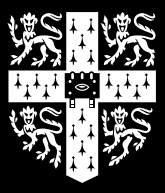


UNIVERSITY OF CAMBRIDGE

Bayesian Nonparametric Hidden Markov Models

Jurgen Van Gael

B.Sc. Catholic University of Leuven (2005)
M.Sc., University of Wisconsin Madison, (2007)

Wolfson College
University of Cambridge

THESIS

Submitted for the degree of
Doctor of Philosophy, University of Cambridge

2011

I hereby declare that my dissertation, entitled "Bayesian Nonparametric Hidden Markov Models", is not substantially the same as any that I have submitted for a degree or diploma or other qualification at any other university. No part of my dissertation has already been, or is concurrently being, submitted for any degree, diploma, or other qualification. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and acknowledgements. This dissertation does not exceed sixty thousand words in length.

for Noah & Lien

Abstract

The Bayesian approach to statistical modelling is a consistent and intuitive framework for dealing with uncertainty about the world. In this approach, we encode any prior knowledge about variables (observed or unobserved) with the goal of inferring a posterior distribution over unobserved variables. The most common approaches to Bayesian modelling to date are the so-called parametric Bayesian models: these are specified with a finite number of unobserved variables. With vast amounts of data readily available today, these models generally fail to leverage a learning opportunity: no additional structure beyond that which was defined in the prior can be learned. Any increase in data passed into the model will only affect the accuracy of the inferred posteriors. Non-parametric Bayesian models address this problem: they are probabilistic models whose additional flexibility allows for learning the structure of complex datasets.

In this thesis we present new models and inference algorithms for non-parametric Bayesian models in the context of hidden Markov models. Our contribution is three-fold: we introduce for the first time, a family of algorithms for efficient and exact Monte Carlo inference in non-parametric Bayesian Markov models. Secondly, we apply non-parametric Bayesian hidden Markov models to the part-of-speech tagging problem in natural language processing. Thirdly, we introduce a new family of non-parametric Bayesian hidden Markov models with a factorial latent Markov chain structure.

More specifically, in chapter 1 we motivate nonparametric Bayesian models using a simple mixture model example and give an overview of the literature on Bayesian approaches to hidden Markov modelling. Chapter 2 presents an overview of the foundations for Bayesian non-parametric modelling by introducing a number of fundamental and well understood Bayesian non-parametric building blocks.

Using the building blocks introduced in chapter 2, chapter 3 describes a non-parametric extension to the hidden Markov model, called the infinite hidden Markov model (iHMM) and introduces a family of fast and exact Monte Carlo inference algorithms for this model. We also present an overview of extensions for the iHMM which exist in the literature while introducing some new ones.

Chapter 4 presents a case study on the iHMM in the area of natural language processing. In particular, we look at the task of unsupervised part-of-speech tagging. We compare the non-parametric Bayesian approach against its parametric counterpart and introduce an alternative way of evaluating any unsupervised part-of-speech tagger.

Our final chapter 5 introduces a new Bayesian non-parametric building block called the Markov IBP which we then use to build a non-parametric extension of the factorial hidden Markov model, called the infinite factorial hidden Markov model (iFHMM). We apply this model to the well-known cocktail party problem, where we separate the audio from an arbitrary number of speakers using a limited number of microphones.

Given the important role of hidden Markov models in time series and sequence modeling, and the flexibility of nonparametric approaches, there is great potential for many future applications and extensions of non-parametric Bayesian hidden Markov models.

Acknowledgements

The most important person to thank is my PhD advisor Zoubin Ghahramani. He has inspired me every step of the way by being a great mentor, teacher and collaborator. I am also extremely grateful for having spent three fantastic years in the machine learning group at the University of Cambridge, both Zoubin and Carl have made this an extremely stimulating environment. I want to say thanks to all fellow students and visitors at CBL for great collaborations and friendships. I owe a lot to Jerry Zhu for advising me during the first two years of graduate research.

I have been blessed to have been able to work with great collaborators people; I'd like to thank Yee Whye, Yunus, Finale, Kurt, David, Andreas, Sebastien, Jerry, Andrew, David, Mark and Burr.

Microsoft Research has supported me financially and I am very grateful for that. During my PhD I have been fortunate to visit Microsoft Research Cambridge twice for internships. I'd like to thank Ralf and Thore for letting me explore out-of-the-box ideas and David, Ulrich, Allen, Stuart, Giuseppe, Joaquin and Gjergji for being a great team.

This thesis would've probably never happened if it weren't for Thomas showing me the way - thanks! Being abroad for five years means missing out on lots of fun times at home; it was always great to be back with Paul and Comic Action Girl to have some old school fun once in a while.

I'd also like to thank mama en papa for supporting me every step of the way; even when things looked a bit bleak. Last but by no means least I want to dedicate this work to my partner Lien. Your love and support made this three year journey feel like a holiday. Thanks!

Contents

List of algorithms

Chapter 1

Introduction

In the last twenty years, business, government and individuals have been producing data at an accelerating pace. It appears that Moore's law applies to data: every five years, the amount of digital information increases tenfold (Eco, 2010). At the same time, advances at the intersection of computer science and statistics enable us to analyse this data deluge. The result of this effort is a wealth of data driven services such as machine translation, speech recognition, search engines and many more.

This revolutionary change in the amount of data available for statistical analysis suggests an equally dramatic change in the statistical tools we use. In 1897, it took J.J. Thomson months of experimentation (Dahl, 1997) to generate the data for estimating the mass-to-charge ration of the electron. Simple descriptive statistics were sufficient to make a valuable scientific contribution. Today, the Large Hadron Collider at CERN produces 15 peta bytes of data annually (CERN, 2010). Such a vast amount of data offers the opportunity for accurately uncovering lots of hidden structure and patterns.

One useful tool we can use to analyse large data sets are the so called *nonparametric models*. Most statistical models are designed with a fixed number of parameters which we learn from data, we call them *parametric models*. For many interesting tasks, parametric models can typically be learnt well with a modest amount of data. Training the model with more data only leads to an increase in the number of accurate significant digits of the learnt parameters. Nonparametric models adapt the number of parameters in the model to the complexity of the data. Nonparametric models are not just theoretical statistical constructions but are increasingly being used in large scale applied data analysis solutions (Halevy et al., 2009).

In this thesis we investigate and propose a new set of nonparametric models for sequential data: more specifically, we study *Bayesian Nonparametric Hidden Markov Models*. In the next three sections, we carefully explain and motivate each of the elements of the title of this thesis: "Bayesian", "Nonparametric" and "Hidden Markov Models".

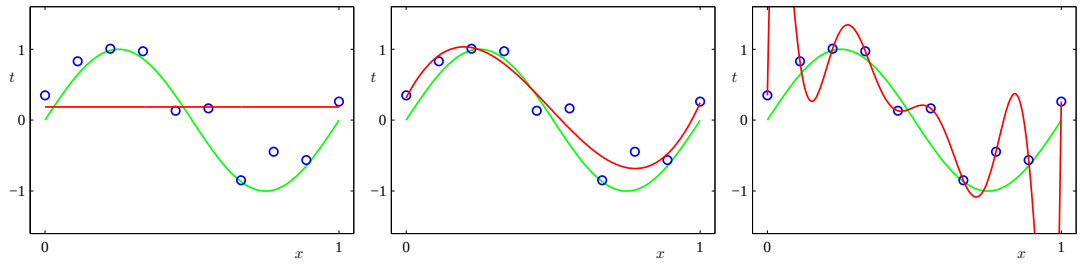
1.1 Bayesian Models

A major concern in all statistical analyses is the problem of generalisation: finding the sweet spot in the spectrum between simple models which might under fit and complex models which can over fit. Under fitting is the problem where a model is too constrained and doesn't learn from data points. Over fitting is the problem when a model starts to memorise rather than generalise the training data. We illustrate both phenomena using a regression example.

Consider the problem of fitting a polynomial to N data points x_n, y_n; we will denote with $\{x_n\}$ the set of all inputs and $\{y_n\}$ the set of all outputs. If we parameterize the polynomial or order D using parameter $w \in \mathcal{R}^D$, we can construct the following probabilistic model

$$p(\{y_n\}|\{x_n\}, w) = \prod_{n=1}^{N} \left(\text{Normal}\left(y_n; \sum_{d=0}^{D} x_n^d w_d, 1 \right) \right). \tag{1.1}$$

We can fit the parameter w using a maximum likelihood estimator. Figure 1.1 illustrates three polynomials of different order which are fit to 10 data points randomly sampled from the function $f(x) = \sin(2\pi x)$. From this figure it is clear that the $D = 0$ polynomial under fits the data: the model is too constrained and can only model the mean $\sum_n y_n/N$. Also note that the $D = 9$ polynomial over fits the data: the polynomial interpolates (or memorises) the data points and hence makes bad predictions for all points other than the observations.



(a) $D = 0$ (b) $D = 3$ (c) $D = 9$

Figure 1.1: Plots of polynomial regression of various orders D. The red line represents the model fit whereas the green line represents ground truth. With permission (Bishop, 2006).

There are various ways of finding the right tradeoff between under fitting and over fitting. A common Bayesian approach consists of treating the model complexity as just another unknown quantity we want to learn from data. A Bayesian analysis of the polynomial fitting model resolves the generalisation problem by introducing a prior on

the parameters $\{w_d\}$ and computing the marginal likelihood or evidence

$$p(\{y_n\}|D) = \int_{\{w_d\}} \left(\prod_{n=1}^{N} p(\{y_n\}|\{x_n\}, w) \right) \prod_{d=1}^{D} p(w_d). \tag{1.2}$$

The evidence computes the probability of the data under polynomial regression with degree D with all parameters integrated out. Integrating out the parameters is crucial as it weighs each value of the likelihood $p(\{y_n\}|\{x_n\}, w)$ by its plausibility under the prior $p(w_d)$. In other words, by integrating over unknown variables, the posterior distribution will concentrate around parameter settings which give high probability *on average*. If we let the prior on w be a multivariate normal distribution with zero mean and covariance Σ_0, then the log marginal likelihood is

$$\log p(y|X, \Sigma_0) \propto -0.5 \log |I + X\Sigma_0 X^T| + 0.5 y^T X (\Sigma_0^{-1} + X^T X)^{-1} X^T y, \tag{1.3}$$

where $X_{nd} = x_n^d$. We now consider two specific choices for the covariance matrix Σ_0.

A first proposal for the prior on $\{w_d\}$ would be to choose $p(w_d) = \text{Normal}(w_d; 0, 1)$, or $\Sigma_0 = I$. The left plot in figure 1.2 shows the marginal likelihood for different model complexities, which in our case corresponds to different polynomial degrees. The plot peaks at $D = 3$ which indicates that a third order polynomial is the most likely to have generated the data. This model optimally trades off prior and likelihood. Any model corresponding to a higher order polynomial would lead to a better fit, e.g. the $D = 9$ plot in figure 1.1, but would be less likely under the prior. Any model corresponding to a lower order polynomial is more likely under the prior but has a bad fit, or low likelihood.

The method of finding an optimal model complexity by maximizing the marginal likelihood is very common and in the Bayesian literature often refered to as *model selection*.

On second consideration, the prior $p(w_d) = \text{Normal}(w_d; 0, 1)$ induces erratic polynomials when D becomes large. A draw from this prior would most likely have non-neglegible higher order terms: for polynomials, this means very rapidly changing functions. In our setting it is arguably not desirable to generate these erratic functions for large D. Our second experiment will explicitly encode that higher order terms have small magnitude: $p(w_d) = \text{Normal}\left(w_d; 0, \frac{1}{2^d}\right)$. The right plot in figure 1.2 illustrates the evidence for different model complexities. In this case we observe very different behavior: the data supports increasingly large models! The variance of the higher order terms increases fast enough so that they do not penalize models of high order. More data can always overrule the prior; hence, although this model doesn't penalize higher orders it has the flexibility to use higher order terms if there is data to support it.

This leads us to the core idea of nonparametric models: by choosing an appropriate prior, we can let the number of parameters increase to infinity so we don't under fit the

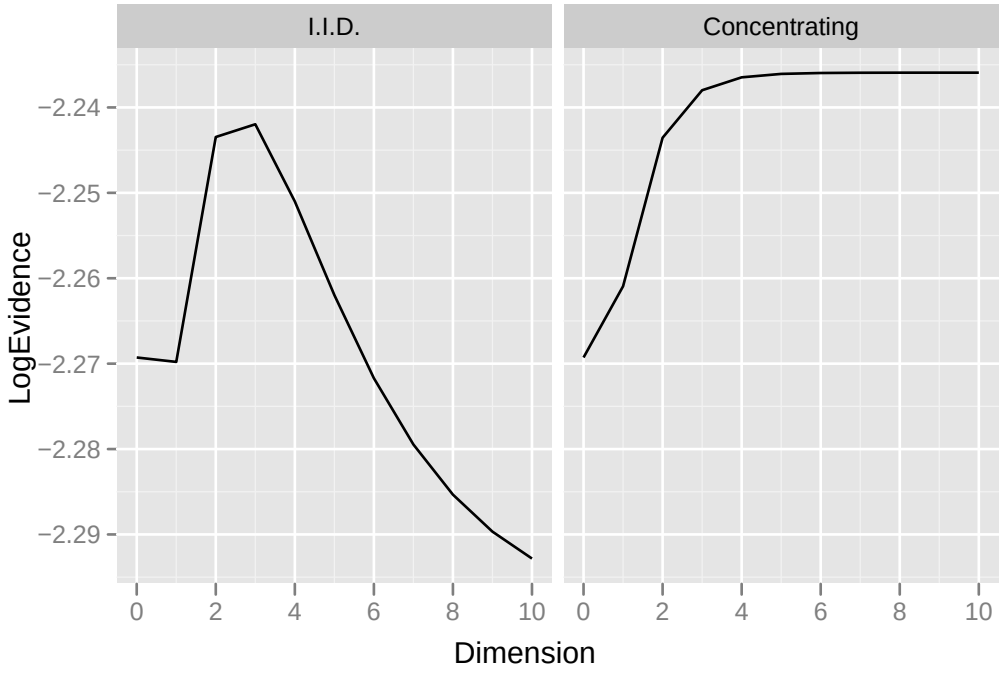


Figure 1.2: Marginal likelihood for the Bayesian treatment of polynomial regression. In the left plot, we use an i.i.d. prior on the $\{w_n\}$, in the right plot, we choose a prior where higher order w_n have lower variance a priori.

data while at the same time use Bayesian methods to prevent over fitting the data. In the following section we will illustrate this idea again using Gaussian mixture models.

It is important to add that there are many other ways of preventing over- and under fitting for both parametric and non-parametric models: e.g. cross-validation, bootstrapping, regularisation, etc (Hastie et al., 2003). Although these are of great value to the data analysis community, they are often ad-hoc additions to existing models. The Bayesian paradigm integrates both the description of the model as well as the method for preventing over fitting into one and the same paradigm: probability theory. For this reason we believe our search for Bayesian nonparametric models is a promising area of research.

1.2 Bayesian Nonparametric Models

When modelling complex distributions, it is hard to - a priori - choose a good complexity for a parametric model. Choosing a parametric model that is too simple can introduce under fitting whereas choosing a parametric model that is too complex can result in either over fitting or wasted computational cycles. In the previous section, we discussed the possibility of evaluating several models of increasing complexity and then choosing the optimal one according to some criterion. In this section we illustrate the advantages

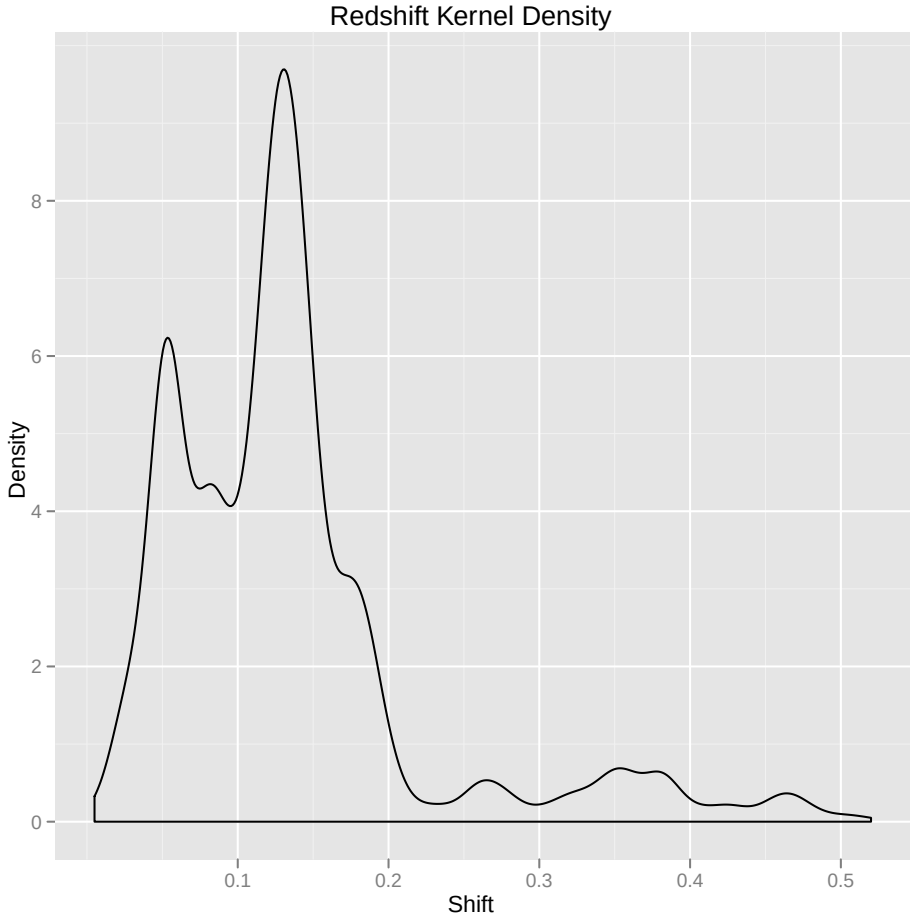


Figure 1.3: A kernel density plot of the redshift data set from Wasserman (2006). We used a Gaussian kernel where the bandwidth was chosen using the default *nrd0* method in R. The kernel density estimate illustrates that the data is multi modal with each mode having a different width.

of using a Bayesian *nonparametric* approach to data modelling.

We analyse the one dimensional Sloan Digital Sky Survey data set from Wasserman (2006). This data set consists of 1266 data points, each of which represents the redshift of a galaxy. There is astrophysical evidence that a peak in the density of the data set corresponds to a galaxy cluster. Hence we want to learn about the number of separate modes in the density. The individual data points and a kernel density estimator of the data set are illustrated in figure 1.3. Although the kernel plot might be sufficient to estimate the number of modes *for this simple data set*, we want to analyse the data using a Bayesian model as an example of what could be done in a more complex Bayesian analysis.

We use a very simple yet widely used Bayesian probabilistic model as our density estimator: the normal mixture model. A normal mixture model assumes that a set of

$$
\begin{aligned}
\pi &\sim \mathsf{Dirichlet}(\gamma_0), \\
\forall k \in \{1 \cdots K\}, & \\
\mu_k, \tau_k &\sim \mathsf{NormalGamma}(\mu_0, \lambda_0, \alpha_0, \beta_0), \\
\forall n \in \{1 \cdots N\}, & \\
c_n|\pi &\sim \mathsf{Discrete}(\pi), \\
x_n|\mu, \tau, c_n &\sim \mathsf{Normal}(\mu_{c_n}, \tau_{c_n})
\end{aligned}
$$

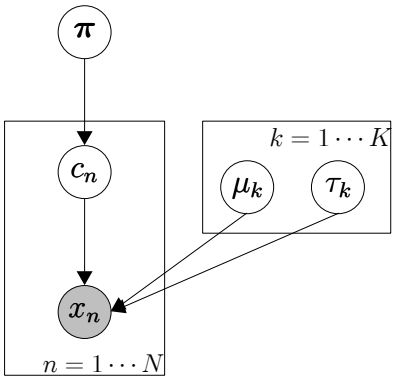


Figure 1.4: Graphical Model for a normal Mixture Model

data points $x_1, x_2, \cdots, x_N$ is generated as follows: first we choose a fixed number of clusters K. Then for each of K clusters we chose a mean μ_k and a precision τ_k from a NormalGamma prior distribution. Moreover, each cluster is assigned a weight π_k with $\pi \sim \mathsf{Dirichlet}(\gamma_0)$, a K-dimensional Dirichlet distribution. Finally, each data point x_n is generated by first drawing a cluster assignment $c_n \sim \pi$ and then drawing the data point x_n from a normal distribution with mean μ_{c_n} and precision τ_{c_n}. Figure 1.4 illustrates the graphical model for the normal mixture model.

An analytical solution to the posterior is intractable but we can approximate it by computing samples from the posterior distribution using a collapsed Gibbs sampler (Neal, 1991, Rasmussen, 2000). The posterior distribution takes the form

$$
p(\{c_n\}|\{x_n\}, \mu_0, \lambda_0, \alpha_0, \beta_0, \gamma_0) \propto \int d\mu d\tau d\pi \prod_{n=1}^{N} \left(\pi_{c_n} \mathsf{Normal}(x_n; \mu_{c_n}, \tau_{c_n})\right). \tag{1.4}
$$

The collapsed Gibbs sampler produces these samples by re-sampling each c_n individually keeping the other $c_{\neg n}$ fixed. More specifically, using Bayes rule and the marginalisation result in (Paquet, 2007, Appendix A.6)

$$
\begin{aligned}
& p(c_n|\{c_{\neg n}\}, \{x_n\}, \mu_0, \lambda_0, \alpha_0, \beta_0, \gamma_0) \\
& \propto p(c_n|c_{\{\neg n\}}, \gamma_0) p(x_n|\{x_i\}_{i:c_i=c_n}, \mu_0, \lambda_0, \alpha_0, \beta_0) \\
& = \left(\int d\pi p(c_n|\pi) p(\pi|c_{\neg n}, \gamma_0)\right) \left(\int d\mu d\tau \mathsf{Normal}(x_n; \mu, \tau) p(\mu, \tau|\{x_i\}_{i:c_i=c_n}, \mu_0, \lambda_0, \alpha_0, \beta_0)\right) \\
& = \frac{m_{c_n} + \gamma_0/K}{N - 1 + \gamma_0} \\
& \quad \mathsf{StudentT}\left(\frac{\lambda_0\mu_0 + m_{c_n}\mu_{ml}}{\lambda_0 + m_{c_n}}, \frac{(\lambda_0 + 1)\beta + \sum_{i:c_i=c_n}(x - \mu_{ml})^2 + \frac{\lambda_0 m_{c_n}\mu_{ml}^2}{\lambda_0 + m_{c_n}}}{\lambda_0(\alpha + m_{c_n})}, 2.0(\alpha + m_{c_n})\right)
\end{aligned} \tag{1.5}
$$

where m_k is the number of data points in cluster k, $m_k = |\{i : c_i = k\}|$ and μ_{ml} is the maximum likelihood estimate of the cluster mean parameter $\mu_{ml} = \sum_{i:c_i=c_n} x_i/(|\{i : c_i = c_n\}|)$. A key property of the collapsed Gibbs sampler is that it integrates out the parameters μ_{c_n}, τ_{c_n} and only re-samples the cluster assignments c_n.

We chose the hyper parameters of the mixture to be $\gamma_0 = 1/K$. This corresponds to choosing a Dirichlet prior which puts more mass on low entropy configurations for π; see appendix A for a detailed explanation of this property of the Dirichlet distribution. This choice of prior will encourage the model to use as few clusters as possible to model the data. For the cluster mean and precision hyper parameters we chose $\mu_0 = 0, \lambda_0 = 1, \alpha_0 = 1$ and $\beta_0 = 1$ to model our initial belief that clusters are zero centred with a variance around 1. This choice of prior for the cluster parameters has the flexibility for generating very tight clusters (if there is enough data to increase the posterior cluster precision) as well as generating very wide clusters.

Recall that the scientific question we need to answer is: how many galaxies are represented in our data set. For that, we need to measure how many modes, or clusters there are in our data set. A priori, we don't know how many clusters are sufficient to model the data set. Hence, we run multiple experiments, varying the number of clusters $K \in \{10, 12, 14, 16, 19, 20\}$ in each run and for each posterior sample we generate, count how many occupied clusters we find. Hence, for each posterior sample we compute the cluster occupancy: i.e. how many clusters with any data points in them are there? Figure 1.5 shows the cluster occupancy histograms for different values of K.

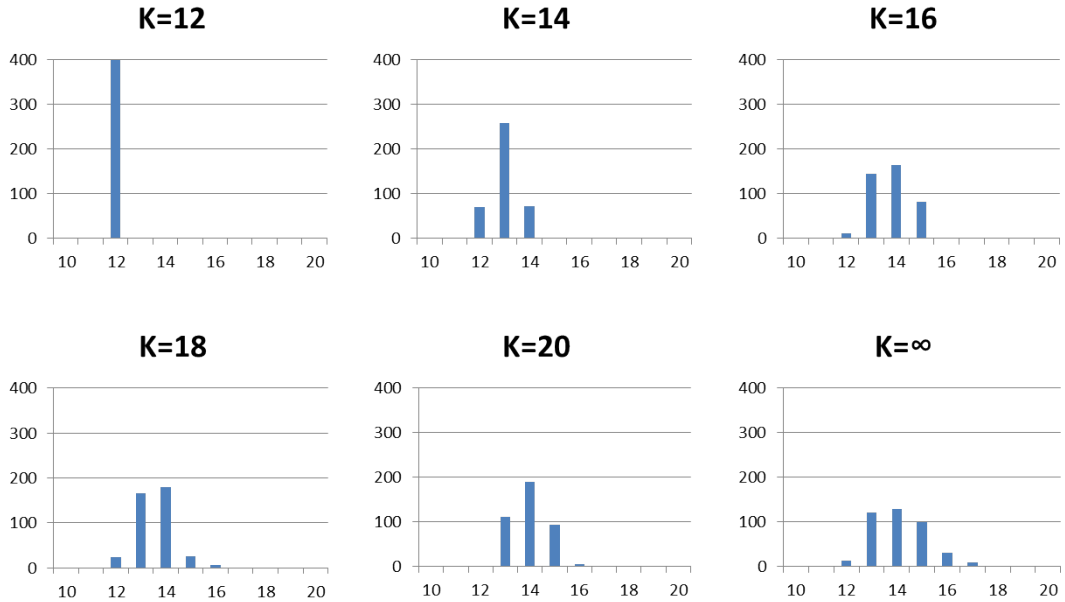


Figure 1.5: The cluster occupancy histograms for Gibbs iteration 100 to 500 for $K \in \{10, 12, 14, 16, 19, 20\}$.

