.

In the experiments the automatic pruning of hidden states by the VB method has been welcomed as a means of inferring useful structure in the data. However, in an ideal Bayesian application one would prefer all states of the model to be active, but with potentially larger uncertainties in the posterior distributions of their transition and emission parameters; in this way all parameters of the model are used for predictions. This point is raised in MacKay (2001) where it is shown that the VB method can inappropriately overprune degrees of freedom in a mixture of Gaussians.

Unless we really believe that our data was generated from an HMM with a finite number of states, then there are powerful arguments for the Bayesian modeller to employ as complex a

model as is computationally feasible, even for small data sets (Neal, 1996, p. 9). In fact, for Dirichlet-distributed parameters, it is possible to mathematically represent the limit of an infinite number of parameter dimensions, with finite resources. This result has been exploited for mixture models (Neal, 1998b), Gaussian mixture models (Rasmussen, 2000), and more recently has been applied to HMMs (Beal et al., 2002). In all these models, sampling is used for inferring distributions over the parameters of a countably infinite number of mixture components (or hidden states). An area of future work is to compare VB HMMs to these infinite HMMs.