It is clear that when K is small, the parametric model needs all clusters to model the data properly. As K grows, the mixture model becomes more complex and not all clusters are necessary to fit the data. The plots suggest that about 14 clusters are needed to explain the data well, but some uncertainty regarding the number of clusters remains. Although the Bayesian approach "prunes" out unnecessary clusters when K is large, we pay a computational cost for doing so. At $K = 20$ the collapsed Gibbs sampler will need to evaluate for each data point whether it needs to be assigned to each of the $K = 20$ clusters. Since all empty clusters are indistinguishable, a computationally more efficient approach would not consider each empty cluster separately.

Disregarding computational efficiency, we might wonder whether we need to specify the maximum number of clusters beforehand. For a complex data set, we might have no idea for any reasonable value for the maximum. Can we just make our mixture model infinitely large by letting $K \to \infty$? The naive approach to doing so fails miserably: when the number of potential clusters $K \to \infty$ the distribution over the mixture distribution π becomes "too sparse": all but one entry get zero mass. In other words, in the limit $K \to \infty$ the Dirichlet prior on π degenerates to a point mass of probability 1 on a random dimension.

Interestingly enough, following (Neal, 1991, Rasmussen, 2000) we can compute the distribution over the cluster assignments when we integrate out the "misbehaving" variable π. Because of conjugacy between the Dirichlet and the Discrete distribution, we can analytically find

$$\begin{aligned} p(c|\hat{\gamma}_0) &= \int \left(\prod_{n=1}^{N} p(c_n|\pi) \right) p(\pi|\hat{\gamma}_0) d\pi \\ &= \int \left(\prod_{k=1}^{K} \pi_k^{m_k} \right) \left(\frac{\Gamma(\hat{\gamma}_0)}{\Gamma(\frac{\hat{\gamma}_0}{K})^K} \prod_{k=1}^{K} \pi_k^{\hat{\gamma}_0/K-1} \right) d\pi \\ &= \frac{\prod_{k=1}^{K} \Gamma(m_k + \frac{\hat{\gamma}_0}{K})}{\Gamma(\frac{\hat{\gamma}_0}{K})^K} \frac{\Gamma(\hat{\gamma}_0)}{\Gamma(N + \hat{\gamma}_0)} \\ &= \left(\frac{\hat{\gamma}_0}{K} \right)^{K_+} \left(\prod_{k=1}^{K_+} \prod_{j=1}^{m_k - 1} (j + \frac{\hat{\gamma}_0}{K}) \right) \frac{\Gamma(\hat{\gamma}_0)}{\Gamma(N + \hat{\gamma}_0)}, \end{aligned} \tag{1.6}$$

where m_k is defined as the number of data points in class k and K_+ is the number of classes with $m_k > 0$. If we take the limit $K \to \infty$ for equation (1.6) we notice that for every possible setting of c, $p(c|\hat{\gamma}_0) = 0$. This is not a problem since we are not interested in the exact assignment of c but only in partitions of c. E.g. if we had a data set with three data points, we do not distinguish between the class assignments $\{c_1, c_2, c_3\} = \{1, 1, 2\}$ and $\{c_1, c_2, c_3\} = \{2, 2, 1\}$: they correspond to the same partition. Hence we say that a partitioning represents an equivalence class of class assignment vectors and denote the equivalence class of assignment vector c as $[c]$.

At this point we would like to compute the distribution over equivalence classes $p([c]|\hat{\gamma}_0)$ as $K \to \infty$. The final ingredient necessary to compute this limit is to count how many class assignment vectors c are in the equivalence class $[c]$. We define $K = K_0 + K_+$ where K is the total number of classes, K_0 is the number of classes to which no data point is assigned and K_+ is (as defined above) the number of used classes. Starting from one assignment vector c in the equivalence class $[c]$, we can find all other assignment vectors by permuting the assignment indices: there are $K!$ of these permutations. We need to adjust this number by the number of permutations of the unused class assignments as we are over-counting assignments that only differ by a permutation of the unused class assignments. In other words, there are $K!/K_0!$ different class assignment vectors in equivalence class $[c]$. Finally using the mathematical results in appendix A we can compute

$$\begin{aligned}
\lim_{K\to\infty} p([c]|\hat{\gamma}_0) &= \lim_{K\to\infty} \sum_{c\in[c]} p(c) \\
&= \lim_{K\to\infty} \frac{K!}{K_0!} \left(\frac{\hat{\gamma}_0}{K}\right)^{K_+} \left(\prod_{k=1}^{K_+} \prod_{j=1}^{m_k-1} (j + \frac{\alpha}{K})\right) \frac{\Gamma(\hat{\gamma}_0)}{\Gamma(N+\hat{\gamma}_0)} \\
&= \alpha^{K_+} \left(\prod_{k=1}^{K_+} (m_k - 1)!\right) \frac{\Gamma(\hat{\gamma}_0)}{\Gamma(N+\hat{\gamma}_0)}. \qquad (1.7)
\end{aligned}$$

Equation (1.7) defines a proper distribution over partitions. In other words, if we only consider which data points belong to the same cluster, this construction allows for the number of clusters to be arbitrarily large. In section 2.1 we will show how sampling can be done for this model and how, perhaps ironically, it is more efficient than the sampler for a finite mixture model. The bottom right plot of figure 1.5 illustrates that the number of clusters for the $K \to \infty$ model are qualitatively very similar to that of the parametric model with a large K. Finally, we refer to Aitkin (2001) who more extensively evaluates various model selection techniques for the normal mixture model including likelihood, Bayesian and Bayesian nonparametric methods on the galaxy data set we used here.

We illustrated how a model with a potentially infinite number of parameters has very similar characteristics to a very large finite model. This leads to the starting point for our thesis: *some problems cannot be described by a finite number of parameters, for other problems we do not know the true generating process, in both cases infinite capacity or Bayesian nonparametric models can guard against mis-specification.* Our hope is that when enough data is available, the nonparametric model will converge to the true statistics for the data. Moreover, as we have demonstrated above, a nonparametric model might even lead to computational savings compared to a large parametric model.

In this thesis we are particularly interested in studying the properties of Bayesian nonparametrics in the context of Markov models. In the next section we give a brief

overview of the relevant literature in this area.

1.3 Bayesian Nonparametric Hidden Markov Models?

Sequential data are at the core of many statistical modelling and machine learning problems. For example, text consists of sequences of words, financial data are often sequences of prices, speech signals are represented as sequences of short term power-spectra coefficients (cepstral), proteins are sequences of amino acids, DNA are sequences of nucleotides and video is a sequence of still images. Although it is possible to directly model the relationships between subsequent elements of a time series, e.g. using auto-regressive or n-gram models, in some cases we believe the data has some underlying hidden structure. For example, the observed pixels in a video might correspond to objects, the power-spectra coefficients in a speech signal might correspond to phones, and the price movements of financial instruments might correspond to underlying economic and political events. Models that explain sequential data in terms of such underlying hidden variables can be more interpretable and have better predictive properties than models that try to directly relate observed variables.

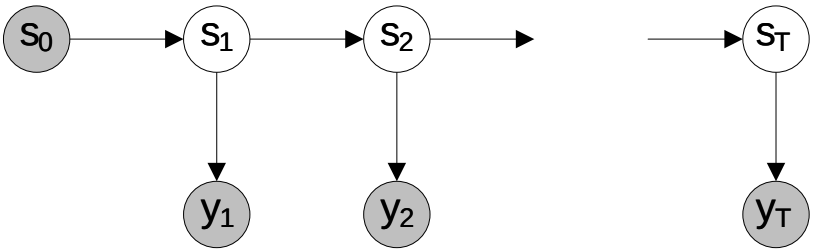


Figure 1.6: The graphical model for the hidden Markov model.

The hidden Markov model (HMM) is an influential model for sequential data that captures such hidden structure (Baum and Petrie, 1966, Baum et al., 1970, Rabiner, 1989). An HMM describes a probability distribution over a sequence of observations $y_1, y_2, \cdots, y_T$ of length T. The HMM assumes there exists a Markov chain denoted by $s_1, s_2, \cdots, s_T$ where each s_t is in one of K possible states. The distribution of the state at time t only depends on the states before it, through the state at time $t-1$ by a K by K stochastic transition matrix π, where $\pi_{ij} = P(s_t = j | s_{t-1} = i)$. This is the first-order *Markov property*, which gives the HMM its middle name. Although it is straightforward to generalise the HMM to higher orders, for simplicity we will only consider first order Markov models in this thesis. We will refer to the variable that indexes sequences as time, and assume discrete time steps. However, the models described are readily applicable to sequences indexed by any other scalar variable. Generally, we do not directly observe the Markov chain, but rather an observation y_t which only depends on an observation model

F parametrised by a state-dependent parameter θ_{s_t}. For example, if we model an object moving through a video using an HMM, we could assume that the position of the object at time t (s_t), is only dependent on its position at time $t-1$. Moreover, we don't directly observe this position but rather we observe pixels y_t whose configuration is dependent on the state at time t. We can write the probability distribution induced by the HMM as follows[1]:

$$p(y_{1:T}, s_{1:T}|K, \pi, \theta) = \prod_{t=1}^{T} p(s_t|s_{t-1})p(y_t|s_t) = \prod_{t=1}^{T} \pi_{s_{t-1},s_t} F(y_t; \theta_{s_t}). \tag{1.8}$$

Figure 1.6 shows the graphical model for the HMM. The observation model F can be made arbitrarily complex: in a natural language processing application, Gao et al. (2007) used a multinomial output distribution, Jurafsky and Martin (2000) describes how in speech recognition a normal distribution or mixture of normal distributions is commonly used.

In practise we often use the HMM in a setting where the sequence $y_{1:T}$ is given and we want to learn something about the hidden representation $s_{1:T}$, and perhaps about the parameters π, θ and K. The form of the observation model F is also important, but for this chapter we assume that F is fixed and any flexibility in F is captured by its parametrisation through θ. As an example of learning in HMMs, consider speech recognition: we can use an HMM where the hidden state sequence corresponds to phones and the observations correspond to acoustic signals. The parameters π, θ might come from a physical model of speech or be learnt from recordings of speech. Depending on how much domain knowledge is available, we distinguish three computational questions.

- π, θ, K **given.** With full knowledge of the parameters π, θ and K we only need to infer $s_{1:T}$ given the observations $y_{1:T}$. We can apply Bayes rule to equation 1.8 to find the posterior distribution over $s_{1:T}$

$$\begin{aligned} p(s_{1:T}|K, \pi, \theta, y_{1:T}) &= \frac{p(y_{1:T}, s_{1:T}|K, \pi, \theta)}{p(y_{1:T}|K, \pi, \theta)}, & (1.9) \\ &\propto \prod_{t=1}^{T} p(s_t|s_{t-1})p(y_t|s_t). & (1.10) \end{aligned}$$

  The last line follows from the fact that $p(y_{1:T}|K, \pi, \theta)$ is a constant that is independent of $s_{1:T}$. Computing this distribution can be done using a beautiful application of dynamic programming which is called the forward-backward algorithm in the context of HMM's; we review this algorithm in appendix B.

[1] To make notation more convenient, we assume that for all our time series models, all latent chains start in a dummy state that is the 1 state: e.g. for the HMM $s_0 = 1$.

- K **given,** π, θ **learnt.** If only the number of hidden states K and observations $y_{1:T}$ are known, we often want to learn the best parameters θ and π in addition to the hidden representation $s_{1:T}$. This problem is underspecified: we need a criterion to decide what the "best parameters" are. Common criteria are maximum likelihood and maximum a posteriori objectives. The former finds θ, π which maximise $p(y_{1:T}|\theta, \pi)$ while the latter introduces a prior distribution for θ, π and finds the θ, π which maximise $p(y_{1:T}|\theta, \pi)p(\theta, \pi)$. Algorithms like expectation maximization (Dempster et al., 1977) can search for the maximum likelihood and maximum a posteriori solutions but will generally only find locally optimal estimates.

- π, θ, K **learnt.** Finally, given observations $y_{1:T}$, consider the problem of discovering a statistical meaningful value for K in addition to the hidden representation $s_{1:T}$ and the other parameters π, θ. Using the maximum likelihood criterion turns out to be a bad idea as more states always lead to a better fit of the data: the nonsensical solution where $K = T$ and each state s_t has its own emission and transition parameters, maximises the likelihood. The Akaike Information Criterion (Akaike, 1974) and Bayesian Information Criterion (Schwarz, 1978) can be used to adjust the maximum likelihood estimate by penalising the number of parameters.

Another principled approach to learning π, θ or K is a fully Bayesian analysis of the model as we described in section 1.1. The Bayesian analysis treats the parameters π, θ as unknown quantities and introduces them as random variables in the model. This requires adding a prior distribution, e.g. $p(\theta|H)$ and $p(\pi|\alpha)$, and extending the full joint distribution to

$$\begin{aligned} p(y_{1:T}, s_{1:T}, \pi, \theta|K) &= p(\pi|\alpha)p(\theta|H)\left(\prod_{t=1}^{T} p(s_t|s_{t-1})p(y_t|s_t)\right) \\ &= p(\pi|\alpha)p(\theta|H)\left(\prod_{t=1}^{T} \pi_{s_{t-1},s_t} F(y_t; \theta_{s_t})\right). \end{aligned} \tag{1.11}$$

A common choice for the prior on π is to use a symmetric Dirichlet distribution on each row: if we denote with π_k the k'th row of π then $\pi_k \overset{iid}{\sim} \text{Dirichlet}(\frac{\alpha}{K}, \frac{\alpha}{K}, \cdots, \frac{\alpha}{K})$ i.i.d. for all $k \in \{1, K\}$. Similarly, a common prior on θ factorises for each state k: $\theta_k \overset{iid}{\sim} H$ i.i.d. for all $k \in \{1, K\}$, where θ_k denotes the parameter for state k. H can be any distribution but will frequently be chosen to be conjugate to the observation model F. Figure 1.7 shows the graphical model for the Bayesian analysis of the HMM.

Similar to our analysis of the regression problem in section 1.1, we can now compute the posterior distributions $p(\pi, \theta|y_{1:T}, \alpha, H)$ or $p(s_{1:T}|y_{1:T}, \alpha, H)$ by integrating over respectively $s_{1:T}$ or π, θ. Moreover, in a Bayesian analysis of the HMM we can compute the marginal likelihood or evidence $p(y_{1:T}|K) = \int p(y_{1:T}|K, \theta, \pi)p(\theta, \pi|K)$ for comparing,

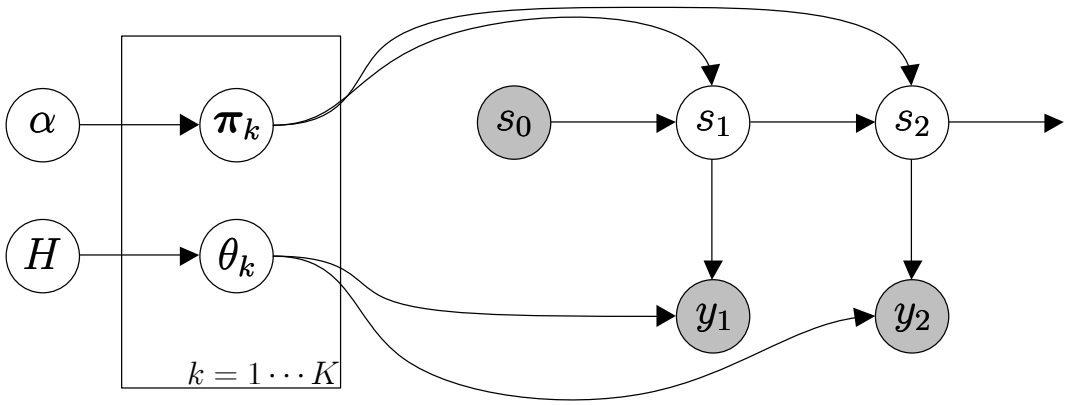


Figure 1.7: The graphical model for the Bayesian hidden Markov model.

choosing or averaging over different values of K. Unfortunately, analytically computing the marginal likelihood for an HMM is intractable. We briefly review three different methods to deal with this intractability.

- There is a large body of literature in statistics on how to use *Markov Chain Monte Carlo* (MCMC) techniques to learn the number of states in HMMs and related models, Scott (2002). We can distinguish two main approaches: MCMC methods which estimate the marginal likelihood explicitly and methods which switch between different K. Examples of the former are Annealed Importance Sampling by Neal (2001) and Bridge Sampling by Fruhwirth-Schnatter (2004) which have been successfully applied in practice. The disadvantage of these methods is that it can be computationally expensive to find an accurate estimate of the marginal likelihood for a particular K. If one needs to run the estimation procedure for each different K, the computational overhead becomes high. Reversible jump MCMC methods pioneered in Green (1995) are a family of methods which "jump" between models of different size. In the context of HMM's, Robert et al. (2000) have implemented this idea to jump between HMM models of different K.

- A very elegant approximation to the exact marginal likelihood is the approach developed by Stolcke and Omohundro (1993). Note that in the graphical model in figure 1.7, if the hidden states $s_{1:T}$ were observed, the parameters π and θ become independent and assuming that the prior and likelihood are conjugate, we can compute the marginal likelihood analytically. Stolcke and Omohundro (1993) propose to choose a good state sequence and integrating out the other parameters to compute an approximation to the marginal likelihood. They devise a state-merging algorithm based on this idea.

- A third technique to approximate the marginal likelihood is based on *variational Bayesian* (VB) inference. VB computes a lower bound on the marginal likelihood; MacKay (1997) and Beal (2003) describe VB inference algorithms that bound the marginal likelihood of an HMM. VB generalises EM as it doesn't use a point estimate of the parameters π, θ but rather an approximate posterior of these parameters. Moreover, VB also generalises the idea in Stolcke and Omohundro (1993) as it doesn't use a point estimate of the state sequence $s_{1:T}$ but rather a full distribution over these random variables.

1.4 Overview

The key problem we address in this thesis is how to deal with choosing the number of states in an HMM or any of its extensions (Bengio and Frasconi, 1995, Ghahramani and Jordan, 1997, Ghahramani and Hinton, 2000). If we have prior knowledge about the underlying physical process generating the observed sequence, and we know the number of states in that process, then we can set K to that value. For example, HMMs have been used to model ion channel currents, where it is known that the ion channel protein can be in some discrete number of physical conformations. In speech recognition, we could impose the constraint that the hidden states correspond to known phones of a language. However, in many applications the number of underlying states is not known a priori and must be inferred from the data.

In section 1.3 we reviewed several Bayesian approaches to learning the number of states for HMMs. Unfortunately, these Bayesian approaches have both statistical and computational limitations. The main statistical limitation is the assumption that a (usually small) finite number of states provides an adequate model of the sequence. In many settings, it is unlikely one can bound a priori the number of states needed. For example, if the states correspond to political and economic circumstances affecting financial variables, it's hard to say how many such discrete circumstances are needed, and to be confident that new, as yet unobserved circumstances won't arise in the future. The computational limitation is that these approaches have to compare different finite numbers of states, and each such comparison requires some method of approximating intractable marginal likelihoods.

This brings us to the main topic of the thesis: nonparametric Bayesian approaches to hidden Markov models. This topic was first introduced in Beal et al. (2002), in particular as the *infinite hidden Markov model* (iHMM). In chapter 3 we show how this model overcomes the statistical and computational limitations of the Bayesian approach to the HMM by defining a Markov chain with a countably infinite (i.e. unbounded) number of hidden states. For any finite observed sequence, only a finite number of these states

can be visited. Moreover, as the sequence length is extended and new "circumstances" arise, new states can be recruited from the unbounded pool of states just as in the nonparametric mixture model from section 1.2. Chapter 4 describes a more detailed experiment using the iHMM for a task in natural language processing called part-of-speech tagging. In chapter 5 we describe an extension of the iHMM called the infinite factorial hidden Markov model (iFHMM), first introduced in Van Gael et al. (2008a). Analogously to how the Factorial HMM extends the HMM, the iFHMM is a Bayesian nonparametric Markov model with a factorised latent state space. We conclude the thesis with a number of future research ideas. Before we embark on our journey through the world of Bayesian nonparametric Markov models, we dedicate the next chapter to an in-depth look at some Bayesian nonparametric building blocks.

Chapter 2

Nonparametric Bayesian Building Blocks

In this chapter we introduce some basic but important Bayesian nonparametric building blocks. The goal of this chapter is to provide enough theoretical foundation to build Bayesian nonparametric Markov models out of these basic components and provide insight into how we can construct efficient sampling algorithms. For each of the building blocks we describe different constructions, summarise the literature on inference and provide empirical insight into the behaviour of these distributions. A more rigorous mathematical description of the stochastic processes and random measures described in this chapter can be found in Müller and Quintana (2004), Pitman (2006), Teh (2010), Hjort et al. (2010).

In section 2.1 we introduce the *Chinese Restaurant Process* (CRP) and the related *Dirichlet Process* (DP): these distributions are commonly used as a prior for mixture models. In section 2.2 we describe an extension of the Dirichlet process called the *hierarchical Dirichlet process* (HDP) and its related combinatorial construction called the *Chinese Restaurant Franchise* (CRF). The HDP is the key construction which allows us to use the DP in hierarchical Bayesian models. This will be the main building block for the nonparametric Markov model which we describe in chapter 3. Finally, in section 2.3 we describe the *Indian Buffet Process* (IBP) and the related *Beta Process*. This distribution is used in a generalisation of finite factor models which we describe in chapter 5.

2.1 Chinese Restaurants and Dirichlet Processes

In our discussion of the finite mixture model in section 1.2 we empirically showed that it is unnecessary to constrain the number of mixture components a priori: with an appropriate choice of prior distribution, a Bayesian approach to mixture modelling automatically prunes states not needed to explain the data. By moving to partitions of the data rather than individual cluster assignments, we showed how we can safely take the infinite limit

of a finite mixture model.

In this section we relate this infinite mixture model to a Bayesian nonparametric mixing distribution. This mixing distribution comes in various flavours: a random measure called the Dirichlet Process (Ferguson, 1973), a combinatorial stochastic process called the Chinese restaurant process (Pitman, 2006) and a stick breaking construction (Sethuraman, 1994).

2.1.1 Definition and Constructions

First we introduce the combinatorial stochastic process called the *Chinese Restaurant Process* (CRP) and show how it relates to the infinite mixture model in section 1.2. Next, we introduce a second combinatorial stochastic process called the Polya Urn Scheme and relate it to the CRP. Then we show how the CRP is the marginal distribution of a random measure called the Dirichlet process (DP). We conclude the section with a description of the DP as a stick breaking construction.

The Chinese Restaurant Process.

Imagine a Chinese restaurant with an infinite number of tables. Consider the following recipe by which customers decide where to sit. The first customer entering the restaurant sits at the first table by default. The second customer enters the restaurant and sits at the first table with probability $\frac{1}{1+\alpha}$ and at table two with probability $\frac{\alpha}{1+\alpha}$. Imagine he chooses to join the first customer at the first table. The third customer enters and he chooses to sit at the first table with probability $\frac{2}{2+\alpha}$ and at the second table with probability $\frac{\alpha}{2+\alpha}$. More generally, assume n_i people are sitting at table i when the n'th customer enters the restaurant; then customer n chooses to sit at table i with probability $\frac{n_i}{n-1+\alpha}$ and at a new table with probability $\frac{\alpha}{n-1+\alpha}$. After N customers have entered the restaurant, we can interpret each table as representing a partition in a partitioning of the integers $1 \cdots N$. In other words, two people sitting at the same table are in the same partition while two people sitting at different tables are in different partitions. Figure 2.1 illustrates this process for $N = 6$. This stochastic process which defines partitions over the integers is called the Chinese restaurant process (Pitman, 2006).

A key property of this distribution is known as infinite exchangeability. We say an infinite collection of random variables $a_1, a_2, \cdots$ is infinitely exchangeable under distribution p if and only if for all integers l and all permutations σ on L, $p(a_1 = \hat{a}_1, a_2 = \hat{a}_2, \cdots, a_l = \hat{a}_l) = p(a_1 = \hat{a}_{\sigma(1)}, a_2 = \hat{a}_{\sigma(2)}, \cdots, a_l = \hat{a}_{\sigma(l)})$ where $\hat{a}_i$ are possible values for the random variables a_i. This property says that exchanging the value of any random variables will not change the probability of the configuration.

For the CRP let us denote with c_n the index of the table at which customer n is

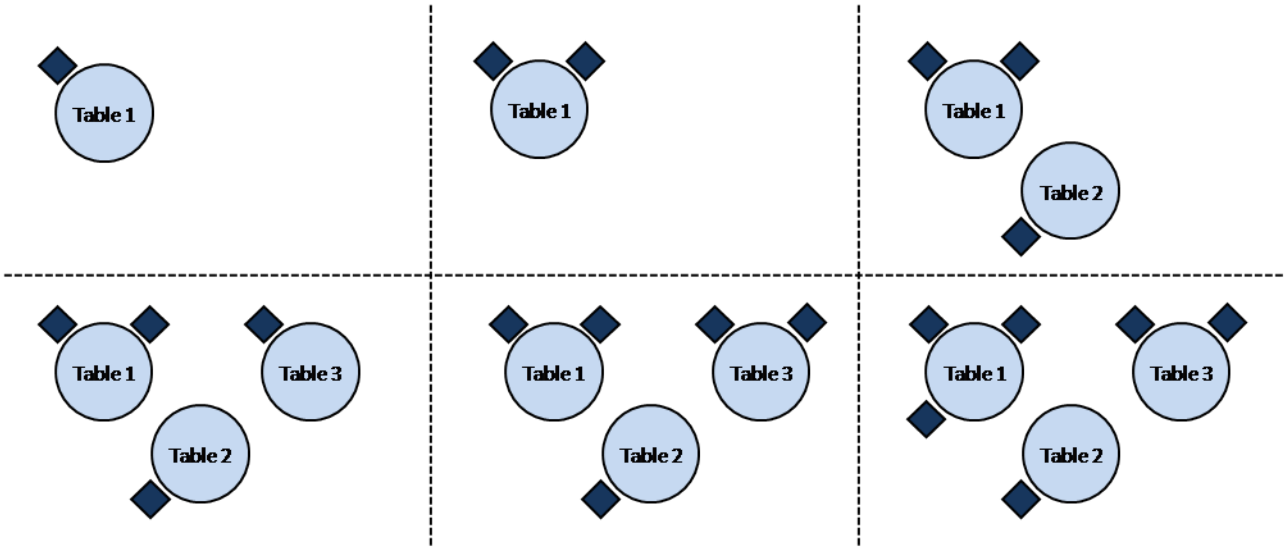


Figure 2.1: Chinese restaurant process simulation. An example Chinese restaurant process with 6 customers. The first customer always sits at the first table. The second customer sits at the first table with probability $1/1+\alpha$ and at a new table with probability $\alpha/1+\alpha$. In our example, the customer decided to join table 1. The third customer decided to sit at a new table; the probability of this happening was $\alpha/2+\alpha$. After 6 customers entered the restaurant, the induced partitioning is $[126;3;45]$ with $p([126;3;45]) = (\alpha \cdot 1 \cdot \alpha \cdot \alpha \cdot 1 \cdot 2)/(\alpha \cdot (1+\alpha) \cdot (2+\alpha) \cdot (3+\alpha) \cdot (4+\alpha) \cdot (5+\alpha))$.

seated. A simple proof by induction shows that the sequence $c_1, c_2, \cdots$ is an exchangeable sequence. Exchangeability for the CRP thus implies that the order in which people join tables has no effect on the probability of a particular partitioning. In other words, the probability of a partitioning Π_N induced by a CRP with N customers is only dependent on the number of people seated at each table.

Let us denote with n_i the number of people at table i under Π_N and let T be the total number of occupied tables. Note that the number of tables T is not a deterministic quantity: it is a random variable itself. We now compute the probability of Π_N by ordering the customers so they first fill up table 1 until n_1 people are seated, then table 2 until n_2 people are seated, etc. Then the probability distribution for the partitioning Π_n is

$$\begin{aligned} p(\Pi_N|\alpha) &= \underbrace{\frac{\alpha}{\alpha}\frac{1}{1+\alpha}\cdots\frac{n_1-1}{n_1-1+\alpha}}_{\text{Table 1}}\underbrace{\frac{\alpha}{n_1+\alpha}\frac{1}{n_1+1+\alpha}\cdots\frac{n_2-1}{n_1+n_2-1+\alpha}}_{\text{Table 2}}\cdots \\ &= \frac{\alpha^T\prod_{t=1}^{T}(n_t-1)!}{\prod_{n=0}^{N-1}(n+\alpha)} = \frac{\alpha^T\left(\prod_{t=1}^{T}(n_t-1)!\right)\Gamma(\alpha)}{\Gamma(N+\alpha)}. \end{aligned} \tag{2.1}$$

The similarity between the distribution defined by the infinite mixture model in equa-

tion (1.7) and the distribution defined by the CRP in equation (2.1) is now clear: up to a renaming of variables $n_\cdot$ with $c_\cdot$, and α with $\hat{\gamma}_0$, the formula are exactly the same. In other words, the distribution over partitions defined by the CRP is exactly the same distribution as the one defined by the infinite limit of the finite mixture model. This is the first in a series of exciting connections between seemingly different constructions which in essence define the same distribution.

Connection to the Polya Urn Scheme

The Chinese Restaurant Process is equivalent to another common stochastic process called the Polya urn scheme (Blackwell and MacQueen, 1973). Polya urn schemes are a family of algorithms which define a discrete probability distribution through the metaphor of filling an urn with coloured balls. For our purposes we consider a Polya urn scheme that is parametrised by a single real number $\alpha > 0$. We will count the total number of balls with colour i in the urn as n_i. Initially the urn is empty (all $n_i = 0$) but at each time step, with probability $\frac{n_i}{\alpha+\sum_i n_i}$ we add a ball with colour i to the urn and with probability $\frac{\alpha}{\alpha+\sum_i n_i}$ we add a ball with a new colour to the urn. Depending on the colour we chose we augment the appropriate n_i variable by one. First note that if we execute this recipe N times there will be N balls in the urn. The final number of different colours that are represented in the urn can be anything between 1 and N. Also, if there are a lot of balls with colour j in the urn, the probability of adding an extra ball with colour j is high. The parameter α controls the growth of the number of colours: if α is large with respect to $\sum_i n_i$ then it is very likely that a new colour will be added to the urn. A Polya urn scheme can be interpreted as a nonparametric prior for a clustering: each data point corresponds to a ball and each cluster to a colour. If we identify colours in the Polya urn with tables in the CRP and balls in the Polya urn with customers in the CRP, these two stochastic processes define exactly the same distribution over partitions of balls/customers. The reason we introduce the urn model is that the initial construction of a nonparametric Bayesian Markov model in chapter 3 is based on a variation of the Polya urn scheme.

Since the Polya urn scheme is so similar to the CRP, in what follows we assume the Polya urn is implicit when we mention the CRP.

The Dirichlet Process

Next we introduce the Dirichlet Process: this object is in essence a probability distribution over probability distributions. This third perspective on essentially the same distribution as the CRP will allow us to more easily extend infinite capacity models to hierarchical Bayesian models.

The Dirichlet Process (DP) (Ferguson, 1973) is a distribution over probability distributions with some very specific properties. Before we give the formal definition, let us give some intuitions. First of all since a DP is a distribution over distributions, if we draw a sample from a DP we get a distribution; we write $G \sim \mathsf{DP}$ to mean that G is a sample from a DP. Next, since distributions are over spaces of objects we are interested in, say real vectors, we must specify the space over which G is a distribution: we will use Θ to denote this space.

Since G is a distribution over Θ it must either: a) assign probability mass (possibly zero) to all the points in Θ if Θ has a countable cardinality or b) assign probability mass to all subsets[1] of Θ. This means we can ask questions like: what is the probability mass in some set $A \subset \Theta$ according to G; we write this as $G(A)$. Now we give the formal definition of a DP.

Definition 1. *Let H be a probability distribution over Θ and α be any positive real number; we say G is a draw from a Dirichlet process with concentration parameter α and base distribution H, or $G \sim \mathsf{DP}(\alpha, H)$, if and only if for any finite partition $A_1, A_2, \cdots, A_n$ of Θ, $(G(A_1), \cdots, G(A_n)) \sim \mathsf{Dirichlet}(\alpha H(A_1), \cdots, \alpha H(A_n))$.*

In other words, G is a draw from a DP if all its possible finite marginal distributions are Dirichlet distributed. It is nontrivial to show that a measure with the property above actually exists. Originally, this was shown using Kolmogorov's Consistency Theorem (Ferguson, 1973); a more recent construction is based on the normalized Gamma process (James et al., 2006).

From the DP's definition we can intuitively explain why α is called the concentration parameter and H is called the base distribution. First we consider the role of H and assume $\alpha = 1$: from the properties of the Dirichlet distribution[2] we know that $\mathbb{E}[G(A_i)] = H(A_i)$. Thus we can think of H as specifying where the mass of G is distributed, on average. Assume now that H is fixed and let $\alpha \to 0$. Since $(G(A_1), G(A_2), \cdots, G(A_n))$ is a draw from a Dirichlet distribution with very small α, we know that $(G(A_1), G(A_2), \cdots, G(A_n))$ will be sparse: one subset A_i will get most of the mass while the others tend to 0. As $\alpha \to \infty$ the distribution of $(G(A_1), G(A_2), \cdots, G(A_n))$ will get closer to $(H(A_1), H(A_2), \cdots, H(A_n))$. This rough intuition explains the names *concentration* and *base distribution*. In what follows we will build more intuition about the form that a draw from a DP takes.

As we mentioned above, if $G \sim \mathsf{DP}(\alpha, H)$, G is a distribution and we can draw samples $\theta_n \sim G$. We will show next that if $\theta_1, \cdots, \theta_N$ are draws from G they exhibit a clustering property equivalent to the CRP. Let us derive an expression for the posterior distribution $G|\theta_1, \cdots, \theta_N$. First of all we show that the posterior of G is a DP again. For any partition

[1] Technically, G must assign probability mass to all elements of a σ-algebra over Θ.

[2] See appendix A for more background on the Dirichlet distribution.

$A_1, \cdots, A_l$, let c_i be the number of θ's in A_i: $c_i = |\{k|\theta_k \in A_i\}|$. Because of the conjugacy of the Dirichlet and multinomial distributions, we have that

$$(G(A_1), \cdots, G(A_l)) \sim \mathsf{Dirichlet}(\alpha H(A_1) + c_1, \cdots, \alpha H(A_l) + c_l). \tag{2.2}$$

Since this is true for any partitioning of Θ, $G|\theta_1, \cdots, \theta_N$ satisfies the defining property of a DP and hence the posterior is a DP again. Now we want to derive an explicit update for the posterior of the concentration parameter and base distribution. Let $\theta_1^*, \cdots, \theta_l^*$ be the unique values among $\theta_1, \cdots, \theta_N$. Choose a partition $A_1, \cdots, A_l, A_{l+1}, \cdots, A_{l+k}$ such that $A_i = \{\theta_i^*\}$ and $A_{l+1} \cup \cdots \cup A_{l+k} = \Theta \setminus A_1 \setminus \cdots \setminus A_l$. Let c_i be the number of θ_n's that are equal to θ_i^*: $c_i = |\{k|\theta_k \in A_i\}|$. From the definition of a Dirichlet process and the conjugacy of the Dirichlet and multinomial distributions we have

$$\begin{aligned}(G(A_1), \cdots, G(A_{l+1})) &\sim \mathsf{Dirichlet}(\alpha H(A_1) + c_1, \cdots, \alpha H(A_l) + c_l, \alpha H(A_{l+1}), \cdots, \alpha H(A_{l+k})) \\ &\sim \mathsf{Dirichlet}(c_1, \cdots, c_l, \alpha H(A_{l+1}), \cdots, \alpha H(A_{l+k})).\end{aligned} \tag{2.3}$$

Since this holds for any partition of the space Θ, this is by definition again a Dirichlet Process with concentration parameter $\alpha + N$ and base measure $\frac{\alpha H + \sum_{n=1}^N \delta_{\theta_n}}{\alpha + N}$.

Equation (2.3) allows us to compute the predictive distribution $\theta_{N+1}|\theta_1, \cdots, \theta_N$ as follows. Choose any set $A \subseteq \Theta$ and let us compute $p(\theta_{N+1} \in A|\theta_1, \cdots, \theta_N)$. We integrate out the DP G and find

$$\begin{aligned}p(\theta_{N+1} \in A|\theta_1, \cdots, \theta_N) &= \int p(\theta_{N+1} \in A|G) p(G|\theta_1, \cdots, \theta_N) dG \\ &= \int G(A) p(G|\theta_1, \cdots, \theta_N) dG \\ &= \mathbb{E}[G(A)|\theta_1, \cdots, \theta_N] \\ &= \frac{1}{\alpha + N}\left(\alpha H(A) + \sum_{n=1}^N \delta_{\theta_n}(A)\right) \\ &= \frac{\alpha}{\alpha + N} H(A) + \sum_{i=1}^l \frac{c_i}{\alpha + N} \delta_{\theta_i^*}(A).\end{aligned}$$

How should we interpret this formula? If we let $A = \Theta \setminus \{\theta_1, \cdots, \theta_n\}$, then we find $\theta_{N+1}|\theta_1, \cdots, \theta_N \sim H$ with probability $\frac{\alpha}{\alpha+N}$. In other words with probability $\frac{\alpha}{\alpha+N}$, θ_{N+1} is a draw from the base measure H. If we let $A = \{\theta_i^*\}$ then $\theta_{N+1} = \theta_i^*$ with probability $\frac{c_i}{\alpha+N}$. This means that with probability $\frac{c_i}{\alpha+N}$ the new draw will be equal to an existing θ_i^*. This argument makes it clear that draws from G cluster together around the same θ_i^*; hence we will often refer to the θ_i^* as *atoms* of the distribution G. Note how the predictive probabilities are exactly the same as the one generated by the CRP.

There is a deeper mathematical connection between the DP and the CRP. Recall that the CRP defines an exchangeable distribution on the table assignments. The *de Finetti*

theorem (Aldous, 1983) says that for any infinitely exchangeable distribution $x_1, x_2, \cdots$ there exists a random measure F, called the de Finetti mixing distribution which renders the x_i conditionally independent, in equations

$$p(x_1, \cdots x_N) = \int \prod_{n=1}^{N} P(x_n) F(dP). \tag{2.4}$$

This is quite a remarkable theorem. The exchangeability condition is a very natural assumption to make: it essentially says that the order of the data points does not play a role in the probability of a dataset. E.g. for the Sloan Digital Sky survey problem we analyzed in section 1.1, it is unimportant which data point we call observation 1 and which data point we call observation 2: this information is irrelevant with respect to the problem we are solving. The de Finetti theorem now says that if the order of the data points is irrelevant, there *must* be an underlying statistical model, potentially very complicated, which renders the data points conditionally independent. Applying the de Finetti theorem to the CRP, there must be a random measure F so that the samples from the CRP are conditionally independent. It can be shown that the de Finetti mixing distribution for the CRP is the DP.

The Stick Breaking Construction

A final perspective on the CRP and DP comes in the form of the stick breaking construction by Sethuraman (1994). This will prove to be a valuable addition to our set of representations for the DP on which we can build slice sampling inference algorithms.

$$\begin{aligned}
&\forall k \in \{1, \cdots, \infty\}, \\
\beta_k|\alpha &\sim \mathsf{Beta}(1, \alpha) \\
\pi_k|\beta_{1:k} &= \beta_k \prod_{l=1}^{k-1}(1-\beta_l) \\
\theta_k|H &\sim H \\
G(\cdot)|\pi, \theta &= \sum_{k=1}^{\infty} \pi_k \delta_{\theta_i}(\cdot)
\end{aligned}$$

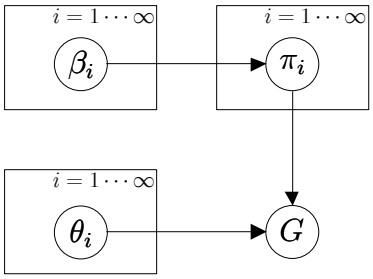


Figure 2.2: Graphical model for a Dirichlet process built using the stick-breaking construction.

The CRP representation suggests that a DP can be represented by a countable (one for each table) number of atoms. In other words, we expect to be able to express the

DP as a weighted sum of point masses. Sethuraman (1994) shows this intuition holds and derives the distribution for the mixture weights. The construction can be described as follows: we start with a stick of length 1 and draw $\beta_1 \sim \mathsf{Beta}(1, \alpha)$. We break a β_1 fraction off the stick and assign its length to a variable π_1. Then we draw a new variable $\beta_2 \sim \mathsf{Beta}(1, \alpha)$ and break a fraction from the remaining stick (of length $1-\pi_1$) and assign its length to π_2. We iteratively break new chunks off ad infinitum. Finally, we draw an atom $\theta_i \sim H$ for each stick π_i and construct the following density: $G(\cdot) = \sum_{k=1}^{\infty} \pi_k \delta_{\theta_i}(\cdot)$. Figure 2.2 shows the graphical model for the stick breaking construction.

Sethuraman (1994) showed that $G \sim \mathsf{DP}(\alpha, H)$. The argument is roughly as follows: if $G \sim \mathsf{DP}(\alpha, H)$ and $x \sim G$ then $G|x \sim \mathsf{DP}(\alpha + 1, H + \delta_x(\cdot))$. We also know that $G(\{x\}) \sim \mathsf{Beta}(1, \alpha)$ from the defining property of the DP. It can be shown (Hjort et al., 2010, section 2.2) that the DP satisfies a self-similarity property: restricted to the complement set $\{x\}^c$, G still follows a DP with parameters $\mathsf{DP}(\alpha, H)$. Hence, we can sample from the base distribution and a stick length from $\mathsf{Beta}(1, \alpha)$ to find the location and weight of the one atom of the DP and then recurse. We will often use the shorthand notation $\pi \sim \mathsf{Stick}(\alpha)$ to denote the distribution over π.

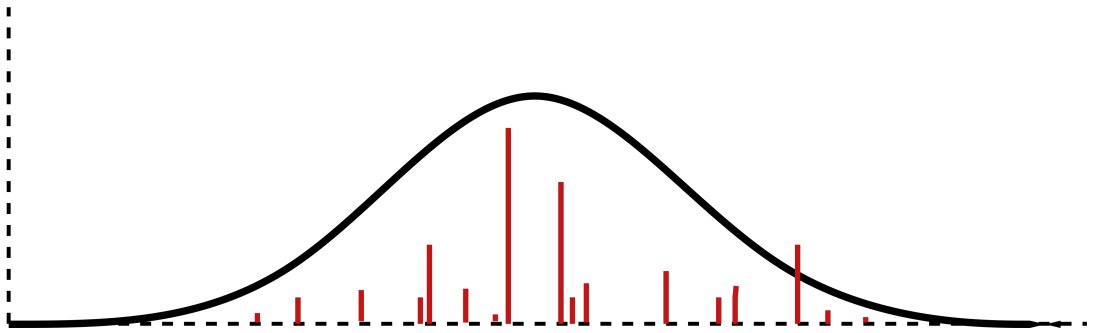


Figure 2.3: Visualisation of a Dirichlet process with Gaussian base measure.

Figure 2.3 illustrates how one can imagine a Dirichlet process; as an infinite collection of sticks at random points in the space spanned by the base measure H. The sticks represent probabilities and must sum up to one. The locations of the sticks are random draws from the base measure H.

Infinite Mixture Models

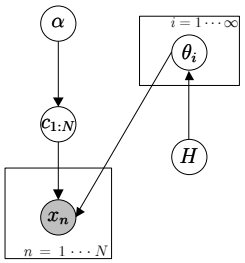
We motivated this section as a way to build infinite mixture models: we now conclude our discussion by describing an abstract infinite mixture model using the different constructions described above. The infinite limit in section 1.2 and Polya urn are equivalent to the CRP so we leave these out of our discussion. We will assume that the parameters of the mixture components come from an arbitrary base distribution H and the likelihood model is F.

To extend the CRP into a full blown mixture model we first draw a table assignment from the CRP: $p(c_{1:N}|\alpha)$. Then for each table i we draw parameters $\theta_i \sim H$. Finally, we add a likelihood $p(x_n|c_n, \theta)$ by independently drawing data from a likelihood model $x_n \sim F(\theta_{c_n})$.

We can extend the DP into a mixture model by first drawing $G \sim \mathsf{DP}(\alpha, H)$. Then for each data point we draw $\theta_n \sim G$ and $x_n \sim F(\theta_n)$. Because of the properties of the DP described above the parameters θ_n will cluster together.

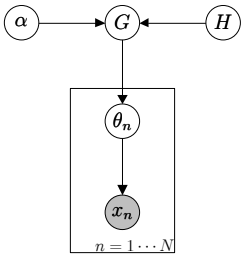
Finally, we can use the stick breaking construction to build a mixture model as follows. First we draw $\pi \sim \mathsf{Stick}(\alpha)$ and an infinite set of atoms $\theta_i \sim H$ for $i \in \{1 \cdots \infty\}$. Then, for each data point n we draw a cluster assignment $c_n \sim \pi$ and set $x_n \sim F(\theta_{c_n})$.

Figure 2.4 illustrates the graphical model for each of these three constructions.



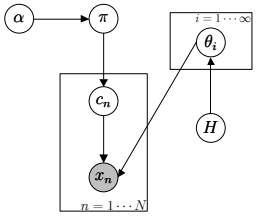
$\forall n \in \{1 \cdots N\}$,
$\forall i \in \{1 \cdots \infty\}$,

$$c_{1:N}|\alpha \sim \mathsf{CRP}(\alpha)$$
$$\theta_i|H \sim H$$
$$x_n|\theta_{c_n} \sim F(x_n|\theta_{c_n})$$



$\forall n \in \{1 \cdots N\}$,

$$G|\alpha, H \sim \mathsf{DP}(\alpha, H)$$
$$\theta_n|G \sim G$$
$$x_n|\theta_n \sim F(x_n|\theta_n)$$



$\forall n \in \{1 \cdots N\}$,
$\forall i \in \{1 \cdots \infty\}$,

$$\pi|\alpha \sim \mathsf{Stick}(\alpha)$$
$$\theta_i|H \sim H$$
$$c_n|\pi \sim \pi$$
$$x_n|\theta_{c_n} \sim F(x_n|\theta_{c_n})$$

Figure 2.4: Comparison of three different constructions of an infinite capacity mixture model. Left: using the CRP; middle: using the DP; right: using the stick-breaking construction.

2.1.2 Inference

There are a number of inference tasks related to infinite mixture models. The most common task is to find the posterior partitioning of the data points: which data points belong to which cluster? The most common inference algorithms are based on the CRP and stick breaking representation where inference amounts to computing the posterior

distribution $p(c_{1:N}|x_{1:N}, \alpha, H, F)$. As a (deterministic) side effect, we can easily compute the posterior distribution over the number of clusters from this representation. Another common task is to infer the posterior distribution of cluster parameters, the θ_i in the CRP, DP and stick breaking constructions. Finally, one might be interested in learning the posterior for the concentration measure of the DP: $p(\alpha|x_{1:N}, \alpha, H, F)$. This task requires us to introduce a prior distribution on α; since α needs to be positive, a Gamma distribution is a common choice.

All of the posterior distributions mentioned above are analytically intractable to compute but various approximate inference algorithms exist. All currently known techniques can be classified into two categories: the deterministic optimisation based methods and the randomised MCMC methods. Blei and Jordan (2006) pioneered a variational approximation to the DP mixture by approximating the DP posterior with a truncated stick breaking representation. Kurihara et al. (2007a) further explored the space of truncated variational approximations by deriving a collapsed variational approximation scheme and comparing it to the scheme in Blei and Jordan (2006) and to a standard finite mixture model based on a symmetric Dirichlet prior on the mixture components. Kurihara et al. (2007b) further extended the method in Kurihara et al. (2007a) using kd-trees for improved computation time. Zobay (2009) performs a careful analysis of variational optimisation for the DP mixture and makes several remarkable conclusions:

1. The fixed point iterations in variational solutions suppress the attachment of data points to new components.

2. The collapsed variational solutions quantitatively lead to very similar results as the uncollapsed solutions.

3. Posterior inference on the concentration parameters α generally leads to overconfident approximations; with larger truncation levels, the approximation on α becomes a delta spike with a constant mean.

4. Predictive distributions under the variational approximation are often good, whereas there generally are strong discrepancies regarding clustering and number of posterior components compared to the true posterior.

Minka and Ghahramani (2003) describe a variational solution based on the expectation propagation algorithm. This result is exciting as it does not rely on a truncated approximation. Its main disadvantage is that the approximating posterior does not exhibit the clustering properties of a DP: each data point sits in its own cluster with no overlap in cluster parameters between the data points.

In the class of deterministic algorithms there are other inference algorithms based on combinatorial search which are often superior to the variational algorithms above while

much simpler to implement. The first is the tree based search algorithm of Xu et al. (2009). This algorithm manages to efficiently integrate the CRP over an exponential number of partitions. Another noteworthy algorithm is the A* based search algorithm in Daume (2007).

Various MCMC algorithms for DP mixtures have been introduced (Neal, 1991, MacEachern, 1994, Escobar, 1994, Escobar and West, 1995, MacEachern and Müller, 1998, Rasmussen, 2000, Ishwaran and James, 2001, Porteous et al., 2006, Papaspiliopoulos et al., 2008). A great overview paper comparing various algorithms is Neal (2000). We briefly describe a collapsed Gibbs sampler as it is the foundation for one of our sampling algorithm in chapter 3.

This collapsed Gibbs sampler uses the CRP representation and only resamples the cluster assignments $c_{1:N}$. The algorithm, illustrated in algorithm 1 is extremely simple. In

Algorithm 1 The collapsed sampler for the DP mixture.

Initialise $c_{1:N}$ randomly.
loop
  for $n = 1$ to N **do**
    Sample cluster assignments $c_n|c_{-n}, x_{1:N}, F, H, \alpha$
  end for
end loop

each iteration, each c_n is re-sampled from its posterior distribution $p(c_n|c_{-n}, x_{1:N}, F, H, \alpha) \propto p(c_n|c_{-n}, \alpha) \cdot p(x_n|x_{-n}, c_n, F, H)$. The prior contribution $p(c_n|c_{-n}, \alpha)$ is trivial to compute: because of exchangeability, we can assume data point n is the last data point, hence $p(c_n|c_{-n}, \alpha) \propto m_{c_n}$ where m_{c_n} is the number of data points in cluster c_n or α when c_n represents a new cluster. The likelihood contribution $p(x_n|x_{-n}, c_n, F, H)$ can be computed analytically when the base distribution H is conjugate to the likelihood F: $p(x_n|x_{-n}, c_n, F, H) = \int d\theta p(x_n|\theta)p(\theta|x_{-n}, c_n, F, H)$.

In our next chapter we will construct a sampler that is inspired by yet a different type of sampler for the DP mixture: the slice sampler (Neal, 2003, Walker, 2007, Kalli et al., 2008). Because of its relevance to our thesis we briefly review the core idea here. The slice sampler, based on the stick breaking construction, re-samples the mixture weights π and the cluster assignments $c_{1:N}$. Algorithm 2 describes the sampler.

Using the marginalisation property of the DP, we know that we can sample the mixture parameters as $\pi \sim \mathsf{Dirichlet}(m_1, m_2, \cdots, m_K, \alpha)$ where the last element represents the mass of all new clusters. The crucial step is re-sampling the cluster assignments; for each $n \in \{1 \cdots, N\}$, we need to sample from $p(c_n|\pi, u_n, F, H)$. Using Bayes rule, we know

$$p(c_n|x_{1:N}, u_n, \pi, F, H) \propto p(u_n|c_n, \pi) \cdot p(x_n|x_{-n}, F, H) \tag{2.5}$$

Algorithm 2 The slice sampler for the DP mixture.

Initialise $c_{1:N}$ randomly.

loop

Sample the mixture parameters $\pi|c_{1:N}, \alpha$

Sample auxiliary variables $u_n \sim \mathsf{Uniform}(0, \pi_{c_n})$

Sample cluster assignments $c_{1:N}|\pi, u_{1:N}, x_{1:N}, F, H$

end loop

From the collapsed Gibbs sampler we already know that

$$p(x_n|x_{-n}, F, H) = \int d\theta p(x_n|\theta)p(\theta|x_{-n}, c_n, F, H) \tag{2.6}$$

which can be computed analytically when H is conjugate to the likelihood F. We can write the uniform distribution on the auxiliary variable as $p(u_n|c_n, \pi) = \frac{1}{\pi_{c_n}}\mathbb{I}[0 \leq u_n \leq \pi_{c_n}]$. The key insight into equation (2.5) is that although c_n can take on any of an infinite number of cluster assignments (the K currently occupied clusters or any of the infinite number of remaining clusters), any assignment needs to satisfy the $\mathbb{I}[0 \leq u_n \leq \pi_{c_n}]$ constraint: we cannot assign c_n to a cluster such that $\pi_{c_n} < u_n$. There can only be a finite number of clusters that satisfy this constraint since $\sum_i \pi_i = 1$. Hence, in each iteration, for each data point, we only need to consider a finite number of cluster assignments. The slice sampler thus adaptively truncates the DP mixture using auxiliary variables. In that sense it combines the computational advantages of a truncated representation while maintaining the property that all samples represent the true posterior distribution.

We conclude this section with a brief description of methods to estimate the parameters of the DP. The most common method for learning the concentration parameter and base measure is a full Bayesian treatment. More specifically, for the concentration parameter we can use the property that the posterior $p(\alpha|c_{1:N})$ is only dependent on the number of clusters K in the sample represented by $c_{1:N}$

$$p(\alpha|c_{1:N}) = p(\alpha|K, N) \propto p(K|\alpha, N)p(\alpha), \tag{2.7}$$

where from Antoniak (1974) we know that $p(K|\alpha, N) \propto \alpha^K \frac{\Gamma(\alpha)}{\alpha+N}$. In Escobar and West (1995) an auxilary variable sampler is introduced to sample from the posterior $p(\alpha|c_{1:N})$: it uses the property that when the prior $p(\alpha)$ is a Gamma distribution, the posterior is the marginal of a bi-variate distribution that is a mixture of two Gamma distributions and a Beta distributed auxiliary variable. Rasmussen (2000) uses the property that the posterior on α is log concave so the adaptive rejection sampler can be used for re-sampling α. McAuliffe et al. (2006) uses an empirical Bayes estimate of α by using the property that the marginal maximum likelihood estimate of α must satisfy

$$\mathbb{E}[K] = \sum_{n=1}^{N} \frac{\alpha}{\alpha + n - 1}. \tag{2.8}$$

The left hand side can be estimated using the samplers described above; then numerical optimisation can be used to solve for α.

A second parameter which can be learned from data is the base measure of the DP. (Hjort et al., 2010, chapter 7) discusses this problem in full detail and offers three solutions. The first is a full Bayesian treatment for estimating the base measure; we refer to section 2.2 for an application of this idea. Another possibility is to use an empirical Bayes estimate of the base measure, McAuliffe et al. (2006) describes this approach in detail and gives an example using kernel density estimates. A final alternative is to explicitly specify the base distribution using prior knowledge.

2.1.3 Properties

The CRP, DP and its stick breaking construction have a number of interesting properties which we need to understand for applied statistics using this Bayesian nonparametric distribution.

Number of Clusters We can easily derive the distribution on the number of partitions K in the CRP: let I_n be an indicator variable for the event that customer n sits at a new table. Then $K = \sum_{n=1}^{N} I_n$ and each $I_n \sim \mathsf{Bernoulli}(\alpha/n)$ independently. It is then straightforward to see that $\mathbb{E}[K] = \sum_{n=1}^{N} \mathbb{E}[I_n] = \sum_{n=1}^{N} \frac{\alpha}{n} = \alpha H_n$ where H_n is the nth Harmonic number. Pitman (2006) strengthens this result to show that K converges in distribution to a $\mathsf{Normal}(\log(n), \log(n))$ random variable. Figure 2.5 illustrates the scaling of the number of partitions K as the number of customers N in a CRP and the concentration parameter α varies. These plots were obtained by sampling from a CRP; for each parameter setting, 100 CRP samples were taken and the mean and standard deviation K in the 100 samples were computed. The scaling with varying N used an $\alpha = 1$ concentration parameter whereas the scaling with varying α used $N = 100000$ customers. The left plot in figure 2.5 shows the logarithmic growth of the number of clusters, a least squares fit results in a trend line $y = 0.9599 \ln(x) + 0.8046$. The right plot in figure 2.5 shows the linear growth in function of α. Note how in both plots the variance increases with increasing parameter.

Distribution of Cluster Sizes Another important property of the DP is how the cluster sizes are distributed. We perform the following experiment: we sample 2000 DP's each with $N = 10000$ customers and $\alpha = 1$. For each sample, we order the clusters according to size and compute the mean cluster size at each rank. Figure 2.6 shows the log cluster size versus log rank.

The key property to point out from this plot is how fast the log mean cluster size decreases with increasing rank. Many natural phenomena show power-law behaviour:

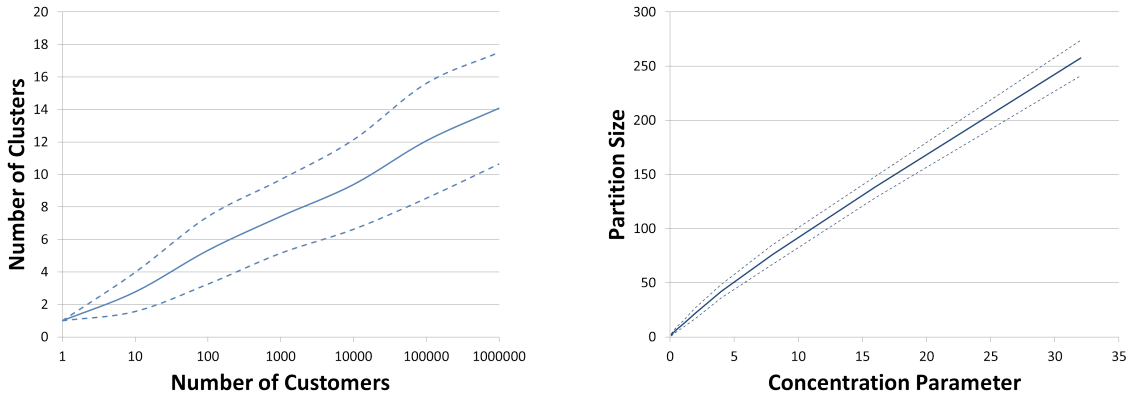


Figure 2.5: Empirical scaling behaviour of the number of partitions. In the left plot we fix $\alpha = 1$ and vary the number of customers N; in the right plot we fix $N = 100000$ and we vary the concentration parameter α.

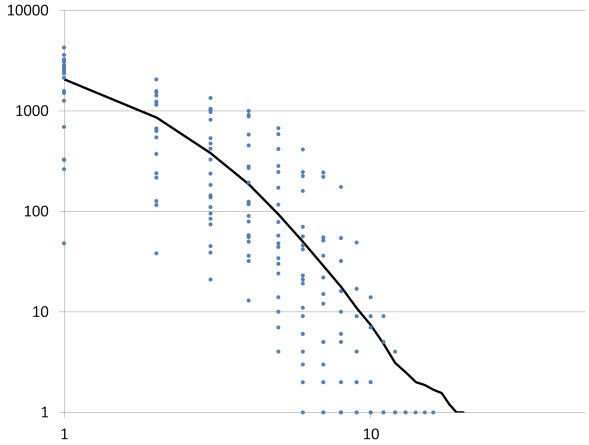


Figure 2.6: Log-log plot of mean cluster size versus cluster rank. The black line shows the mean cluster rank, the blue dots show the actual datapoints for the first 20 samples.

the number of clusters of size k is proportional to c^k. Power-law behaviour would result in many small clusters; given the fast decrease of the log mean cluster size we expect that the DP cluster sizes do not follow a power-law. Indeed, it is the generalisation of the DP, called the Pitman-Yor process (Pitman and Yor, 1997) that exhibits power-law behaviour. We will return to this distribution in chapter 3.

Dependency on α Antoniak (1974) proves that the distribution of the number of clusters K conditional on α can be written as

$$p(K|\alpha) \propto \alpha^K \frac{\Gamma(\alpha)}{\Gamma(\alpha + N)} \tag{2.9}$$

where the normalisation constant involves Stirling numbers of the second kind. This is an important distribution when we want to learn α in a hierarchical Bayesian fashion.

We perform the following experiment: we put a Gamma prior on α and try to learn the posterior $p(\alpha|K)$. First of all, note that this is a much easier task than learning α from a full mixture model: we assume full knowledge of K whereas in a full mixture model this variable is unknown as well. The posterior on α can be written as

$$p(\alpha|K) \propto p(K|\alpha)p(\alpha|a,b) \propto \alpha^{a+K-1}\frac{\Gamma(\alpha)}{\Gamma(\alpha+N)}e^{-b\alpha}. \tag{2.10}$$

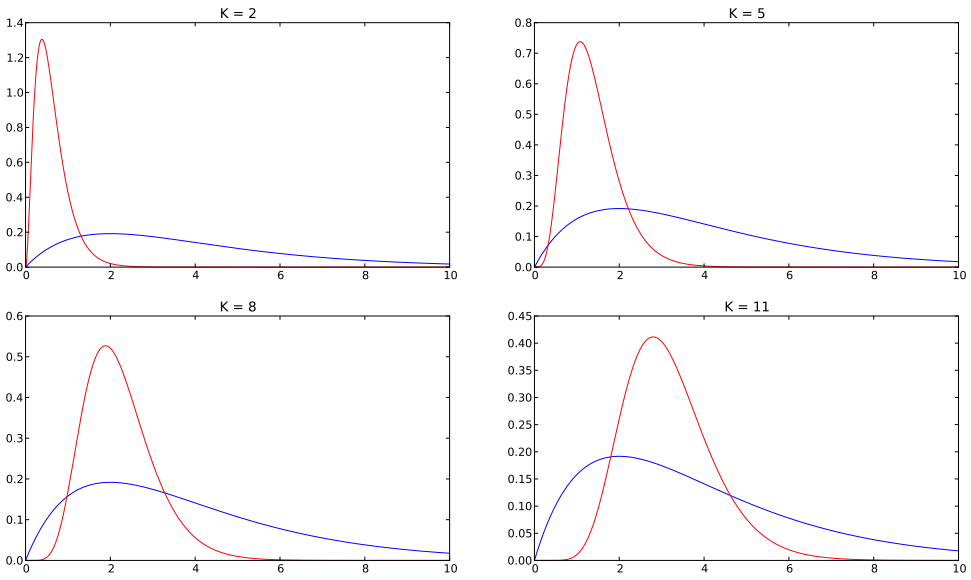


Figure 2.7: Gamma$(2.0, 0.5)$ prior (blue) and posterior (red) distribution for a DP with $N = 100$ customers and varying number of clusters K.

Figure 2.7 illustrates a prior on α and the corresponding posterior for a DP with $N = 100$ customers. The interesting observation to make here is that when the number of clusters K is large, the posterior is rather similar to the prior. This is an often overlooked fact in the literature: the marginal likelihood $p(K|\alpha)$ gives very little information about the true value of α when K is large. In other words, we cannot learn a very specific value of α for problems with a large number of clusters. Note that when K is not known exactly (as in a mixture model), the variance on α can only increase further. We will have to consider this effect carefully in applied problems involving the DP.

2.1.4 Discussion

Applications of the DP are numerous; ranging from statistical models in physics and bio statistics to machine learning and natural language processing; we refer to the overview paper (Teh, 2010) for a brief summary and further references. An area which we haven't

touched upon are asymptotic properties of the DP such as posterior consistency and convergence rates. We refer to (Hjort et al., 2010, chapter 2) for an overview of the current state of the art in this area.

2.2 Chinese Restaurant Franchises and Hierarchical Dirichlet Processes

A common technique in Bayesian modelling it that whenever we are unsure about a parameter in our model, we assume it is a random variable, introduce an appropriate prior and learn about the parameter through data. As we discussed previously, in the case of the DP, if we do not know the base measure but we have several experiments in which it interacts, we can build a hierarchical Bayesian model involving the base measure of a DP as a random variable. In this next section we consider this model in more detail.

Consider the problem where we are given measurements of multiple patients in different hospitals and we need to cluster patients. We could cluster the patients using a DP mixture for each hospital separately but this would very likely result in clusters across hospitals being different. We might believe that clusters of patients in one hospital also exist as clusters in a different hospital. In other words, we would like to share clusters between hospitals. More formally, we want to model the following data set: we have measurements x_{ji} where $j \in \{1 \cdots J\}$ denotes the group of the data points and $i \in \{1 \cdots n_j\}$ denotes the i'th data point in group j. We now want to cluster the data points in each group such that cluster parameters are shared between groups.

The hierarchical Dirichlet process (HDP) introduced in Teh et al. (2006a) addresses this problem. In section 2.1 we introduced the Dirichlet process as a way to share cluster parameters (the atoms of the DP) between data points. In our hospital setting we still want to share cluster parameters between data points but different hospitals ask for different mixture probabilities (or atom weights): the HDP addresses exactly this issue.

2.2.1 Definition and Constructions

Just as with the DP, we will describe a measure theoretic construction, a combinatorial process and a stick breaking construction for essentially the same distribution.

The Hierarchical Dirichlet Process

Section 2.1.1 showed how a DP can be represented as an infinite mixture of atoms drawn IID from the base measure H: $G(\cdot) = \sum_{k=1}^{\infty} \pi_k \delta_{\theta_i}(\cdot)$. We now investigate the role of the base measure H on the infinite mixture by drawing two DP's $G_1(\cdot) = \sum_{k=1}^{\infty} \pi_k \delta_{\theta_{1i}}(\cdot)$ and $G_2(\cdot) = \sum_{k=1}^{\infty} \pi_k \delta_{\theta_{2i}}(\cdot)$ from the same base measure. If H is a continuous measure, say

a multivariate Gaussian distribution, then all $\theta_{\cdot i} \sim H$ will be different almost surely. In particular, G_1 and G_2 will almost surely not share any atoms. On the other hand, if H is a discrete measure (either finite or infinite) it is easy to show that G_1 and G_2 will share atoms. To see this, consider any atom from G_1, say θ_{1i}. Since H is discrete, $H(\theta_{1i})$ is a nonzero probability. Using the stick-breaking construction of the DP we know that when we construct G_2 we will draw a countable infinite number of atoms from H and hence with probability one draw θ_{1i} at least once. In other words, G_1 and G_2 will almost surely share atoms.

$$
\begin{aligned}
&\forall j \in \{1 \cdots J\}, \\
&\forall i \in \{1 \cdots n_j\}, \\
&\qquad G_0|\gamma, H \;\sim\; \mathsf{DP}(\gamma, H) \\
&\qquad G_j|\alpha, G_0 \;\sim\; \mathsf{DP}(\alpha, G_0) \\
&\qquad \theta_{ji}|G_j \;\sim\; G_j \\
&\qquad \boldsymbol{x}_{ji}|\theta_{ji} \;\sim\; F(\theta_{ji})
\end{aligned}
$$

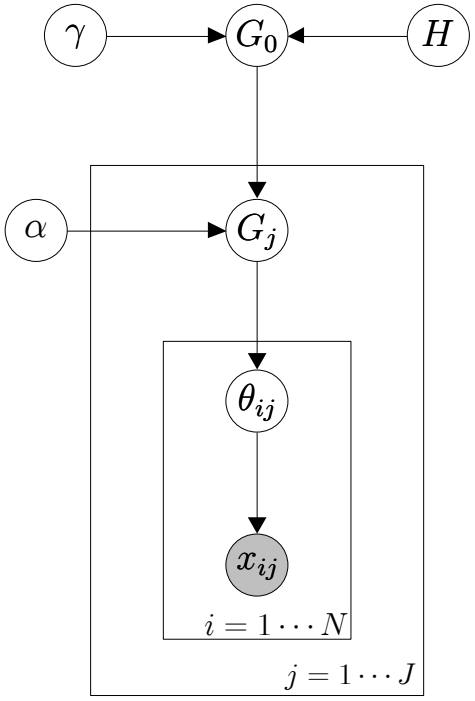


Figure 2.8: The graphical model for a Hierarchical Dirichlet Process.

This insight suggests the following approach for our group clustering problem: we define an appropriate discrete base measure G_0 for the parameters of the clusters and then draw a $G_j \sim \mathsf{DP}(\alpha, G_0)$ for each group $j \in \{1 \cdots J\}$. If G_0 is a finite discrete measure the G_js will share cluster parameters but there will only be a finite number of them. This would be a step back to parametric modelling. Hence, we would like G_0 to be an infinite discrete measure: we can achieve this by letting G_0 be a draw from a DP itself! Figure 2.8 shows the generative model for this construction which has been coined the hierarchical Dirichlet process. Note that the HDP can be extended to more than two levels by setting up a tree structure where each node is a DP with its parent as the base measure and that the distribution defined by the HDP is still exchangeable in customers

but is now also exchangeable in the groups.

The Stick Breaking Construction

A natural question one could ask is whether a stick-breaking representation for the HDP exists. Fortunately it does and in this section we will describe how the construction works. Let us consider the graphical model for the HDP in figure 2.8. The stick-breaking construction of the DP in section 2.1.1 learns us that we can write

$$G_0(\cdot) = \sum_{k=1}^{\infty} \pi_{0k} \delta_{\theta_k}(\cdot) \tag{2.11}$$

with $\pi_0 \sim \mathsf{Stick}(\gamma)$. From the previous section we know that each G_j will have exactly the same set of atoms as G_0. In other words, we can write

$$G_j(\cdot) = \sum_{k=1}^{\infty} \pi_{jk} \delta_{\theta_k}(\cdot) \tag{2.12}$$

where the question is: what form does the vector π_j take? In other words, we want to know how much probability mass G_j assigns to each atom θ_i. This follows directly from the definition of the DP: let us consider the first k atoms and define a partitioning of Θ (the space of our atoms) into k singleton sets $A_1, \cdots, A_k$ with $A_i = \{\theta_i\}$ and one set to cover the rest of Θ namely $A_{k+1} = \Theta \setminus A_1 \setminus \cdots \setminus A_k$. The definition of a DP now dictates that the marginals induced by the partitioning follow a Dirichlet distribution

$$(G_j(A_1), \cdots, G_j(A_k), G_j(A_{k+1})) \sim \mathsf{Dirichlet}(\alpha G_0(A_1), \cdots, \alpha G_0(A_k), \alpha G_0(A_{k+1})). \tag{2.13}$$

By construction, $G_j(A_i)$ is the probability of an atom appearing in set A_i but since A_i is the singleton set $\{\theta_i\}$ we know that $G_j(A_i)$ directly specifies the probability mass on atom i; in other words $G_j(A_i) = \pi_{ji}$. Similarly $G_j(A_{k+1})$ is the probability of an atom appearing in set A_{k+1}. Since A_{k+1} covers all atoms other than the first k we know that $G_j(A_{k+1}) = \sum_{l=k+1}^{\infty} \pi_{jl}$. Note that these two properties also hold for $j = 0$. Then it follows that

$$\left(\pi_{j1}, \cdots, \pi_{jk}, \sum_{l=k+1}^{\infty} \pi_{jl}\right) \sim \mathsf{Dirichlet}\left(\alpha\pi_{01}, \cdots, \alpha\pi_{0k}, \alpha\left(\sum_{l=k+1}^{\infty} \pi_{0l}\right)\right). \tag{2.14}$$

Intuitively, equation (2.14) says that any finite representation of G_j follows a Dirichlet distribution around the stick representation of G_0. Equation (2.14) also makes clear the role of the random variable α: if α is small the Dirichlet distribution will introduce a lot of variability between different G_j's, while if α is large, each G_j will be very similar to G_0 and hence also to each other. Finally, after some algebraic manipulation it can be

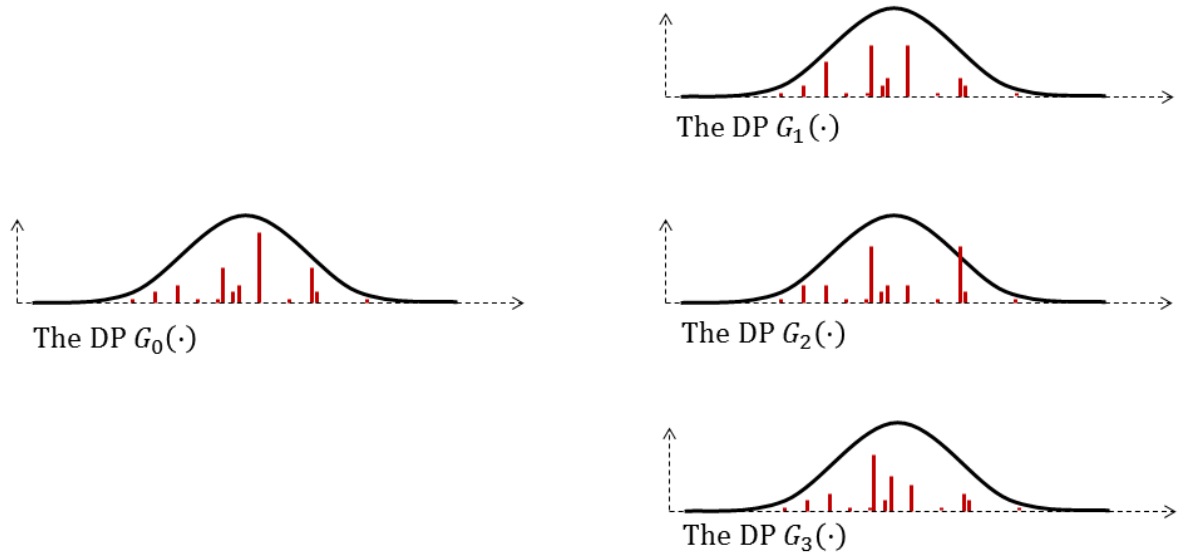


Figure 2.9: Visualisation of the hierarchical Dirichlet process. Left: the stick-breaking representation of the base measure; right: the stick-breaking representations of the child Dirichlet processes.

shown (Teh et al., 2006a) that equation (2.14) is similar to the following stick-breaking construction: let $\beta_{jk} \sim \text{Beta}\left(\alpha\pi_{0k}, \alpha\left(1 - \sum_{l=1}^{k}\pi_{0l}\right)\right)$, then $\pi_{jk} = \beta_{jk}\prod_{l=1}^{k-1}(1-\beta_{jl})$.

The stick-breaking representation allows us to visualise the hierarchical Dirichlet process: since G_0 and all G_js are DP's, we can visualise the sharing of atoms by considering the stick breaking representations. Figure 2.9 shows how G_0 consists of draws from a Gaussian base measure H and each G_j in turn shares the same atoms as G_0 but has slightly different weights. From the stick breaking properties of the DP we know that the hyper parameter γ in the HDP controls how fast the stick weights decay for G_0; in other words, it directly influences the amount of clusters we are likely to see.

The Chinese Restaurant Franchise

Analogous to the CRP representation of a DP we introduce the Chinese restaurant franchise (CRF) representation for the HDP. In this metaphor there are multiple restaurants that share a menu. Each restaurant corresponds to a group of data points while each dish on the menu corresponds to an atom in Θ. Each restaurant has a potentially infinite number of tables and at each table one particular dish is served. Different tables in a restaurant can serve the same dish.

Customer x_{ji} will sit at table t_{ji} in restaurant j. Each dish on the shared menu will be denoted with θ_i while we introduce the index k_{jt} to denote the dish that is served at table t in restaurant j; in other words, at table t in restaurant j, customers are eating

dish $\theta_{k_{jt}}$. We denote with θ_{ji} the dish that is eaten by customer i in restaurant j. Let n_{jt} be the number of customers sitting at table t in restaurant j and let m_k be the number of tables serving dish k in all restaurants.

The Chinese restaurant franchise works as follows: in each restaurant, the customers enter sequentially. A customer decides to sit down at occupied table t with probability proportional to n_{jt} and decides to sit at a new table with probability proportional to α. If he sits at an occupied table he eats dish $\theta_{k_{jt}}$, whereas if he decides to sit at a new table he orders a dish k that has been ordered before with probability proportional to m_k and he orders a new dish with probability proportional to γ.

We can now derive the predictive distribution of the indicator variables t_{ji} and k_{jt} by integrating over $G_0(\cdot)$ and $G_j(\cdot)$

$$\begin{aligned} p(t_{ji}|t_{-ji},\alpha) &\propto \sum_t n_{jt}\delta(t_{ji},t) + \alpha\delta(t_{ji},\bar{t}) \\ p(k_{jt}|k_{-jt},\gamma) &\propto \sum_k m_k\delta(k_{jt},k) + \gamma\delta(k_{jt},\bar{k}) \end{aligned}$$

where t_{-ji} denotes all customer-table assignments except for customer ji, k_{-jt} denotes all table-dish assignments except for table jt, $\bar{t}$ denotes a new table and $\bar{k}$ denotes a new dish. A convenient way to look at the CRF is that each table represents a draw from $G_0(\cdot)$ and each customer represents a draw from some $G_j(\cdot)$.

2.2.2 Inference

Teh et al. (2006a) describe three different Gibbs samplers for the HDP; we briefly list the main ideas here and refer to the paper for full details. The first sampler, called *Posterior Sampling in the CRF*, re-samples t_{ji} and k_{jt} while integrating out the base measure $G_0(\cdot)$. This sampler couples the variables t_{ji} and k_{jt} for different restaurants j; a second sampler, named the *Posterior Sampling with an Augmented Representation*, explicitly samples t_{ji}, k_{jt} and a stick breaking representation of $G_0(\cdot)$ through the stick variables β. This decoupling can sometimes improve mixing times. A third sampler, the *Posterior Sampling by Direct Assignment*, only tracks the assignments of variables to mixture components through the variables $z_{ji} = k_{t_{ji}}$, rather than the indirect association using t_{ji} and k_{jt}. As far as we are aware, a thorough evaluation of all three samplers has not been performed but the direct assignment sampler is often preferred because of simpler bookkeeping. Johnson et al. (2009) improve the above samplers for the HDP by removing unnecessary representations.

There has been very little work on deterministic inference methods for the HDP. The collapsed variational inference scheme in Teh et al. (2008) is a variational inference scheme for a two level HDP. Although the method is computationally efficient, empirical evaluations suggest it is less accurate than the Gibbs samplers.

2.2.3 Discussion

We showed how the HDP is the natural extension of the DP for hierarchical Bayesian modelling. One interesting theoretical result is how the HDP can be recovered as the infinite limit of a finite hierarchical Bayesian mixture model (Teh et al., 2006a); analogous to the limit construction of the DP.

It can be shown that the number of clusters in the HDP grows as $\mathcal{O}(\log(J)+\log\log N)$ where J is the number of child DP's and N is the number of data points per child DP. This is extremely slow growth with respect to N and is often an inappropriate prior. In (Hjort et al., 2010, chapter 5), an overview of possible relaxations of the HDP assumption are described.

2.3 Indian Buffets and Beta Processes

The mixture models we have seen so far assume the existence of an (unknown) number of classes and postulate that each data point belongs to one and only one class. When we design a spam filter an email is either spam or ham; when we build the tree of life by clustering DNA, a sequence is either from Eubacteria, Archaea or Eukarya, ... This assumption is a useful one in the examples given but it is also rather limiting in other applications. Consider the news article abstract below.

Choosing Beijing Was a Drastic Mistake
The question as to how free reporters will be to cover the Olympic Games still hasn't been solved to everyone's satisfaction. Commentators in Germany say the spirit of the Olympics is at stake. Just days before the Aug. 8 opening ceremonies usher in the 2008 Beijing Olympics, reporters from around the world who are in China to cover the games are pulling their hair out. Two things, in particular, are driving them nuts: not knowing what they will be able to cover and not knowing how much the Chinese government will censor their online coverage ...

Table 2.1: News article from Spiegel Online, August 4 2008: http://www.spiegel.de/international/world/0,1518,569903,00.html

It would be wrong to classify this article as a pure sports article as it probably has a political message too. The mixture model assumption doesn't seem appropriate here. We would rather say that an article can have a number of themes some of which are present, some of which aren't. This idea has been introduced in the machine learning

$$
\begin{aligned}
&\forall m \in \{1 \cdots M\}, \\
&\quad \forall n \in \{1 \cdots N\}, \\
&\qquad A_{m\cdot} \sim \mathsf{Normal}(0, \Sigma) \\
&\qquad z_{nm} \sim \mathsf{Bernoulli}(\pi_m) \\
&\qquad x_{n\cdot} \sim \mathsf{Normal}(z_n A, \sigma^2 I)
\end{aligned}
$$

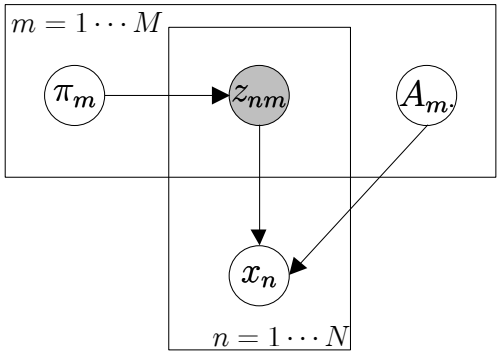


Figure 2.10: The graphical model for a finite factor model.

literature as latent feature models or factor models Barlow (1989), Hinton and Zemel (1994), Ghahramani (1995).

A simple example of a factor model is the following generative procedure. There is a set of M unknown features $A_{m\cdot}$ each drawn from a multivariate normal distribution. Whether data point n possesses feature m is indicated by the variable z_{nm}. The z_{nm} form a binary N by M feature matrix Z. Let π_m be the probability that a data point possesses feature m, then $z_{nm} \sim \mathsf{Bernoulli}(\pi_m)$ independently for all n. The observation for data point n is generated by summing up all features with $z_{nm} = 1$ and adding Gaussian noise: $x_n \sim \mathsf{Normal}(z_n A, \sigma^2)$. Figure 2.10 illustrates the graphical model for the factor model.

One could argue that the factor model is equivalent to having a clustering model with 2^M clusters (one cluster for each configuration of the latent features). However note that in the factor model, the cluster parameters $A_{m\cdot}$ would be very strongly dependent as they should all be sums of a set of M "base vectors". Hence it is much more reasonable and tractable to learn factorised representations than an exponentially larger amount of clusters. Similarly to the finite mixture model, the model in figure 2.10 explicitly represents the number of features M. We now turn to Bayesian nonparametric methods to explicitly model the uncertainty in the number of features.

2.3.1 Definition and Constructions

Central to the factorial model defined in figure 2.10 is the N by M matrix Z which specifies for all data points which of the M features are present. To turn this into a nonparametric distribution requires us to extend Z to have a potentially infinite number of columns. Our description will follow the same recipe as in the mixture model setting.

First we define a nonparametric distribution using a stochastic process following Griffiths and Ghahramani (2006). Then we define the same distribution by taking the infinite limit of a finite factorial model. Next we describe the de Finetti mixing distribution for the nonparametric object following Thibaux and Jordan (2007). Finally, we introduce an explicit stick-breaking representation based on the work in Teh et al. (2007).

The Indian Buffet Process

The Indian Buffet Process is a procedure for generating a binary matrix Z with a specific number of rows and a potentially unbounded number of columns. It is very similar to the Chinese restaurant process which we've discussed previously.

Imagine an Indian restaurant with an infinitely long buffet of dishes. The first customer enters the restaurant and takes a serving from each dish stopping after a $\mathsf{Poisson}(\alpha)$ number as his plate becomes overburdened. The nth customer moves along the buffet serving himself dish m with probability c_m/m where c_m is the number of previous customers who chose dish m. After reaching the end of the previously chosen dishes, customer n tries $\mathsf{Poisson}(\alpha/n)$ new dishes. Let z_{nm} be the binary indicator variable which denotes whether customer n chose dish m. We denote with Z the matrix of all indicator variables z_{nm}. After N customers enter the restaurant it is clear that the matrix Z will have N rows. Moreover, as each customer samples how many new dishes they will taste from a $\mathsf{Poisson}$ distribution and since this distribution can generate arbitrarily large numbers, the matrix Z could potentially have an arbitrary number of columns. We call this process the *Indian Buffet Process* or IBP. Figure 2.11 illustrates a draw from the Indian Buffet Process.

Let us denote with $M^{(n)}$ the number of new dishes sampled by customer n and with M_+ the total number of dishes tried. The probability of generating a particular feature matrix Z according to the procedure above is

$$p(Z|\alpha) = \frac{\alpha^{M_+}}{\prod_{n=1}^{N} M^{(n)}!} \exp\{-\alpha H_N\} \prod_{m=1}^{M_+} \frac{(N-c_m)!(c_m-1)!}{N!}, \tag{2.15}$$

where H_n is the nth harmonic number: $H_n = \sum_{j=1}^{N} \frac{1}{j}$. Equation (2.15) shows that the distribution is exchangeable as the probability distribution is only dependent on the counts c_m and not on the particular ordering of the N data points.

An Infinite Latent Feature Model

In this section we will show how we can recover the distribution induced by the IBP by taking the infinite limit of a finite model. We will do this in three steps following Griffiths and Ghahramani (2006): first we extend the model in figure 2.10 and do a full Bayesian analysis. Then we introduce equivalence classes of matrices and take an infinite limit.

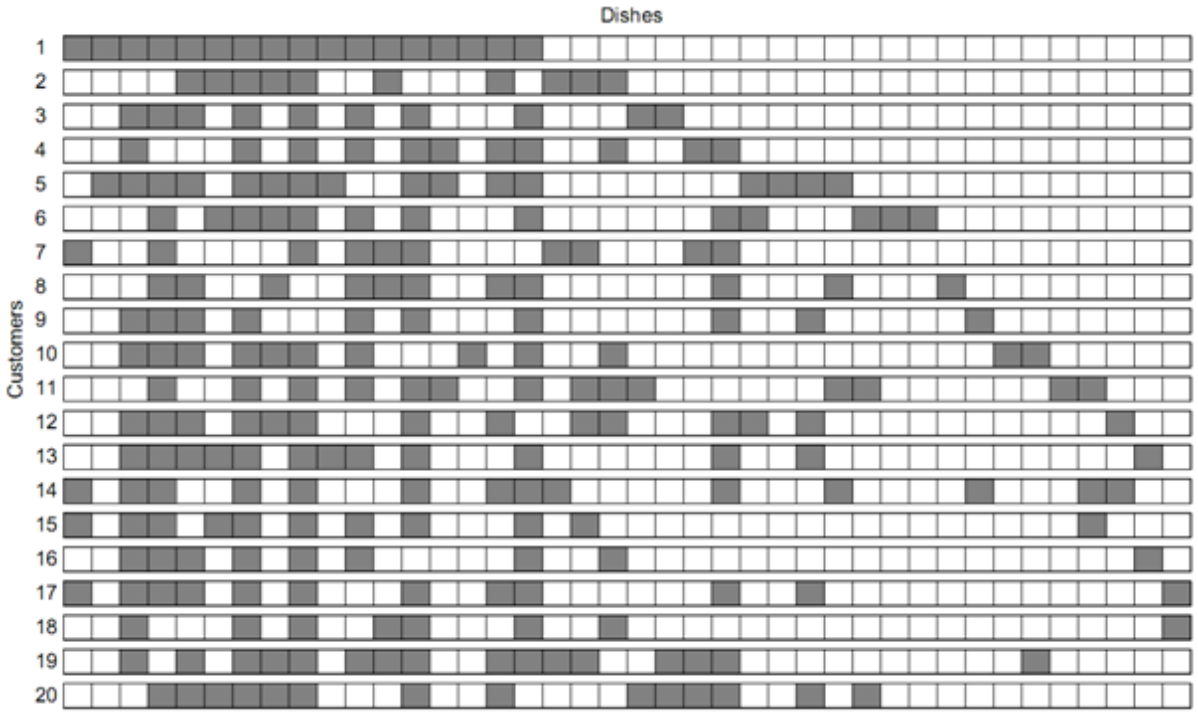


Figure 2.11: Illustration of a matrix generated by the IBP with $\alpha = 10$. Customer 1 chose 17 dishes, customer two chose 7 dishes sampled by customer 1 and 3 new dishes, etc ... After 20 customers entered the restaurant, 20 dishes were sampled. (With permission from Griffiths and Ghahramani (2006))

A Finite Feature Model We augment the model in figure 2.10 by introducing priors on the feature probabilities π_m and integrating them out. Recall that we draw $z_{nm} \sim \mathsf{Bernoulli}(\pi_m)$ and if we want to integrate out π_m it is natural to choose the conjugate prior for the $\mathsf{Bernoulli}$ distribution on π_m. This is the $\mathsf{Beta}(a, b)$ distribution with density

$$p(\pi_m|a, b) = \frac{\Gamma(a)\Gamma(b)}{\Gamma(a+b)}\pi_m^{a-1}(1-\pi_m)^{b-1}. \tag{2.16}$$

For reasons that will become clear later we will choose $a = \alpha/M, b = 1$. We now have the following graphical model

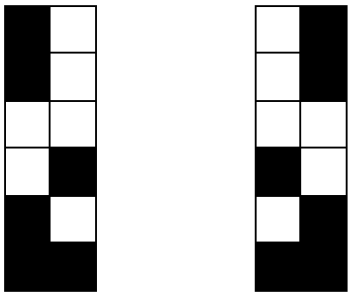
$$\begin{aligned} \pi_m &\sim \mathsf{Beta}\left(\frac{\alpha}{M}, 1\right) \\ z_{nm} &\sim \mathsf{Bernoulli}(\pi_m) \end{aligned}$$

for which we want to integrate out π_m. Using the conjugacy of the Bernoulli and Beta distributions we can compute the marginal probability on Z

$$\begin{aligned} p(Z|\alpha) &= \prod_{m=1}^{M} \int \left(\prod_{n=1}^{N} p(z_{nm}|\pi_m) \right) p(\pi_m|\alpha) d\pi_m \\ &= \prod_{m=1}^{M} \int \pi_m^{c_m}(1-\pi_m)^{N-c_m} p(\pi_m|\alpha) d\pi_m \\ &= \prod_{m=1}^{M} \frac{\frac{\alpha}{M}\Gamma(c_m + \frac{\alpha}{M})\Gamma(N - c_m + 1)}{\Gamma(N + 1 + \frac{\alpha}{M})}, \end{aligned} \tag{2.17}$$

where $c_m = \sum_{n=1}^{N} z_{nm}$ denotes the number of datapoints that possess feature m. This also shows that the model is exchangeable as its density only depends on the counts c_m.

Equivalence Classes If we naively take the limit $M \to \infty$ we find that any matrix Z has probability 0. This is not a serious problem as we are not interested in the probability of a single matrix Z. More specifically, consider a model with $M = 2$ and let us visualise two different Z matrices



Although technically the left and right Z matrices are different, they encode the same latent feature assignments up to a change of feature indices. It is common to assume that

features are exchangeable in that our probabilistic models do not depend on an ordering of the features. This illustrates that we really only care about feature matrices up to permutations of the columns.

Therefore, we introduce the concept of equivalence classes of binary matrices. We will define a function $lof(\cdot)$ which maps a binary matrix to its left-ordered form. Then we define the equivalence class of a matrix to be all binary matrices which map to the same left-ordered form. $lof(Z)$ can be defined as follows: we interpret each column z_m of Z as encoding the binary number $2^{T-1}z_{1m} + 2^{T-2}z_{2m} + \cdots + z_{Nm}$. We call this number the *history* of the column. We denote with M_h the number of columns in the matrix Z that have history h. $lof(Z)$ is the matrix Z with its columns left to right by decreasing history. Let $[Z]$ denote the set of all matrices that map to the same left-ordered form: we call $[Z]$ the lof-equivalence class. It is easy to check that the number of elements in the lof-equivalence class of Z is equal to $\frac{M!}{\prod_{h=0}^{2^N-1} M_h!}$. Figure 2.12 illustrates a matrix and its left-ordered form.

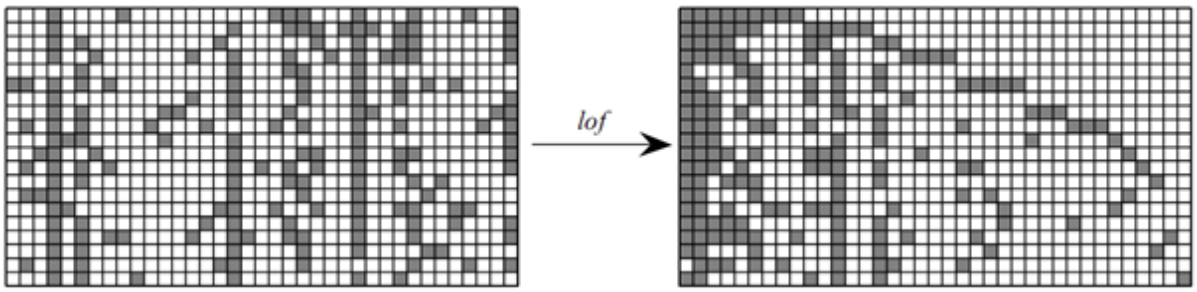


Figure 2.12: An example of a matrix and its left-ordered form.

Before we take the infinite limit we want to point out that lof-equivalence classes play the same role for factor models as partitions for mixture models. E.g. imagine we have a data set with 5 data points labelled a,b,c,d,e: we don't distinguish between assigning a,b,c to cluster 1 and d,e to cluster 2 or a,b,c to cluster 2 and d,e to cluster 1. This is exactly what the partitioning achieves: it only encodes that a,b,c share the same partition and d,e share a different partition.

The Infinite Limit In this final step we take the infinite limit of equation (2.17) for a particular lof-equivalence class and find

$$\begin{aligned} \lim_{M\to\infty} p([\boldsymbol{Z}]|\alpha) &= \lim_{M\to\infty} \sum_{\boldsymbol{Z}\in[\boldsymbol{Z}]} p(\boldsymbol{Z}|\alpha) \\ &= \lim_{M\to\infty} \frac{M!}{\prod_{h=0}^{2^N-1} M_h!} \prod_{m=1}^{M} \frac{\frac{\alpha}{M}\Gamma(c_m + \frac{\alpha}{M})\Gamma(N - c_m + 1)}{\Gamma(N + 1 + \frac{\alpha}{M})}. \end{aligned} \tag{2.18}$$

We divide the columns of Z into two sets: M_0 columns with $c_m = 0$ and M_+ columns with $c_m > 0$ implying that $M = M_0 + M_+$. We break up the product in equation (2.18)

with respect to these two subsets

$$\begin{aligned}
&\prod_{m=1}^{M} \frac{\frac{\alpha}{M}\Gamma(c_m + \frac{\alpha}{M})\Gamma(N - c_m + 1)}{\Gamma(N + 1 + \frac{\alpha}{M})} \\
&= \left(\frac{\frac{\alpha}{M}\Gamma(\frac{\alpha}{M})\Gamma(N+1)}{\Gamma(N+1+\frac{\alpha}{M})}\right)^{M-M_+} \prod_{m=1}^{M_+} \frac{\frac{\alpha}{M}\Gamma(c_m + \frac{\alpha}{M})\Gamma(N - c_m + 1)}{\Gamma(N + 1 + \frac{\alpha}{M})} \\
&= \left(\frac{\frac{\alpha}{M}\Gamma(\frac{\alpha}{M})\Gamma(N+1)}{\Gamma(N+1+\frac{\alpha}{M})}\right)^{M} \prod_{m=1}^{M_+} \frac{\Gamma(c_m + \frac{\alpha}{M})\Gamma(N - c_m + 1)}{\Gamma(\frac{\alpha}{M})\Gamma(N+1)} \\
&= \left(\frac{N!}{\prod_{j=1}^{N} j + \frac{\alpha}{M}}\right)^{M} \left(\frac{\alpha}{M}\right)^{M_+} \prod_{m=1}^{M_+} \frac{(N - c_m)! \prod_{j=1}^{c_m - 1}(j + \frac{\alpha}{M})}{N!}. \qquad (2.19)
\end{aligned}$$

Substituting equation (2.19) back into equation (2.18) and using the results from appendix C, we find the following probability for the equivalence class $[Z]$

$$\lim_{M\to\infty} p([\boldsymbol{Z}]|\alpha) = \frac{\alpha^{M_+}}{\prod_{h=0}^{2^N-1} M_h!} \exp\{-\alpha H_n\} \prod_{m=1}^{M_+} \frac{(N - c_m)!(c_m - 1)!}{N!}, \qquad (2.20)$$

where again H_n is the nth harmonic number.

Connection to the Indian Buffet Process Note the similarity between equation (2.15) from the IBP and equation (2.20) from the infinite factor model. Their likeness suggests that there is a deep connection between both distributions. A natural question to ask is whether the IBP generates left-ordered matrices. As figure 2.12 illustrates, the answer is no. Let us then compute the number of matrices generated by the IBP that map to the same left-ordered form. This is equivalent to computing the opposite: the number of IBP generated matrices that can be generated from one left-ordered matrix. The key observation in counting this number is that given a left-ordered matrix we can permute columns that have the same most significant bit. The reason is that the only constraint on IBP matrices is that they should be sorted by decreasing most significant bit. Since the number of columns with the same significant bit is exactly $M^{(n)}$ (the number of new dishes sampled by customer n), there are exactly $\prod_{n=1}^{N} M^{(n)}!$ such permutations. This overcounts permutations of columns with the same history though. Since there are exactly $M_h!$ permutations for columns with history h, we conclude that

$$\frac{\prod_{n=1}^{N} M^{(n)}!}{\prod_{h=1}^{2^N} M_h!} \qquad (2.21)$$

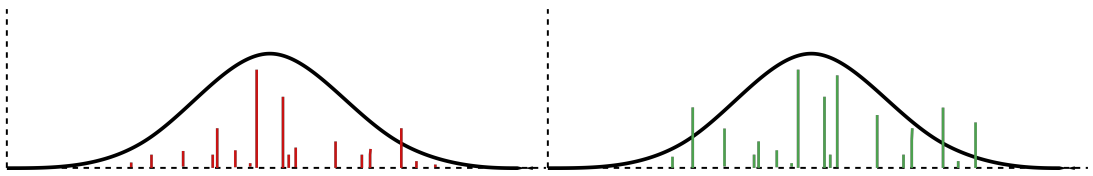
matrices generated by the IBP map to the same left-ordered form. Multiplying equation (2.21) with equation (2.15) we recover the distribution in equation (2.20). In other words, the IBP generates the same distribution over left-ordered equivalence classes as the infinite factor model.

The Beta Process

In the infinite mixture model setting we argued that an exchangeable distribution must have a de Finetti mixing distribution which renders the draws from the mixing distribution conditionally independent. In the previous two sections we discovered that the IBP is exchangeable in the customers. The question thus arises what the de Finetti mixing distribution for the IBP is.

Thibaux and Jordan (2007) showed that a combination of the Beta process and the Bernoulli process is the de Finetti mixing distribution for the IBP. A thorough mathematical description of the Beta process relies on the theory of Lévy processes which we will not review here. Rather, we present an intuitive view which attempts to clarify the connection with Dirichlet processes.

Recall the visualisation of the Dirichlet process which we replicate in the top panel of figure 2.13. A Dirichlet process is a discrete distribution with an infinite number of atoms randomly drawn from a base distribution H. The stick heights represent probabilities and must sum up to one so they represent an infinite multinomial distribution. The sum-to-one condition is natural for mixture models: each data point belongs to one class only, the class drawn from the stick multinomial distribution. In the Beta process we want something slightly different: each data point can posses many out of an infinite number of features. One way to do so is to have an infinite number of atoms drawn independently from the base measure H and for each of the atoms have a stick with length between 0 and 1 to denote the probability that a data point possesses that feature. We do not require the sticks to sum up to one.



(a) A Dirichlet Process (b) A Beta Process

Figure 2.13: Visualisation of the Dirichlet process versus the Beta process. The sum of the sticks in the Dirichlet process is exactly one. The only constraint on the sticks in the Beta process is that they represent probabilities and thus are in the range [0,1]. Both figures use the same base distribution H which is a Gaussian in this case.

We write $B \sim \mathsf{BP}(c, B_0)$ when B is draw from a Beta process over the space Ω with concentration function $c(\cdot)$ and base measure B_0. We require that $c(\cdot)$ is a positive function over Ω and B_0 is a measure over Ω. Technically a Beta process is a Lévy process which means that it has independent increments ($\forall R, S \subseteq \Omega : R \cap S = \emptyset \Rightarrow B(R)$ and

$B(S)$ are independent) and can be characterised by a Lévy measure on $\Omega \times [0, 1]$

$$\nu(d\omega, dp) = c(\omega)p^{-1}(1-p)^{c(\omega)-1}dpB_0(d\omega). \tag{2.22}$$

The intuitive interpretation of this measure is best described as a way to draw B from $\mathsf{BP}(c, B_0)$: draw pairs $(\omega_i, p_i) \in \Omega \times [0, 1]$ from a Poisson process with base measure $\nu(\cdot)$ and let $B(\cdot) = \sum_i p_i \delta_{\omega_i}(\cdot)$. A draw from a Poisson process with base measure $\nu(\cdot)$ on a subset R of the space Ω can be done by first drawing the number of atoms $N \sim \mathsf{Poisson}\left(\int_R \nu(dx)\right)$ and then drawing N atoms i.i.d. from the distribution which we get by normalising $\nu(\cdot)$ (Kingman, 1993). In the context of a Beta process, if $B_0(\cdot) = \sum_i q_i \delta_{\omega_i}(\cdot)$ is discrete then $B(\cdot) = \sum_i p_i \delta_{\omega_i}(\cdot)$ with $p_i \sim \mathsf{Beta}(c(\omega_i)q_i, c(\omega_i)(1-q_i))$. If B_0 is mixed discrete-continuous then B is the sum of two independent contributions.

At this point we have an intuitive picture for a draw from a Beta process: it is an object that represents a countable infinite number of atoms and each atom has a corresponding probability of being on or off. Note that a draw from a Beta process is not a probability distribution since it does not normalise; technically, this is called a measure. Next, we will show how we can "draw" from a Beta process and how this relates to the IBP.

First we need to introduce the notion of a Bernoulli process: we say that X is a draw from a Bernoulli process (we write $X \sim \mathsf{BeP}(B)$) with base measure B (over the space Ω) when

1. If B is continuous then X is a Poisson process with intensity B.
2. If $B(\cdot) = \sum_i p_i \delta_{\omega_i}(\cdot)$ is discrete then $X(\cdot) = \sum_i b_i \delta_{\omega_i}(\cdot)$ where b_i are independent $\mathsf{Bernoulli}(p_i)$ variables.
3. If B is mixed discrete-continuous then X is the sum of the two independent contributions.

We can think of the Bernoulli process as a way to choose which features are on or off. A draw X from a Bernoulli process can be thought of as a data point as it encodes the set of all the features it possesses. One can show that the Bernoulli process and Beta processes are conjugate: if $B \sim \mathsf{BP}(c, B_0)$ and $X_n \sim \mathsf{BeP}(B)$ for all $n \in \{1, \cdots, N\}$ then

$$B|X_{1:N} \sim \mathsf{BP}\left(c+n, \frac{c}{c+n}B_0 + \frac{1}{c+n}\sum_{i=1}^{n} X_i\right). \tag{2.23}$$

We can now describe the connection between the Beta-Bernoulli process and the IBP. Consider the Beta-Bernoulli process $B \sim \mathsf{BP}(c, B_0)$ and $X_n \sim \mathsf{BeP}(B)$ where c is a constant and B_0 is some continuous measure with total mass $B_0(\Omega) = \alpha$. First we compute the marginal distribution of X_1. Since X_1 is a Bernoulli process it is fully specified by its mean. One can show that $\mathbb{E}[X_1] = \mathbb{E}[\mathbb{E}[X_1|B]] = \mathbb{E}[B] = B_0$. Combined

with the fact that B_0 is a continuous distribution we know that X_1 is a mixture of $\mathsf{Poisson}(\alpha)$ number of atoms. This corresponds to the first customer trying a $\mathsf{Poisson}(\alpha)$ number of dishes. For all following customers we apply equation (2.23) and find that

$$\begin{aligned} X_{n+1}|\boldsymbol{X}_{1:n} &\sim \mathsf{BeP}\left(\frac{c}{c+n}B_0 + \frac{1}{c+n}\sum_{i=1}^{n} X_i\right) \\ &= \mathsf{BeP}\left(\frac{c}{c+n}B_0 + \sum_j \frac{n_j}{c+n}\delta_{\omega_j}\right), \end{aligned} \tag{2.24}$$

where n_j is the number of data points for which atom ω_j is on. This means that X_{n+1} is a combination of a discrete and a continuous Beta process. For the discrete part each existing atom will be chosen with probability $n_j/(c+n)$. Hence if we set $c = 1$ this corresponds to the number of existing dishes a new customer tries in the IBP from section 2.3.1. X_{n+1} also gets a contribution from the continuous part where it gets assigned a $\mathsf{Poisson}\left(\frac{\alpha}{1+n}\right)$ number of new dishes. These predictive probabilities exactly match the IBP predictive distribution and hence demonstrates the connection between the IBP and the Beta process.

The Stick Breaking Construction

Our derivations for a nonparametric factor model have followed exactly the same pattern as for the nonparametric mixture model: we first defined a stochastic process, showed how this process is equivalent to the infinite limit of a finite model and then derived the de Finetti mixing distribution. We concluded our discussion in section 2.1 with an explicit stick-breaking representation for the Dirichlet Process. In this section we do the same for the IBP. Note that although we visualised the Beta-Bernoulli process in the previous section using sticks in the space Ω, we did not give an explicit stick breaking construction. This is exactly what we will describe in this section.

Let us denote with $\pi_{(1)} > \pi_{(2)} > \cdots > \pi_{(M)}$ the order statistics of the finite model from section 2.3.1. Teh et al. (2007) showed that the order statistics $\pi_{(m)}$ follow the distribution

$$p(\pi_{(m)}|\pi_{(1:m-1)}) = \alpha \pi_{(m-1)}^{-\alpha} \pi_{(m)}^{\alpha-1} \mathbb{I}(0 \leq \pi_{(m)} \leq \pi_{(m-1)}). \tag{2.25}$$

By introducing a new random variable $\nu_{(m)} = \frac{\pi_{(m)}}{\pi_{(m-1)}}$ and doing a change of variables, the density of $\nu_{(m)}$ turns out to be

$$p(\nu_{(m)}|\nu_{(1:m-1)}) = \alpha \nu_{(m)}^{\alpha-1} \mathbb{I}(0 \leq \nu_{(m)} \leq 1). \tag{2.26}$$

In other words, $\nu_{(m)}$ is independent from $\nu_{(1:m-1)}$ and has a $\mathsf{Beta}(\alpha, 1)$ distribution. From the definition of $\nu_{(m)}$ we know that $\pi_{(m)} = \pi_{(m-1)}\nu_{(m)}$; therefore, expanding out $\pi_{(m-1)}$ we find that $\pi_{(m)} = \prod_{i=1}^{m} \nu_{(i)}$. The construction of $\pi_{(m)}$ can be understood metaphorically

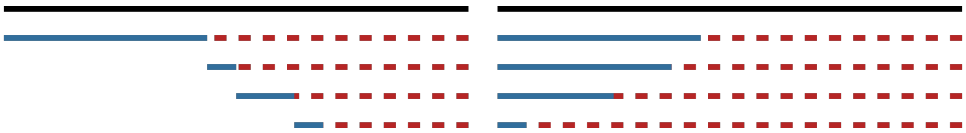


Figure 2.14: Difference between the stick breaking construction of the DP and the IBP. The black line denotes a stick of length 1, the blue lines denote the sticks while the red dotted lines denote the stick remainders after breaking them. Left: the stick breaking construction of the DP, at each step we continue breaking the remainder so that the sum of all stick lengths equals one. Right: the stick breaking construction for the Beta process, at each step we continue breaking the last stick length so that the sticks are monotonically decreasing but do not sum up to one.

as follows: we start with a stick of length 1 and at each iteration we break off a piece $\nu_{(m)} \sim \mathsf{Beta}(\alpha, 1)$ relative to the current length of the stick. We assign the new length to $\pi_{(m)}$ and iterate on the stick of length $\pi_{(m)}$.

Recall for the stick breaking construction for the DP we start with a stick of length one, then draw $\beta_i \sim \mathsf{Beta}(1, \alpha)$ and break off the current stick at relative position β_i, we let π_i be the part we just broke off and iterate on the remaining part. This is equivalent to drawing $\hat{\beta}_i \sim \mathsf{Beta}(\alpha, 1)$, breaking the stick off at relative position $\hat{\beta}_i$, assigning the remainder to π_i and iterating on the part we broke off. Now the two constructions are almost identical: the only difference is that in the IBP stick breaking construction we iterate on the piece we broke off while in the DP we iterate on the remainder. Figure 2.14 illustrates this difference.

2.3.2 Inference

When the IBP was originally introduced, Griffiths and Ghahramani (2006) described a Gibbs sampler for inference; this sampler is particularly convenient for conjugate models. Teh et al. (2007) introduce the stick breaking construction for the IBP. Building on the stick breaking construction they describe two different slice samplers; both not exploiting any conjugacy in the model. Empirical evaluation shows that despite not exploiting conjugacy, the slice sampler has similar performance to the collapsed Gibbs sampler in terms of mixing time. Doshi-Velez and Van Gael (2008) experimented with split-merge MH proposals (Jain and Neal, 2004, Dahl, 2003) for the IBP. Doshi-Velez (2009) introduces a Gibbs sampler for conjugate models which empirically has been found to offer two orders of magnitude faster inference without loss of predictive performance. Wood and Griffiths (2007) describe a particle filter for the IBP and show that it can achieve less error with less computation time. We are aware of only our own work in Doshi-Velez et al. (2009) for deterministic variational inference algorithm for the IBP.

2.3.3 Discussion

The IBP is commonly used in settings where a number of latent features is to be discovered. Griffiths and Ghahramani (2006) use the IBP to learn latent representation of visual images. Titsias (2007) introduces a very similar nonparametric Bayesian factor model based on the conjugacy of the Gamma and Poisson distributions rather than the Beta and Bernoulli conjugacy for the IBP. Wood et al. (2006) develop a prior on top of the IBP to learn an unknown number of hidden causes in a medical setting. Görür et al. (2006) build a nonparametric choice model using the IBP as a prior distribution. Thibaux and Jordan (2007) uses the Beta process and a hierarchical extension of it for improving document classification over Naive Bayes. Wood and Griffiths (2007), Meeds et al. (2007) use the IBP as a prior for binary matrix factorisation where the number of latent factors is unknown. Doshi-Velez and Van Gael (2008) use the IBP for discovering software bugs from trace data where the number of bugs is unknown.

All the models above rely on the properties of the IBP described above: the rows and columns of the IBP matrix are exchangeable. An interesting exception to this distribution is the phylogenetic IBP in Miller et al. (2008a): this model assumes a tree structure on the data points, breaking exchangeability and introducing correlation which is dependent on the distance between data points in the tree. In chapter 5 we construct a different model that breaks exchangeability by assuming data points are related through a Markov process rather than a tree structure.

2.4 Discussion

This concludes our background chapter on nonparametric building blocks. In the next chapters we will extend the infinite capacity mixture model and factor model to the time domain and explicitly describe inference procedures for these nonparametric time series models.

Chapter 3

The Infinite Hidden Markov Model

In this chapter we describe a Bayesian nonparametric HMM on top of the nonparametric distributions from the previous chapter. The Bayesian nonparametric Hidden Markov Model was introduced in Beal et al. (2002) and subsequently called the *infinite hidden Markov model* or iHMM. In this incarnation, the iHMM uses a hierarchy of Polya urn models to represent a generative model which extends the HMM by allowing for an arbitrary large state space. Any finite state sequence $s_{1:T}$ generated by it only visits a finite subset of all states. Beal et al. (2002) showed that the iHMM could be trained using an exact Gibbs sampler with a per iteration complexity of $\mathcal{O}(K^2T^2)$ or an approximate Gibbs sampler with a per iteration complexity of $\mathcal{O}(K^2T)$. In section 3.1.1 we review this construction of the iHMM.

Teh et al. (2006b) re-interpreted the hierarchical Polya urn model in Beal et al. (2002) as a hierarchical Dirichlet process. For this reason, some authors also use the name HDP-HMM for the infinite HMM model. This interpretation embeds the iHMM into the Dirichlet process formalism, leading to a deeper theoretical understanding. The HDP interpretation gives the iHMM new measure theoretic, Chinese Restaurant Franchise and stick breaking representations. Moreover, these new representations lead to an alternative exact Gibbs sampler with a per iteration complexity of $\mathcal{O}(K^2T)$. We review this construction of the iHMM in section 3.1.2.

Although there has been some confusion in the literature as to whether the iHMM and HDP-HMM represent the same model, in section 3.1.3 we formally prove that they do. In section 3.2 we describe fast inference algorithms for the iHMM including two based on dynamic programming based on the work in Van Gael et al. (2008a). We discuss how the iHMM relates to other techniques for automatically learning the number of states in an HMM in section 3.3. Finally, we conclude the chapter with an overview of a number of extensions to the iHMM. This chapter is based on Van Gael et al. (2008a) and previously unpublished results.

3.1 The Infinite Hidden Markov Model

We start our description of the iHMM by introducing the hierarchical Polya urn scheme construction of the iHMM as it was presented in Beal et al. (2002). This scheme makes clear the generative procedure for the hidden representation $s_{1:T}$ and allows us to better understand the effect of the hyper parameters. Section 3.1.2 introduces the HDP-based description of the iHMM. This interpretation will turn out to be useful when designing inference methods for the iHMM.

3.1.1 A Hierarchical Polya Urn Scheme

In section 2.1.1 we reviewed the Polya urn scheme and described how it can generate a clustering of data with an arbitrary large number of clusters. Following Beal et al. (2002), we describe how to extend the Polya urn scheme to use in a time series setting. Consider a countably infinite number of Polya urns with parameter α, one for each possible state the iHMM can be in. We refer to this set as the set of transition urns. In addition to having coloured balls, we also colour the urns and identify both the colours of the urns and balls with states. We draw balls from the hierarchical Polya urn and keep track of their colours: s_t refers to the color of the t'th ball drawn from the hierarchical Polya urn. For bookkeeping purposes, we also keep track of the quantity n_{ij}: the number of balls of colour j in the transition urn with colour i.

Initially, all urns are empty and we assume we observe a dummy ball with arbitrary colour s_0. At each time step t, we record the colour s_{t-1} of the previously drawn ball and then draw according to the Polya urn scheme from the urn with colour s_{t-1}. We set s_t to the colour of the extra ball which we added to urn s_{t-1}. We interpret the number of balls with colour j in the urn with colour i as being proportional to the transition probability of a Markov chain

$$p(s_t = j | s_{t-1} = i, \alpha) = \frac{n_{ij}}{\sum_{j'} n_{ij'} + \alpha}. \tag{3.1}$$

Note that these probabilities do not sum to one: under the Polya urn scheme there is a probability $\frac{\alpha}{\sum_{j'} n_{ij'} + \alpha}$ of drawing a ball outside of the transition urns. To determine the colour of the new ball we introduce an extra *oracle* Polya urn with parameter γ. We denote with c_i the number of balls in the oracle urn with colour i. When we draw a ball from the oracle urn, we record its colour s_t, replace the ball with two balls of the same colour back into the oracle urn and one ball with the same colour to the transition urn with colour s_{t-1}. Formally, when we draw a ball with colour j from the oracle urn we set $c_j \leftarrow c_j + 1$ and $n_{s_{t-1},j} \leftarrow n_{s_{t-1},j} + 1$. Conditional on drawing from the oracle, the

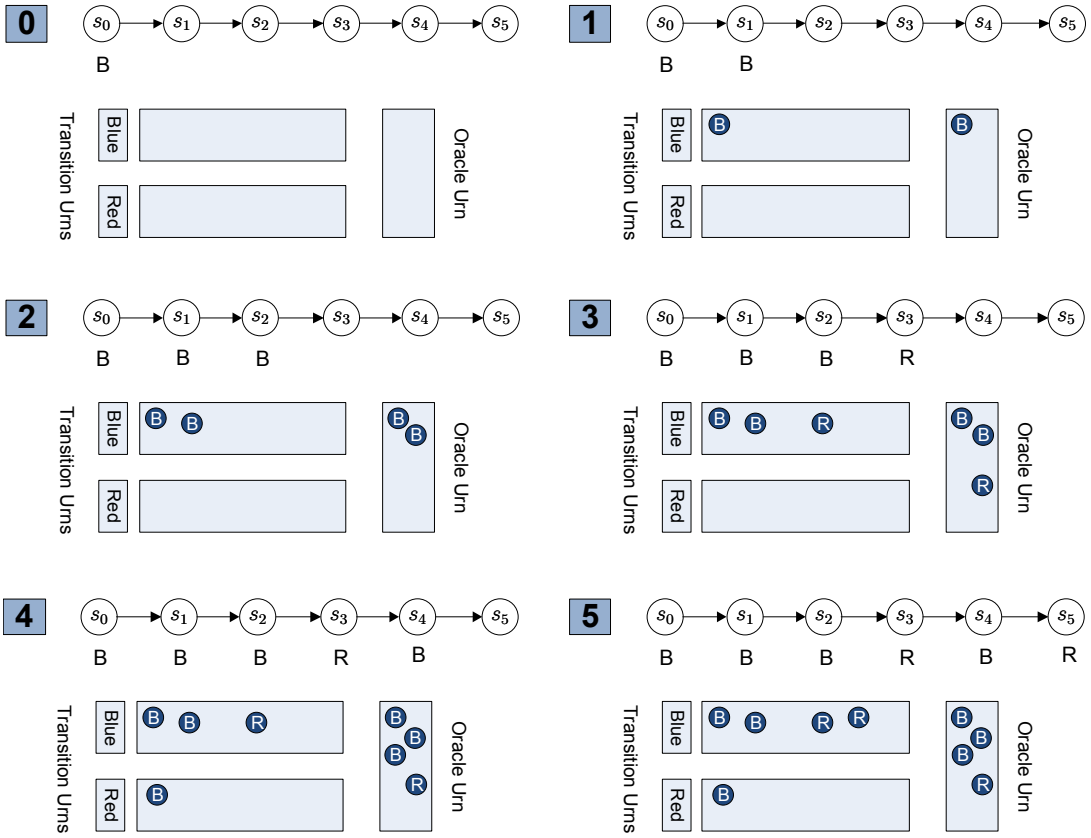


Figure 3.1: Hierarchical Polya Urn Example: we initialise ($t = 0$) our hierarchical Polya urn with both the transition and oracle urns empty and assume our first dummy state carries the color blue. In the first step we check the blue urn and find that there are no balls. This means that with probability $\alpha/\alpha = 1$ we query the oracle. In the oracle urn, the probabilities of drawing a blue ball is $\gamma/\gamma = 1$ so we put a blue ball in both the blue urn and the oracle urn. Next, we generate the state for time step 2: now, with probability $1/(1+\alpha)$ we draw a blue ball and with probability $\alpha/(1+\alpha)$ we query the oracle urn. Imagine we query the oracle then with probability $1/(1+\gamma)$ we draw a blue ball and with probability $\gamma/(1+\gamma)$ a new (red) ball. In our example we draw a blue ball implying that we add an extra blue ball to the oracle and blue transition urns. For time step 3, our example shows that the oracle urn was queried (with probability $\alpha/(2+\alpha)$) and that in the oracle a new red ball was drawn which was added to the blue transition urn as well as to the oracle urn. This means that in time step 4 we first query the red urn and since there are no balls in it, with probability 1 we query the oracle urn; this time returning a blue ball moving us back to the blue urn. Finally in step 5, a red ball is drawn from the blue transition urn without querying the oracle. The resulting sequence of colours *blue,blue,red,blue,red* corresponds to the state sequence $s_1 = 1, s_2 = 1, s_3 = 2, s_4 = 1, s_5 = 2$.

transition probability is

$$p(s_t = j | s_{t-1} = i, \gamma) = \begin{cases} \frac{c_j}{\sum_{j'} c_{j'} + \gamma} & \text{if } j \text{ represents an existing colour,} \\ \frac{\gamma}{\sum_{j'} c_{j'} + \gamma} & \text{if } j \text{ represents a } \textit{new} \text{ colour.} \end{cases} \tag{3.2}$$

In other words, when we query the oracle, with probability proportional to γ we introduce an entirely new colour, otherwise we re-use an existing colour. Re-using existing colours is key to enable the Markov chain to move back to existing states. The combination of transition and oracle urns explains why this scheme is called the *hierarchical Polya urn*[1] scheme: we stack the transition urn and oracle urns on top of each other in a decision hierarchy illustrated in figure 3.2. To conclude the hierarchical Polya urn algorithm, we interpret the s_t as the states of a Markov chain. It is clear that the number of possible states is a random quantity and can grow arbitrarily large. Figure 3.1 illustrates a draw from the hierarchical Polya urn scheme.

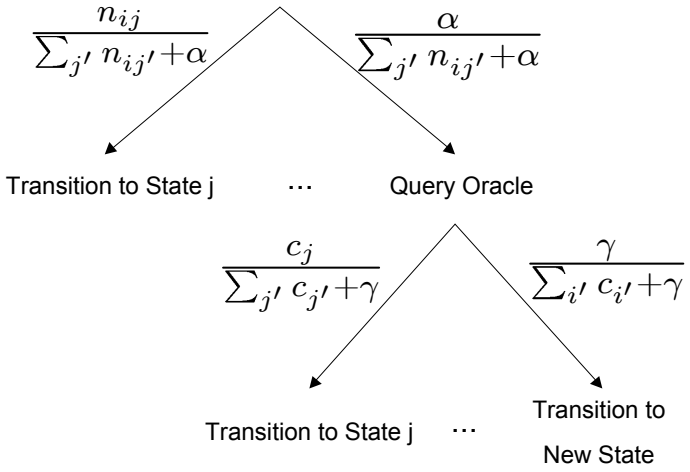


Figure 3.2: Hierarchical Polya urn transition scheme as a decision tree.

The hyper parameters γ and α play an important role in determining the number of states in an iHMM; this is illustrated in figure 3.3. The hyper parameter γ controls how frequently we are going to add a ball with a completely new colour to the urns, in other words, the probability of visiting a new state. The top and bottom plots in figure 3.3 show what happens when we change from $\gamma = 20$ to $\gamma = 2$: the number of states grows much more slowly in the bottom plot than in the top plot. The hyper parameter α on the other hand controls how frequently we query the oracle. The top and middle plots in figure 3.3 illustrate the difference between $\alpha = 20$ and $\alpha = 2$. We see that in the middle plot, once a particular transition accumulates a lot of counts it becomes more

[1] In Beal et al. (2002) this was originally called a hierarchical Dirichlet process but that term has subsequently been used by Teh et al. (2006b) to refer to the closely related Dirichlet process with base measure drawn from another Dirichlet process.

and more likely that we take that transition again. For example, the $4 \to 3$ and $3 \to 4$ transitions are taken frequently. In the limit of $\alpha \to \infty$, we always query the oracle and the distribution of ball colours for each transition urn is going to be very close to the distribution implied by the oracle urn. In this limiting case, we can think of the oracle urn as specifying the expected number of times we visit each state which is the stationary distribution of the Markov chain.

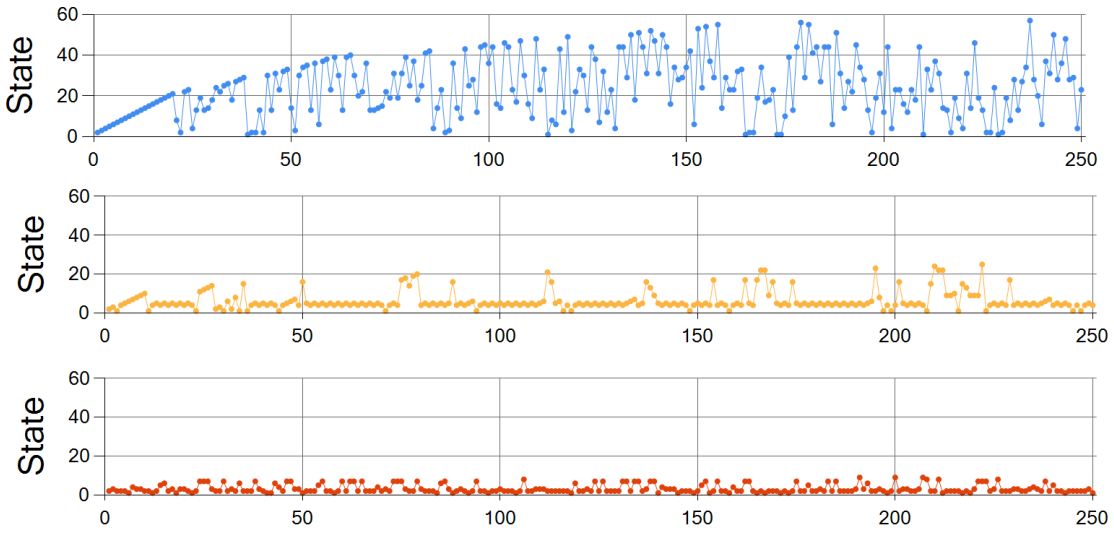


Figure 3.3: Sample Markov chain paths generated by the iHMM. The plots show the state sequence for an iHMM with parameters $\gamma = 20, \alpha = 20$ (top); $\gamma = 20, \alpha = 2$ (middle); $\gamma = 2, \alpha = 20$ (bottom).

Completing the iHMM The hierarchical Polya urn scheme defines a nonparametric distribution on the hidden state sequence $s_{1:T}$. In order to complete it to a full hidden Markov model we need to introduce a mechanism to generate observations conditional on the state sequence $s_{1:T}$. For that we introduce

- a base distribution H from which we draw the output likelihood parameters θ_k for each state $k \in \{1, \cdots, \infty\}$,
- a likelihood model F which takes a state s_t and its parameter θ_{s_t} and generates a data point y_t.

The iHMM presented in Beal et al. (2002) included an additional parameter controlling the rate of self transitions. This idea was revisited by Fox et al. (2008b) as the sticky HDP-HMM and further generalised in Stepleton et al. (2009). We will review these models in section 3.4.3.

3.1.2 The Hierarchical Dirichlet Process

The Polya urn scheme is an intuitive procedure for generating state sequences with arbitrarily large state space. Given the similarities between the Polya urn and the Dirichlet process we might expect that alternative formulations of the iHMM in terms of Dirichlet processes and stick breaking constructions exist. In this section, we review the interpretation of the iHMM using the hierarchical Dirichlet process framework first introduced by Teh et al. (2006b). As we will see in section 3.2, using this representation we will be able to more easily design inference schemes for the iHMM.

One way to look at the distribution defined by an HMM with K states is to write down the marginal distribution of the observation y_t given the previous latent state s_{t-1}

$$p(y_t|s_{t-1}) = \sum_{s_t} p(s_t|s_{t-1})p(y_t|s_t) = \sum_{s_t} \pi_{s_{t-1},s_t} F(y_t;\theta_{s_t}). \tag{3.3}$$

This equation, illustrated in figure 3.4, shows how we can think of the HMM as a dynamic mixture model: our state at time $t-1$ defines a mixture over K emission distributions $F(y_t;\theta_1),\cdots,F(y_t;\theta_K)$ with mixing weights $\pi_{s_{t-1},1},\cdots,\pi_{s_{t-1},K}$.

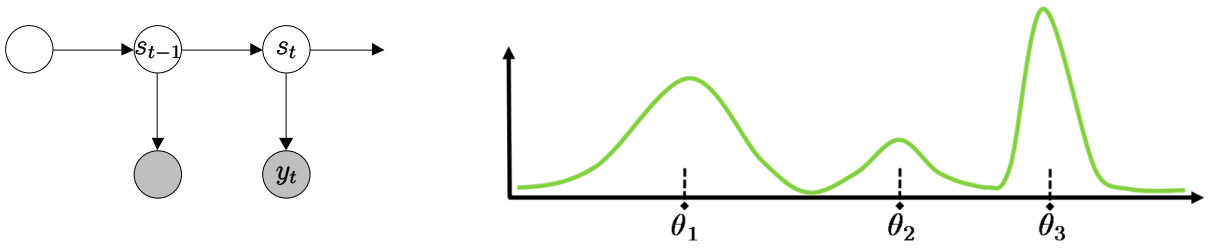


Figure 3.4: The HMM can be thought of as a dynamic mixture model if we look at the marginal distribution of the observation at time t based on the state at time $t-1$. Left: a slice from the HMM corresponding to equation (3.3). Right: the mixture at time t when the output model F is a one dimensional Gaussian distribution.

Building on this insight, we now explore how we can formalise the idea to let $K \to \infty$ to find an infinite state space version of the HMM. The core idea is the following: we introduce the Dirichlet process from section 2.1 to turn the finite dynamic mixture into an infinite mixture. Recall the stick-breaking construction of the DP

$$G(\cdot) = \sum_{k'=1}^{\infty} \beta_{k'} \delta_{\theta_{k'}}(\cdot) \tag{3.4}$$

and note how similar it is to the finite mixture in equation (3.3). If we identify row s_{t-1} from the transition matrix π in equation (3.3) with the stick β in equation (3.4), and the emission parameter θ_{s_t} in equation (3.3) with the atom $\theta_{k'}$ in equation (3.4), then modulo

the emission model F, we can state that $G(\cdot)$ is a infinite capacity version of the finite mixture in equation (3.3).

This similarity applies to one state of the Markov chain. Next, we introduce an infinite number of DP's, one to represent each row of the transition matrix π. Formally, we let $G_k(\cdot) = \sum_{k'=1}^{\infty} \beta_{k,k'} \delta_{\theta_{k,k'}}(\cdot)$ represent the mixture for row k. This introduces a new technical problem though: if we let each $G_k(\cdot)$ be a draw from a DP then almost surely (section 2.2) all atoms $\theta_{k,k'}$ will be different when the base measure is continuous! This is worrisome as equation 3.3 specifies that the emission parameter is only dependent on state s_t and shouldn't depend on state s_{t-1}. In other words, s_{t-1} only influences the mixture weights, not the mixture parameters. Hence, in designing the iHMM, we need to ensure that all $G_k(\cdot)$ share the same atoms: this is exactly what the HDP can do! Formally, the model

$$\begin{aligned} G_0 &\sim \mathsf{DP}(\gamma, H), \\ G_j &\sim \mathsf{DP}(\alpha, G_0), \qquad \forall j \in \{1 \cdots \infty\}, \end{aligned}$$

in the stick-breaking representation will have the form

$$G_0(\cdot) = \sum_{k=1}^{\infty} \beta_k \delta_{\theta_k}(\cdot), \tag{3.5}$$

$$G_j(\cdot) = \sum_{k=1}^{\infty} \pi_{jk} \delta_{\theta_k}(\cdot), \qquad \forall j \in \{1 \cdots \infty\}. \tag{3.6}$$

This is exactly the form we want for an infinite dynamic mixture. Working in the stick-breaking representation allows us to add the hidden Markov chain and observation model: figure 3.5 shows the full model for the iHMM.

$$\begin{aligned} &\forall j \in \{1 \cdots \infty\} \\ &\forall t \in \{1 \cdots T\} \\ \beta &\sim \mathsf{Stick}(\gamma) \\ \pi_{j\cdot} &\sim \mathsf{DP}(\alpha, \beta) \\ \theta_j &\sim H \\ s_0 &= 1 \\ s_t &\sim \pi_{s_{t-1},\cdot} \\ y_t &\sim F(y_t | \theta_{s_t}) \end{aligned}$$

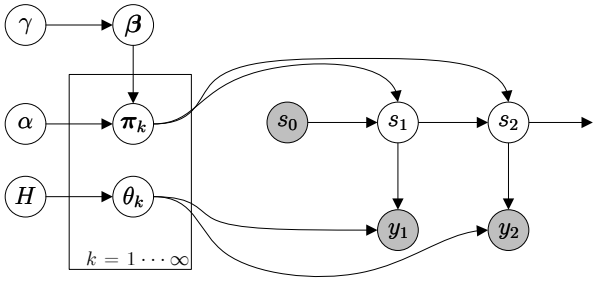


Figure 3.5: The graphical model for the iHMM

First, note that figure 3.5 uses the notation $\pi_{j\cdot} \sim \mathsf{DP}(\alpha, \beta)$ to mean that $\boldsymbol{\pi}_{j\cdot}$ is the stick corresponding to $G_j(\cdot)$ in equation (3.6). Next, observe the striking similarity between the

graphical model for the Bayesian HMM in figure 1.6 and figure 3.5: the only difference between the two models is the introduction of the variable β for the iHMM. This is necessary as the naive limit $K \to \infty$ for the Bayesian HMM would lead to an unusable transition matrix: a draw from a Dirichlet $\left(\frac{\alpha}{K}, \cdots, \frac{\alpha}{K}\right)$ distribution will become sparser with increasing K; in the limit $K \to \infty$, there will be one uniformly chosen random entry with probability mass 1 and all other entries will have mass 0. Hence an infinitely large transition matrix whose rows are of this form will have one unique state per observation.

3.1.3 Hierarchical Polya Urns are Equivalent to Hierarchical Dirichlet Processes

Although it is not immediately obvious, the hierarchical Polya urn construction of the iHMM (Beal et al., 2002) and its HDP interpretation (Teh et al., 2006b) define the same distribution over the Markov chain $s_{1:T}$. In this section we formalise the proof of their equivalence. We will refer to the distribution over $s_{1:T}$ defined by the hierarchical Polya urn scheme as $p_{urn}(s_{1:T})$ and the distribution defined by the hierarchical Dirichlet process as $p_{hdp}(s_{1:T})$.

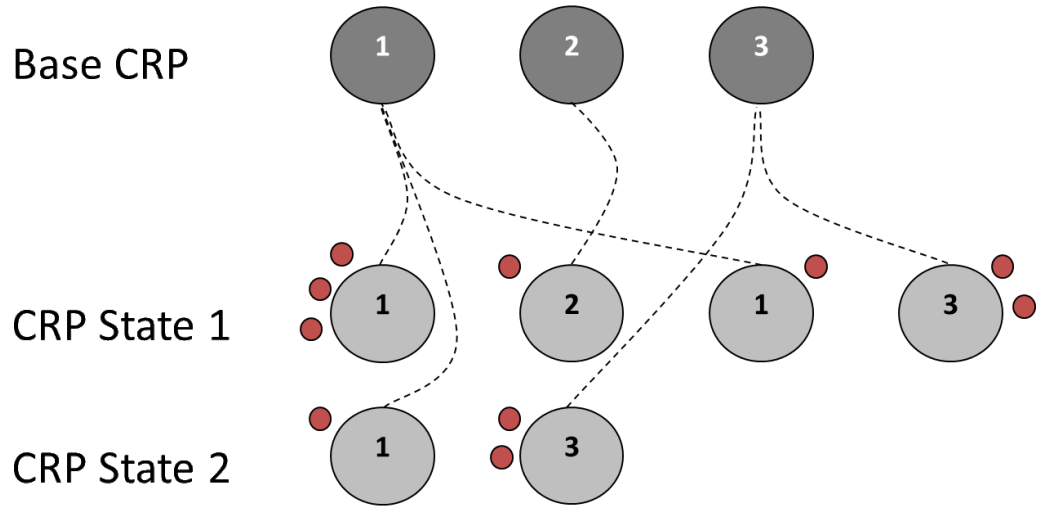


Figure 3.6: CRF visualisation of the HDP underlying the iHMM. The dark grey circles represent the tables in the CRP for the base DP G_0 while the light gray circles represent the tables in the CRP for the child DP's G_j. Each red circle represents a customer sitting at a table in the child CRP's. Each dotted line corresponds to a draw from the base CRP.

In section 2.2 we introduced the Chinese Restaurant Franchise representation of the HDP. From the previous section we know that the iHMM is a special case of the HDP. Since our equivalence proof relies on the Chinese Restaurant Franchise representation of

the HDP, we review how it applies in the context of the iHMM. Figure 3.6 visualises this Chinese restaurant franchise for the iHMM HDP construction. Each transition in the iHMM corresponds to a customer sitting at some table in a child CRP G_j. What makes the CRP representation of the iHMM confusing is that when customer t decides to sit at a new table in a child CRP that doesn't mean that transition t enters a new state. The correct interpretation of equation (3.6) says that when customer t decides to sit at a new table, the child CRP draws the parameter for that new table from the base CRP. Two things can happen: either that parameter is a previously represented parameter in which case transition t goes to a previously represented state; or, the parameter is a completely new parameter in which case transition t moves to a new state. In other words, multiple tables in the child CRP's can represent transitions to the same state. In figure 2.2 the first child CRP has two tables representing the self transition. We will denote the number of customers at the l'th table serving dish j in restaurant i as m_{ijl}. The number of customers sitting at the table serving dish l in the base restaurant will be denoted b_l.

To summarise, we keep track of two different sets of statistics in the equivalence proof:

- for the Polya urn we keep track of the numbers n_{ij} specifying the number of balls in transition urn i of colour j and the number c_i specifying the number of balls of colour i in the oracle urn.

- for the hierarchical Dirichlet process we keep track of the number of customers in the CRF representation (Teh et al., 2006b): m_{ijl} is the number of customers sitting in restaurant i at the l'th table that serves dish j and b_l is the number of customers sitting at the table serving dish l in the base restaurant (or in other words: the number of tables serving dish l in all restaurants together).

Theorem. *The distribution over the sequence of states $s_{1:T}$ defined by the hierarchical Polya urn scheme, $p_{urn}(s_{1:T})$ is the same as the distribution defined by the hierarchical Dirichlet process $p_{hdp}(s_{1:T})$.*

Proof Idea. The core idea of the inductive proof is that the CRF representation keeps track of a bit more information than the Polya urn: in the CRF, m_{ijl} keeps track of who sits at which table serving a particular dish, while in the Polya urn n_{ij} only keeps track of the total number of people who are sitting at a table serving dish j. Since the exact configuration doesn't matter for the predictive probability $p(s_t|s_{1:t-1})$, the two schemes define the same distribution over the sequence $s_{1:T}$. We formalise this idea in the proof below.

Proof. The proof follows an inductive argument. For the base case of the induction we note that $p_{urn}(s_1) = p_{hdp}(s_1) = 1$ because initially there are no customers in the CRF nor are there balls in the Polya urn and hence we can always initialise the first state to be the state with identifier 1. After this assignment, note that we called the oracle once. Hence $c_1 = 1$ and we have set $n_{11} = 1$, analogously, $m_{111} = 1$ since there is one person sitting in the whole CRF and he must be in restaurant 1 eating dish 1 and $b_1 = 1$ since there is only one table serving dish 1 in the whole CRF.

Next we describe the inductive argument: after generating state s_{t-1}, we assume $b_i = c_i$ for *all* i and for *any* configuration $m_{ijl} : n_{ij} = \sum_l m_{ijl}$. Note that we don't assume a particular configuration for m_{ijl} but any configuration that satisfies $n_{ij} = \sum_l m_{ijl}$. We now prove that $p_{urn}(s_t|s_{1:t-1}) = p_{hdp}(s_t|s_{1:t-1})$. If we are in state s_{t-1}, under the Polya urn scheme there are two possibilities: either we query the oracle or we don't. We query the oracle with probability $\frac{\alpha}{\sum_j n_{s_{t-1},j}+\alpha}$ and if we do so, we either go to state i with probability $\frac{c_i}{\sum_k c_k+\gamma}$ or to a new state with probability $\frac{\gamma}{\sum_k c_k+\gamma}$. Under the HDP, the probability that we choose to sit at a new table in restaurant s_{t-1} is $\frac{\alpha}{\sum_{jl} m_{s_{t-1},j,l}+\alpha}$. If we do choose to sit at a new table, we choose an existing dish i with probability $\frac{b_i}{\sum_k b_k+\gamma}$ and a new dish with probability $\frac{\gamma}{\sum_k b_k+\gamma}$. Since choosing a dish in the HDP representation corresponds to transition to a particular state in the iHMM, we can see that under the induction hypothesis conditioned on the fact that we query the oracle (or sit at a new table) the probabilities are exactly the same. Similarly, if under the Polya urn scheme, we decide not to query the oracle, we transition to state i with probability $\frac{n_{s_{t-1},i}}{\sum_j n_{s_{t-1},j}+\alpha}$. Under the CRF, we will sit at a table serving dish i with probability $\frac{\sum_l m_{s_{t-1},i,l}}{\sum_{jl} m_{s_{t-1},j,l}+\alpha}$. Under the induction hypothesis, these two probabilities are again the same. QED.

3.2 Inference

When the iHMM was introduced in Beal et al. (2002) it was presented with two different inference algorithms: a Gibbs sampler with a time complexity of $\mathcal{O}(K^2T^2)$ and an approximate Gibbs sampler with a time complexity of $\mathcal{O}(K^2T)$. The re-interpretation of the iHMM in the HDP framework by Teh et al. (2006b) modified the approximate Gibbs sampler from Beal et al. (2002) and made it exact, whilst retaining the $\mathcal{O}(K^2T)$ complexity. We describe this linear time inference scheme in section 3.2.1.

The linear time Gibbs sampler updates the hidden state at each time step conditioned on all other states in the Markov chain. Typically, the state sequence variables in a Markov chain are strongly correlated. As is well known, Gibbs sampling tends to mix slowly when there are strong correlations between variables. Unfortunately, strong connections are common when modelling sequential data: e.g. when we know a stock's price today we have a good guess about the stock's price tomorrow.

Scott (2002) investigates a Markov chain Monte Carlo method for the Bayesian treatment of the HMM that works around the correlation problem by sampling the whole hidden state sequence at once. We will refer to this dynamic programming based algorithm as the forward-filtering backward-sampling (FF-BS) algorithm. A short summary of the algorithm for finite HMM's can be found in appendix B.

Because the iHMM has a potentially infinite number of states, we cannot naively apply the FF-BS algorithm. In section 3.2.2 we describe the *beam sampler* introduced in Van Gael et al. (2008a). The beam sampler is an auxiliary variable MCMC algorithm which adaptively truncates the iHMM so we can run a variant of the FF-BS algorithm. It re-samples the whole Markov chain at once and suffers less from slow mixing than the Gibbs sampling.

The adaptive truncation of the transition matrix in the beam sampler makes it hard to make exact statements about its complexity; each iteration of the beam sampler might involve a different truncation and hence a different run time. In section 3.2.3 we introduce the embedded HMM sampler for the iHMM. This dynamic programming-based method embeds a finite HMM into the infinite state space of the iHMM to perform inference. As we will prove, this adaptive embedding still allows for exact posterior inference but guarantees a fixed run time cost per iteration.

We conclude the inference section with a brief overview of hyper-parameter learning for the iHMM.

3.2.1 The Collapsed Gibbs Sampler

When we inspect the graphical model in figure 3.5 we see that the unobserved random variables corresponding to the rows of the transition matrix π_k have observed descendants in the graphical model: the observations $y_{1:T}$. Following the standard rules for graphical models we conclude that the π_k are conditionally dependent given $y_{1:T}$. If we observe β and $s_{1:T}$ however, the rows of the transition matrix become conditionally independent. We can consider each transition as a draw from a Polya urn scheme and use our knowledge of Polya urns to re-sample the state sequence. Moreover, if the output distribution F is conjugate to the HDP base distribution H we will show how we can also analytically integrate out the likelihoods involved. One iteration of the collapsed Gibbs sampler will consist of two steps: re-sampling the states $p(s_{1:T}|y_{1:T}, \alpha, \beta, \gamma, H, F)$ and re-sampling the base distribution parameters $p(\beta|y_{1:T}, s_{1:T}, \alpha, \gamma, H, F)$.

Sampling s_t If we condition on β and $s_{1:T}$, the DP's for each of the transitions become independent. Hence, for all $k \in \{1, 2, \cdots\}$ we can treat all the transitions out of state k as draws from a Polya urn. Since they form an exchangeable sequence we can re-sample $s_{1:T}$ by taking out one s_t at a time (essentially removing two transitions) and re-sampling

it from the posterior

$$p(s_t|y_{1:T}, s_{-t}, \alpha, \beta, \gamma, H, F) \propto p(y_t|s_t, s_{-t}, y_{-t}, H, F)p(s_t|s_{-t}, \alpha, \beta, \gamma), \tag{3.7}$$

where s_{-t} denotes all states except s_t. The first factor is the conditional likelihood of y_t given $s_{1:T}, y_{-t}$ and H

$$p(y_t|s_t, s_{-t}, y_{-t}, H, F) = \int p(y_t|\theta_{s_t})p(\theta|s_{-t}, y_{-t}, H)d\theta, \tag{3.8}$$

where y_{-t} denotes all observations except y_t. If the output likelihood F is conjugate to the HDP base distribution H, this quantity can generally be analytically computed.

In order to compute the factor $p(s_t|s_{-t}, \alpha, \beta, \gamma)$, we first use the property that conditional on β the Polya urns for each row are independent. This means that the transition in and the transition out of s_t can be considered draws from a Markov exchangeable sequence (that is, the in-transition is a draw from the exchangeable sequence out of urn s_{t-1} and the out-transition is a draw from the exchangeable sequence out of urn s_t). If we denote with n_{ij}^{-t} the number of transitions from state i to state j excluding the transitions involving time step t, similarly let $n_{\cdot i}^{-t}, n_{i\cdot}^{-t}$ be the number of transitions in and out of state i excluding time step t and K be the number of distinct states in s_{-t}, then we can write

$$p(s_t = k|s_{-t}, \alpha, \beta, \gamma) \propto \begin{cases} (n_{s_{t-1},k}^{-t} + \alpha\beta_k)\frac{n_{k,s_{t+1}}^{-t} + \alpha\beta_{s_{t+1}}}{n_{k\cdot}^{-t} + \alpha} & \text{if } k \leq K,\ k \neq s_{t-1} \\ (n_{s_{t-1},k}^{-t} + \alpha\beta_k)\frac{n_{k,s_{t+1}}^{-t} + 1 + \alpha\beta_{s_{t+1}}}{n_{k\cdot}^{-t} + 1 + \alpha} & \text{if } k = s_{t-1} = s_{t+1} \\ (n_{s_{t-1},k}^{-t} + \alpha\beta_k)\frac{n_{k,s_{t+1}}^{-t} + \alpha\beta_{s_{t+1}}}{n_{k\cdot}^{-t} + 1 + \alpha} & \text{if } k = s_{t-1} \neq s_{t+1} \\ \alpha\beta_k\beta_{s_{t+1}} & \text{if } k = K + 1. \end{cases} \tag{3.9}$$

We re-sample the whole state sequence $s_{1:T}$ by re-sampling each s_t in turn conditional on all other states.

Sampling β Recall that β is the distribution on the atoms for the base distribution of the HDP. If we would have kept track of the Polya urn representation of the base distribution (that is, the counts of each ball colour c_i in the oracle urn) then we could re-sample from the posterior $\beta \sim \mathsf{Dirichlet}(c_1, c_2, \cdots, c_K, \gamma)$. Although we don't have the representation $c_{1:K}$ we can introduce an auxiliary variable m_{ij} with the following interpretation: m_{ij} denotes the number of oracle calls that returned a ball with colour j when we queried the oracle from state i. The total number of colour j balls in the oracle urn equal $\sum_i m_{ij}$: hence, if we can sample m_{ij} then we have implicitly sampled $c_j = \sum_i m_{ij}$ and hence we can re-sample β.

First note that if the number of transitions from state i to j is nonzero ($n_{ij} > 0$) then we know that we must have queried the oracle at least once when we were in state i.

Moreover, we also know that $m_{ij} \leq n_{ij}$ since we can query the oracle at most once for every transition. Let S_{ij} denote the set of transitions from i to j, $S_{ij} = \{s_k | s_k = i \wedge s_{k+1} = j\}$. The exchangeability of the transitions out of state i implies that the sequence S_{ij} is exchangeable as well. Note that m_{ij} is the number of elements in S_{ij} that were gotten from querying the oracle. Because S_{ij} is an exchangeable sequence, for each $s_k \in S_{ij}$ we can write down the conditional distribution

$$p(s_{k+1} = j | S_{ij}^{-s_{k+1}}) \propto \begin{cases} n_{ij}^{-s_{k+1}} - 1 & \text{if we don't query the oracle,} \\ \alpha\beta_j & \text{if we query the oracle.} \end{cases} \tag{3.10}$$

These equations form the conditional distribution of a Gibbs sampler whose equilibrium distribution is the distribution of a Polya urn scheme with parameter $\alpha\beta_j$. In other words, in order to sample m_{ij}, we sample n_{ij} elements from a Polya urn with parameter $\alpha\beta_j$ and count the total number of newly chosen ball colours[2].

Complexity We can break down the computational complexity for an iteration where the sample uses at most K states in two different parts. When sampling the hidden state sequence, for each time step $1 \leq t \leq T$ we need to compute $\mathcal{O}(K)$ probabilities; this results in a $\mathcal{O}(TK)$ contribution to the computational cost. On the other hand, when sampling the β variables, for each of the K^2 entries of the transition matrix, we need to sample n_{ij} random variables. Since $\sum_{ij} n_{ij} = T$ this leads to an extra $\mathcal{O}(K^2 + T)$ complexity. In summary, one iteration of the collapsed Gibbs sampler has an $\mathcal{O}(TK + K^2)$ computational complexity. This is the same computational complexity as the Gibbs sampler for the finite HMM with K states; note that for the iHMM K can vary between iterations though.

Discussion First, note that non-conjugate models can be handled using the sampling methods in Neal (2000). An example can be found in Van Gael et al. (2008a) who experimented with a collapsed Gibbs sampler with normal base distribution and Student-t distributed likelihood for a change point detection problem.

As we discussed above, the collapsed Gibbs sampler suffers from one major limitation: sequential data are likely to be strongly correlated. In other words, it is unlikely that the Gibbs sampler which makes individual updates to s_t can cause large blocks within $s_{1:T}$ to change state. Scott (2002) showed how dynamic programming based methods which compute and sample from the full joint distribution over $s_{1:T}$ for finite HMM's do not suffer from this slow mixing. However, naively applying dynamic programming to the iHMM is impossible due to its infinite state space. In what follows we describe a number of proposals to make dynamic programming work for the iHMM.

[2]Thanks to Emily Fox for bringing this to our attention; a previous implementation computed Stirling numbers of the second kind which are rather expensive to compute for large scale applications.

3.2.2 The Beam Sampler

We can improve on the collapsed Gibbs sampler by introducing a method that re-samples the whole state sequence of the iHMM at once. Methods that achieve this for finite HMM's, e.g. the forward-filtering backward-sampling algorithm reviewed in appendix B, efficiently enumerate all possible state trajectories (using dynamic programming), compute their corresponding probabilities and sample from this set. Unfortunately, the forward-filtering backward-sampling algorithm does not apply to the iHMM because the number of states, and hence the number of potential state trajectories, is infinite.

The core idea of the beam sampling in Van Gael et al. (2008a) is to introduce auxiliary variables $u_{1:T}$ such that conditioned on $u_{1:T}$ the number of trajectories with positive probability is finite. Then, dynamic programming can be used to compute the conditional probabilities of each of these trajectories and efficiently sample *whole* state trajectories. These auxiliary variables do not change the marginal distribution over other variables and hence MCMC sampling still converges to the true posterior. The idea of using auxiliary variables to limit computational cost is inspired by Walker (2007) who applied it to limit the number of components in a DP mixture model that need to be considered during sampling.

Algorithm 3 The beam sampler for the iHMM.

Initialise s, β randomly.
loop
  Sample the transition matrix parameters $\pi | s_{1:T}, \beta$
  Sample the auxilary variables $u_{1:T} | s_{1:T}, \pi$
  Sample the emission parameters $\theta | s_{1:T}, y_{1:T}, H$
  Sample the state sequence $s_{1:T} | u_{1:T}, \pi, y_{1:T}, \theta, \beta$
  Sample the base DP $\beta | s_{1:T}, \alpha, \gamma$
end loop

Sampling π and θ Since the auxiliary variable and dynamic program below need the transition and likelihood model parameters, we can think of them as another set auxiliary of variables which we sample just prior to sampling $u_{1:T}$. Let n_{ij} be the number of times state i transitions to state j in the state sequence $s_{1:T}$, where $i, j \in \{1 \dots K\}$, K is the number of distinct states in $s_{1:T}$, and these states have been re-labeled $1 \dots K$. Merging the infinitely many states not represented in $s_{1:T}$ into one state, the conditional distribution of $(\pi_{k1} \dots \pi_{kK}, \sum_{k'=K+1}^{\infty} \pi_{kk'})$ given its Markov blanket $s_{1:T}, \beta$, α is

$$\mathsf{Dirichlet}(n_{k1} + \alpha\beta_1 \dots n_{kK} + \alpha\beta_K, \alpha \textstyle\sum_{i=K+1}^{\infty} \beta_i).$$

Each θ_k is independent of others conditional on $s_{1:T}$, $y_{1:T}$ and their prior distribution H, i.e. $p(\theta|s_{1:T}, y_{1:T}, H) = \prod_k p(\theta_k|s_{1:T}, y_{1:T}, H)$. When the base distribution H is conjugate to the data distribution F, each θ_k can generally be sampled efficiently. Otherwise Metropolis-Hastings or other approaches may be applied. Note that for beam sampling in the non-conjugate case this is simpler than for Gibbs sampling: we only need to *sample* from a non-conjugate posterior rather than use Monte Carlo integration to compute the integral in equation (3.8).

Sampling $u_{1:T}$ For each time step t we introduce an auxiliary variable u_t with conditional distribution $u_t \sim \mathsf{Uniform}(0, \pi_{s_{t-1}s_t})$. This very simple step lies at the core of the beam sampling idea. By first sampling auxiliary variables uniformly between 0 and the state transition probability $\pi_{s_{t-1}s_t}$, when we later sample the state sequence $s_\cdot$ conditional on the $u_\cdot$, it must be the case that $0 \leq u_t \leq \pi_{s_{t-1}s_t}$ for all s_t. There are only a finite set of state transitions for which this property can hold as we will show in the paragraph below. This essentially reduces the infinite dimensional state space to a finite dimensional one.

Sampling $s_{1:T}$ The key observation is the distribution over state sequences $s_{1:T}$ conditional on the auxiliary variables $u_{1:T}$ will only have non-zero probability where $\pi_{s_{t-1}s_t} \geq u_t$. Other paths are not consistent with the conditional distribution $p(u_t|s_{1:T}, \pi)$. Moreover, there are only finitely many such trajectories with this property: since $\pi_{kk'} > 0$ with probability 1, $u_t > 0$ with probability 1; given the auxiliary variable u_t, note further that for each possible value of s_{t-1}, u_t partitions the set of transition probabilities out of state s_{t-1} into two sets: a finite set with $\pi_{s_{t-1}k} > u_t$ and an infinite set with $\pi_{s_{t-1}k} < u_t$. Thus we can recursively show that for $t = 1, 2 \ldots T$ the set of trajectories $s_{1:t}$ with all $\pi_{s_{t'-1}s_{t'}} > u_t$ is finite. The math below makes this more clear and will also show how dynamic programming can compute the conditional distribution over all such trajectories.

First note that the probability density for u_t is

$$p(u_t|s_{t-1}, s_t, \pi) = \frac{\mathbb{I}(0 < u_t < \pi_{s_{t-1},s_t})}{\pi_{s_{t-1},s_t}}, \tag{3.11}$$

where $\mathbb{I}(C) = 1$ if condition C is true and 0 otherwise. We compute $p(s_t|y_{1:t}, u_{1:t})$ for all

time steps t as follows (we omit the additional conditioning variables π and θ for clarity):

$$\begin{aligned} & p(s_t|y_{1:t}, u_{1:t}) \\ & \propto \; p(s_t, u_t, y_t|y_{1:t-1}, u_{1:t-1}), \\ & = \; \sum_{s_{t-1}} p(y_t|s_t)p(u_t|s_t, s_{t-1})p(s_t|s_{t-1})p(s_{t-1}|y_{1:t-1}, u_{1:t-1}), \\ & = \; p(y_t|s_t) \sum_{s_{t-1}} \mathbb{I}(u_t < \pi_{s_{t-1},s_t})p(s_{t-1}|y_{1:t-1}, u_{1:t-1}), \\ & = \; p(y_t|s_t) \sum_{s_{t-1}:u_t<\pi_{s_{t-1},s_t}} p(s_{t-1}|y_{1:t-1}, u_{1:t-1}). \end{aligned} \tag{3.12}$$

Note that we only need to compute (3.12) for the finitely many s_t values belonging to some trajectory with positive probability. Furthermore, although the sum over s_{t-1} is technically a sum over an infinite number of terms, the auxiliary variable u_t truncates this summation to the finitely many s_{t-1}'s that satisfy both constraints $\pi_{s_{t-1},s_t} > u_t$ and $p(s_{t-1}|y_{1:t-1}, u_{1:t-1}) > 0$. Finally, to sample the whole trajectory $s_{1:T}$, we sample s_T from $p(s_T|y_{1:T}, u_{1:T})$ and perform a backward pass where we sample s_t given the sample for s_{t+1}: $p(s_t|s_{t+1}, y_{1:T}, u_{1:T}) \propto p(s_t|y_{1:t}, u_{1:t})p(s_{t+1}|s_t, u_{t+1})$.

Sampling β We discard the transition matrix π and sample β using the same algorithm as the Gibbs sampler in section 3.2.1.

Complexity For each time step and each state assignment we need to sum over all represented previous states. If K states are represented, the worst case complexity is $\mathcal{O}(TK^2)$. However, the sum in (3.12) is only over previous states for which the transition probability is larger than u_t. In practice this means that we might only need to sum over many fewer states.

Figure 3.7 shows the empirical complexity of the beam sampler on the application which we introduce in chapter 4. As we can see from the plot, initially the beam sampler considers about $K^2/2$ transitions per state but very quickly this number decreases to about 10% of K^2. This means that the beam sampler will run approximately 10 times faster than the dynamic program which considers all transitions. This increase doesn't come for free: by considering fewer potential transitions, the chain might mix more slowly.

Qualitative Comparison Qualitatively, the beam sampler typically outperforms the Gibbs sampler. Although it is quite hard to compute the mixing time of a complex Mote Carlo method; in this section we evaluate the beam sampler on two artificial and two real datasets to illustrate the following properties: (1) the beam sampler mixes in much fewer iterations than the Gibbs sampler; (2) the beam sampler mixes well regardless of

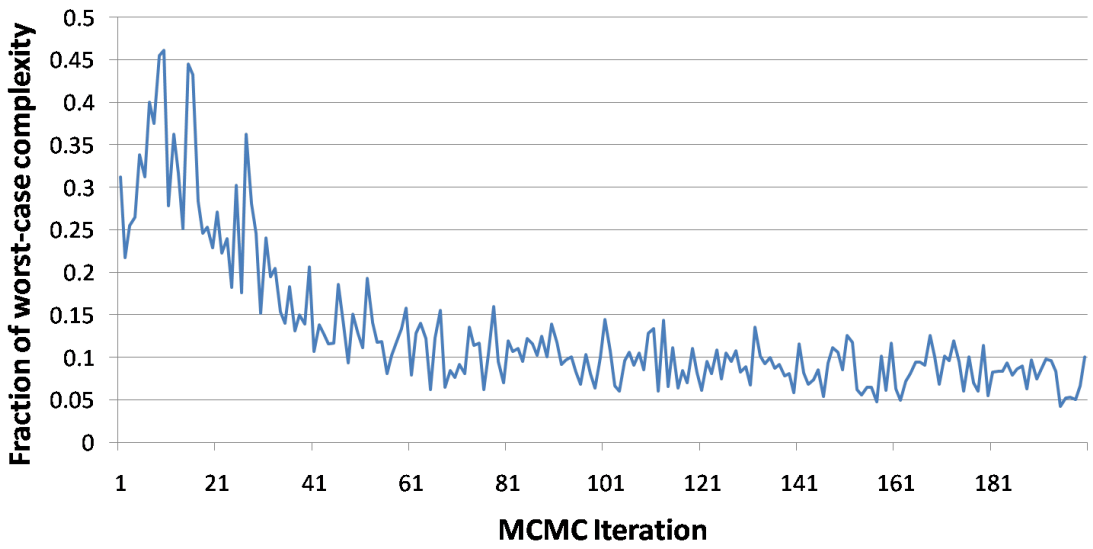


Figure 3.7: This plot shows an example of how efficient the beam sampler is in practice. On the x-axis we plot the iteration; on the y-axis we plot the fraction of the worst case complexity per state. Recall that if the sampler currently represents K states, the worst case complexity for a particular state will be K^2.

strong correlations in the data and (3) the beam sampler is more robust with respect to varying initialization and prior distribution.

Our first experiment compares the performance of the iHMM on a sequence of length 800 generated by a 4 state HMM. The hidden state sequence was almost cyclic (1-2-3-4-1-2-3-...) with a 1% probability of self transition: i.o.w the true distribution of hidden states is strong negatively correlated. We use a multinomial output distribution with the following emission matrix

$$\begin{bmatrix} 0.0 & 0.5 & 0.5 \\ 0.6666 & 0.1666 & 0.1666 \\ 0.5 & 0.0 & 0.5 \\ 0.3333 & 0.3333 & 0.3333 \end{bmatrix}.$$

Next we run the Gibbs and beam sampler 20 times from a random initialization with every state randomly chosen between 1 and 20. We test the performance of both samplers using three different hyperparameter settings: (1) vague gamma hyperpriors for α and γ ($\mathsf{Gamma}(1,1)$ and $\mathsf{Gamma}(2,1)$ respectively); (2) strong gamma hyperpriors for α and γ ($\mathsf{Gamma}(6,15)$ and $\mathsf{Gamma}(16,4)$ respectively); (3) fixed hyperparameters $\alpha = 0.4, \gamma = 3.8$. The latter were chosen using the values the beam and Gibbs samplers converged to. At every iteration, we greedily compute an assignment of sample states to true states to maximize overlap and use the resulting Hamming distance as our error measure. The top plot in figure 3.8 clearly shows that the beam sampler discovers the underlying structure

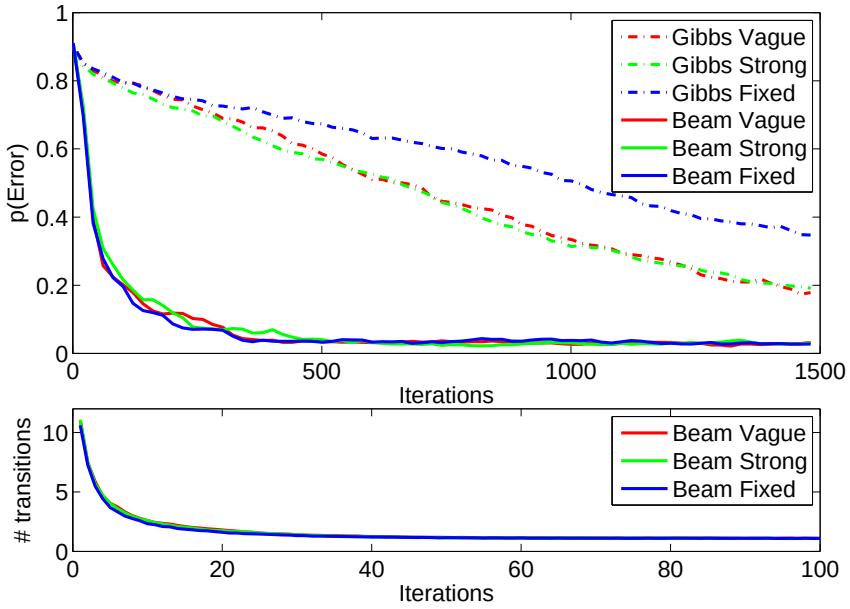


Figure 3.8: iHMM performance on strong negatively correlated data. The top plot shows the error of the Gibbs and beam sampler for the first 1500 iterations averaged over 20 runs. The bottom plot shows the average number of previous states considered in equation (3.12) for the first 100 iterations of the beam sampler.

much faster than the Gibbs sampler. Also, the beam sampler is insensitive to the prior while the performance of the Gibbs sampler becomes worse as we strengthen our prior beliefs. The bottom plot of figure 3.8 shows how many states are summed over in the dynamic program of the beam sampler averaged per timestep, per state. This reinforces our complexity discussion from the previous paragraph: we find that after only about 20 iterations, the beam sampler on average considers a little more than one state suggesting that the actual complexity of the beam sampler is closer to $\mathcal{O}(TK)$ rather than the worst case complexity of $\mathcal{O}(TK^2)$.

Our second experiment illustrates the performance of the beam sampler on data generated from HMM's with increasing positive correlation between the hidden states. We generated sequences of length 4000 from a 4 state HMM with self-transition probabilities increasing from 0.75 to 0.95 and finally 0.999. In one experiment (top plot of figure 3.9) we generated normal distributed observation from an informative output model with means $-2.0, 4.0, 1.0, -0.5$ and standard deviation 0.5, in another experiment (bottom plot of figure 3.9) we generated normal distributed observations from a less informative output model with means $-1.0, 0.5, -0.5, 0.0$ and standard deviation 0.5. We initialize the experiment as above and set the base distribution for the state means to be a 0 mean normal with 2.0 standard deviation. Then, we greedily compute the error compared to

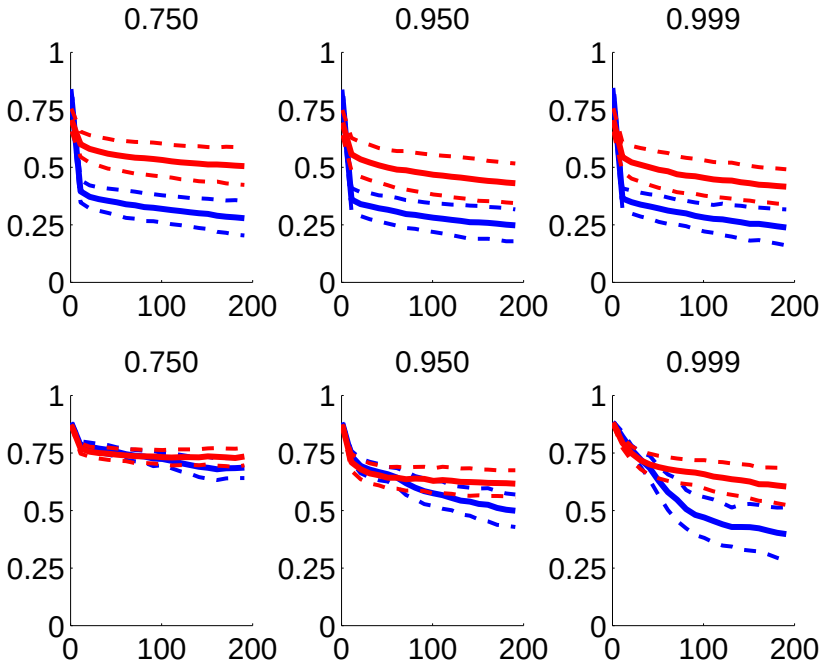


Figure 3.9: iHMM error on increasing positively correlated data. The blue curve shows the beam sampler while the red curve shows the Gibbs sampler performance. The dotted line show the one standard deviation error bars.

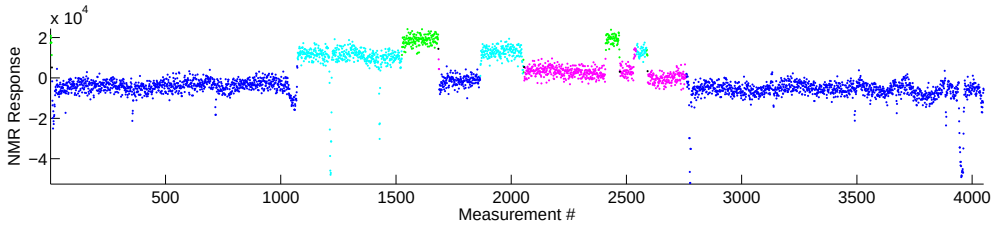


Figure 3.10: The 40'th sample of the beam sampler with every state represented by a different color on the well-log dataset.

ground truth and average the results over 60 different random starting positions. The top row shows that with an informative prior, both the Gibbs and beam sampler can reduce the initial error by at least 50% independent of the correlation between hidden states. When the output model is less informative however and there is little correlation between the hidden states, the learning problem is hardest: the lower left plot shows that both the beam and Gibbs sampler discover structure only slowly. When the correlation increases, the learning problem should become easier. However, as the lower right plot shows, although the beam sampler mixes increasingly well, the Gibbs sampler suffers from slow random walk behavior.

The next experiment illustrates the performance of the iHMM on a changepoint de-

tection problem. The data consists of 4050 noisy measurements of nuclear-response of rock strata obtained via lowering a probe through a bore-hole. Figure 3.10 illustrates this datasets. The data has been previously analyzed in Ruanaidh and Fitzgerald (1996) by eliminating the forty greatest outliers and running a changepoint detection algorithm with a fixed number of changepoints. This approach works well as this one-dimensional dataset can be inspected visually to make a decision on whether to throw away datapoints and get a rough idea for the number of changepoints. However, we believe that with a nonparametric model, we can automatically adapt the number of changepoints. Moreover, by setting up a noise model with fat tails, we hope to automatically handle the outlier problem.

We model the mean of the nuclear-response for every segment. First we normalize the data to have zero mean; then we specify a zero mean normal distribution for the base distribution H. We choose the variance of this normal to be the empirical variance of the dataset. For the output model, we let F correspond to a Student-t distribution with $\nu = 1$, also known as the Cauchy distribution. We set the scale parameter for the Cauchy distribution to twice the empirical standard deviation for the dataset. Since the Cauchy likelihood is not conjugate with respect to the normal base distribution, we modified the Gibbs sampler based on algorithm 8 in Neal (2000). We use the auxiliary variable sampling scheme discussed in Gelman et al. (1995) to resample the segment means.

Figure 3.10 shows the results of one sample from the beam sampler: the iHMM segments the dataset reasonably well and robustly handles the outliers. To compare the Gibbs and beam samplers, we compute 50 samples after a burnin of 5000 iterations with 1000 iterations in between each sample. For every pair of datapoints we compute the probability that they are in the same segment, averaged over the first five samples (left plots in figure 3.11) and the last thirty samples (right plots in figure 3.11). First, note that after the first 10000 iterations, the Gibbs sampler hasn't discovered any structure while the beam sampler has. This supports our claim that the beam sampler mixes faster than the Gibbs sampler. Moreover, we expect that the Gibbs sampler will have trouble to reassign the state assignment for whole segments because of slow random walk behavior. The beam sampler on the other hand resamples whole hidden state sequences and should be able to reassign whole segments more easily. The right plots of figure 3.11 confirm our expectation: a careful inspection of both plots shows that the Gibbs sampler is visually more black-white indicating that either two datapoints are always in the same cluster or never in the same cluster; the beam sampler, on the other hand, has gray areas which indicate that it averages over different assignments of the segments: e.g. the Gibbs plot (upper right) suggests that the leftmost segment and rightmost segment are *always* in the same state, while the beam sampler plot (bottom right) indicates that only part of the time, the left and rightmost segments are in the same state (90% of the time).

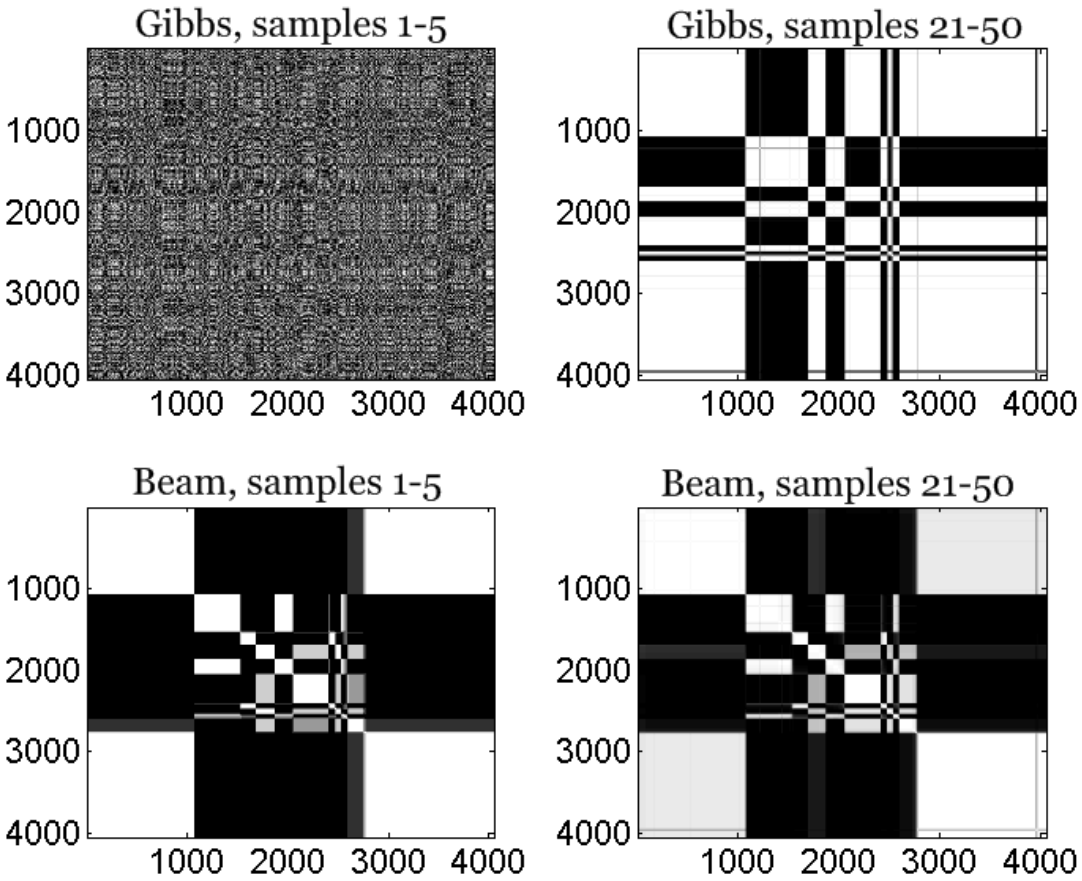


Figure 3.11: The left plots show how frequent two datapoints were in the same cluster averaged over the first 5 samples. The right plots show how frequently two datapoints were in the same cluster averaged over the last 30 samples.

Discussion A practical point of attention is that after sampling u_t, we need to make sure that *all* possible transition probabilities in $\pi_{s_{t-1}} > u_t$ are represented. This might involve extending our representation of $\pi_{s_{t-1}}$ and θ with new states sampled from their posterior distribution. Since there are no observations associated with these new states, the posterior distribution equals the prior distribution.

To conclude our discussion of the beam sampler, it is useful to point out that u_t need not be sampled from the uniform distribution on $[0, \pi_{s_{t-1},s_t}]$. We can choose $u_t \sim \pi_{s_{t-1}s_t}\mathsf{Beta}(a,b)$ which with the appropriate choice of a and b could bias our auxiliary variable closer to 0 or closer to $\pi_{s_{t-1}s_t}$. Smaller auxiliary variables imply that we consider more transitions in the dynamic program, hence mixing can be faster, at a larger computational cost. Larger auxiliary variables imply that the dynamic program is sparser, hence mixing can be slower, at a smaller computational cost.

3.2.3 The Embedded HMM Sampler

Neal et al. (2004) introduced the embedded HMM to do inference in nonlinear dynamical systems. Their idea was to first embed an HMM into the space of the dynamical system by assigning HMM states to a sampled discretisation of the space. Then, one runs a forward-filtering backward-sampling iteration on the HMM to re-sample the state sequence. Since the discretisation of the space is not chosen beforehand but rather adaptively sampled one can prove that the sampled state sequences are exact samples from the posterior. In other words, the embedded HMM for nonlinear dynamical systems addresses the problem that there are an uncountable infinite number of possible hidden states by restricting inference at each iteration to a finite HMM.

Exact inference for the iHMM faces a similar problem: instead of an uncountable infinite number of states, there are a countable infinite number of states. Nonetheless we can apply the same strategy: at each time step we sample a set of states which we call the active pool of states. For reasons that will become clear the current state sequence will always be part of the active pool. Then we sample the state sequence by running a slightly modified version of the forward-filtering backward-sampling algorithm that is restricted to states in the active pool. This idea is illustrated in figure 3.12.

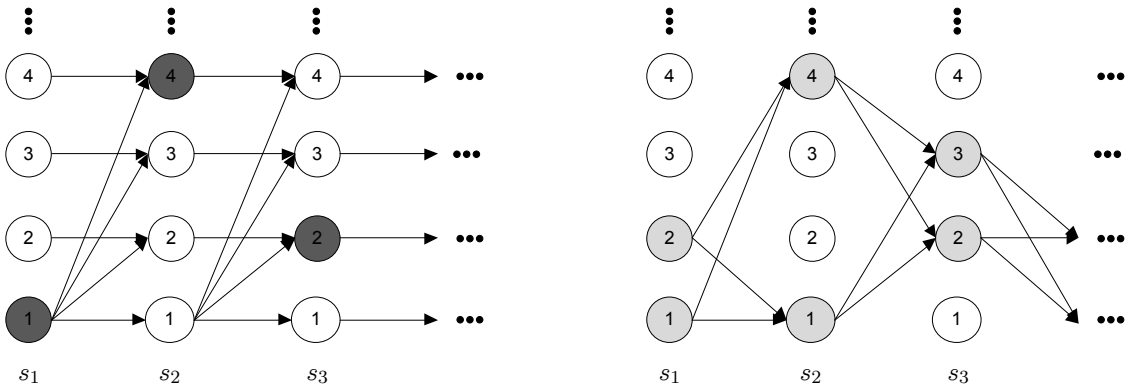


Figure 3.12: Trellis illustration of the active pool. In both pictures, the trellis for the iHMM is shown. The three vertical dots indicate that there are an infinite number of states which aren't drawn. The horizontal dots indicate that the sequence has length T. Note that not all possible transition arcs are draw for clarity but remember that the graph connects all states between adjacent time steps where the transition matrix is nonzero. Left: The state sequence $s1 = 1; s2 = 4; s3 = 2; ...$ is highlighted in dark grey. Right: A sample of an active pool with $P = 2$ is highlighted in light grey. Note that the current state sequence (dark highlight in left picture) is always part of the active pool.

There is only one parameter in the inference algorithm: the size P of the active pool.

Algorithm 4 outlines the embedded HMM sampler for the iHMM.

Algorithm 4 The embedded HMM sampler for the iHMM.

Initialise s, β randomly.

loop

Sample the transition matrix parameters $\pi|s_{1:T}, \beta$

Sample the active pool $C|s_{1:T}, \beta$

Sample the state sequence $s_{1:T}|C, \pi, y_{1:T}, \theta, \beta$

Sample the base DP $\beta|s_{1:T}, \alpha, \gamma$

Sample the emission parameters $\theta|s_{1:T}, y_{1:T}, H$

end loop

If we compare the beam sampler (algorithm 3) and the embedded HMM sampler (algorithm 4), we find that the algorithms are exactly the same except for the steps that re-sample the state sequence. We refer to section 3.2.2 for the description of the common steps and focus here on how the embedded HMM sampler re-samples state sequences.

Sampling $C|s_{1:T}, \beta$ The recipe for sampling the active pool is extremely simple: for each time step t, we set the first state in the pool equal to the current state at time t: $C_{t1} = s_t$. Then for $p \in \{2 \cdots P\}$ we sample a stick index i from β and set $C_{tp} = i$. Note that this allows for two $C_{t\cdot}$'s to be equal. This step has $\mathcal{O}(TP)$ computational complexity.

Sampling $s_{1:T}|C, \pi, y_{1:T}, \theta, \beta$ We re-sample the state sequence $s_{1:T}$ from the posterior distribution[3]

$$\frac{\prod_{t=1}^{T} p(y_t|s_t) \prod_{t=1}^{T} p(s_t|s_{t-1})}{p(y_{1:T}) \prod_{t=1}^{T} \beta_{s_t}} \tag{3.13}$$

where each state is constrained to be in the active pool: $s_t \in \boldsymbol{C}_{t\cdot}$. This is the standard posterior distribution reweighting the states by the probability that they were sampled into the active pool.

Sampling from this posterior distribution can be done using the forward-filtering backward-sampling procedure limited to states in the active pool C. For each time step and each $q \in \{1 \cdots P\}$ we compute

$$p(s_t = C_{tq}|y_{1:t}) \propto \sum_{p=1}^{P} \frac{p(y_t|s_t = C_{tq}) p(s_t = C_{tq}|s_{t-1} = C_{t-1,p}) p(s_{t-1} = C_{t-1,p}|y_{1:t-1})}{\beta_{C_{tq}}}. \tag{3.14}$$

We then follow the same strategy as for the finite HMM and run a backward-sampling step to sample the state sequence $s_{1:T}$: we first sample S_T from the density $p(s_T|y_{1:T}, C, \pi, \theta, \beta)$

[3] In what follows, we leave out some conditioning variables for the sake of clarity.

conditioned on all observed variables. Then we iteratively sample $p(s_t|y_{1:T}, s_{t+1}) \propto p(s_t|y_{1:t})p(s_t|s_{t+1})$.

It is clear that this procedure explicitly controls the computational cost of inference since the forward-filtering backward-sampling has a computational cost of $\mathcal{O}(TP^2)$. Next we show that this sampling scheme is correct by proving that detailed balance holds.

Detailed balance demands that for a transition kernel $Q(s'_{1:T}|s_{1:T})$ and equilibrium distribution $p(s_{1:t}|y_{1:T})$ the following condition holds

$$p(s_{1:t}|y_{1:T})Q(s'_{1:T}|s_{1:T}) = p(s'_{1:t}|y_{1:T})Q(s_{1:T}|s'_{1:T}) \tag{3.15}$$

for all state sequences $s_{1:t}, s'_{1:t}$. For any active pool C, the transition kernel is the product of choosing the active pool given the state sequence and then choosing the new state sequence given the active pool: $Q(s'_{1:T}|s_{1:T}) = p(s'_{1:T}|C)p(C|s_{1:T})$. Since we chose the active pool by sampling i.i.d. states from the distribution β, we write

$$p(C|s_{1:T}) = \prod_{t=1}^{T} \prod_{p \neq s_t} \beta_{C_{tp}}. \tag{3.16}$$

Using Bayes rule to rewrite $p(s_{1:T}|y_{1:T})$, combining it with equation (3.13) and equation (3.16), we can expand out the left hand side of equation (3.15) as

$$\left(\frac{\prod_{t=1}^{T} p(y_t|s_t) \prod_{t=1}^{T} p(s_t|s_{t-1})}{p(y_{1:T})} \right) \left(\prod_{t=1}^{T} \prod_{p \neq s_t} \beta_{C_{tp}} \right) \left(\frac{\prod_{t=1}^{T} p(y_t|s'_t) \prod_{t=1}^{T} p(s'_t|s'_{t-1})}{p(y_{1:T}) \prod_{t=1}^{T} \beta_{s'_t}} \right).$$

We do the same expansion for the right hand side of equation (3.15) and find

$$\left(\frac{\prod_{t=1}^{T} p(y_t|s_t) \prod_{t=1}^{T} p(s_t|s_{t-1})}{p(y_{1:T}) \prod_{t=1}^{T} \beta_{s_t}} \right) \left(\prod_{t=1}^{T} \prod_{p \neq s'_t} \beta_{C_{tp}} \right) \left(\frac{\prod_{t=1}^{T} p(y_t|s'_t) \prod_{t=1}^{T} p(s'_t|s'_{t-1})}{p(y_{1:T})} \right).$$

By moving the factors $\prod_{t=1}^{T} \beta_{s'_t}$ and $\prod_{t=1}^{T} \beta_{s_t}$ from the denominator on one side to the numerator of the other side, we find that both sides are equal to

$$\left(\frac{\prod_{t=1}^{T} p(y_t|s_t) \prod_{t=1}^{T} p(s_t|s_{t-1})}{p(y_{1:T})} \right) \left(\prod_{t=1}^{T} \prod_{p=1}^{T} \beta_{C_{tp}} \right) \left(\frac{\prod_{t=1}^{T} p(y_t|s'_t) \prod_{t=1}^{T} p(s'_t|s'_{t-1})}{p(y_{1:T})} \right).$$

This concludes the proof that detailed balance holds.

Discussion

We run a small experiment to compare the behavior of the embedded HMM sampler with that of the beam sampler. In our experiment we generate a sequence of 1500 datapoints from an HMM with a cyclic transition matrix: $\pi_{i,j} = 0.5$ if $i = j$ or $i + 1 = j$ and 0

otherwise. The observations consist of an alphabet of 8 symbols; each HMM state emits 3 of the 8 symbols uniformly at random, one of the 8 symbols is unique to that state, the two others are overlapping with two other states.

For every pool size $P \in \{2, \cdots, 10\}$ we run the embedded HMM sampler three times and compare it to three runs of the beam sampler. We set the hyperpriors of the iHMM at $\alpha = 1$ and $\gamma = 1$ and run inference for 500 iterations. We compare the samplers based on how fast they mix and wall clock time. The first 9 plots in figure 3.13 illustrate one statistic of the samplers: the number of used states at each iteration. Recall that the true number of states that generated the data is 4. The plots clearly show that the embedded HMM sampler with a small number of states mixes very slowly. This is not very surprising, e.g. for $P = 2$ the sampler will only consider two possible states at each iteration. When $P = 6$ we can see that the embedded HMM mixes as fast as the beam sampler. The bottom right plot shows the wall clock time in seconds of the embedded HMM sampler for different values of P. When P is small, the embedded HMM sampler is faster than the beam sampler; again this is not surprising as we've specifically restricted the number of states the sampler considers.

Finally, we point out that other schemes for sampling the active pool C are possible. In our experience, sampling from the base distribution β often finds a good active pool while minimising the computational effort in choosing it.

3.2.4 Hyper parameter Learning

Section 3.1.1 contained some intuitions about the role of hyper parameters α and γ. To recap, γ influences how frequently the hierarchical Polya urn will generate new states. Hyper parameter α influences how often we will visit the oracle in the hierarchical Polya urn, this has two effects. First it indirectly also influences how often new states are generated. Secondly, it also influences how often the iHMM takes existing transitions and how often it chooses a new transition: a small α will generally use previously taken transitions leading to a sparse transition matrix. In other words, α controls the sparsity of the iHMM's transition matrix.

An important question when applying the iHMM in practice is whether to set or learn the hyper parameters of the iHMM. In chapter 4 we will provide some insight into the question of whether to fix or sample hyper parameters of the iHMM. In case we wish to learn the hyper parameters of the iHMM, a reasonable prior for both α and γ is the Gamma distribution. One can show that a simple auxiliary variable Gibbs sampler can sample from the posterior distribution very efficiently. We refer to (Teh et al., 2006a, appendix a) for full details of this sampler which readily applies to the infinite Hidden Markov Model.

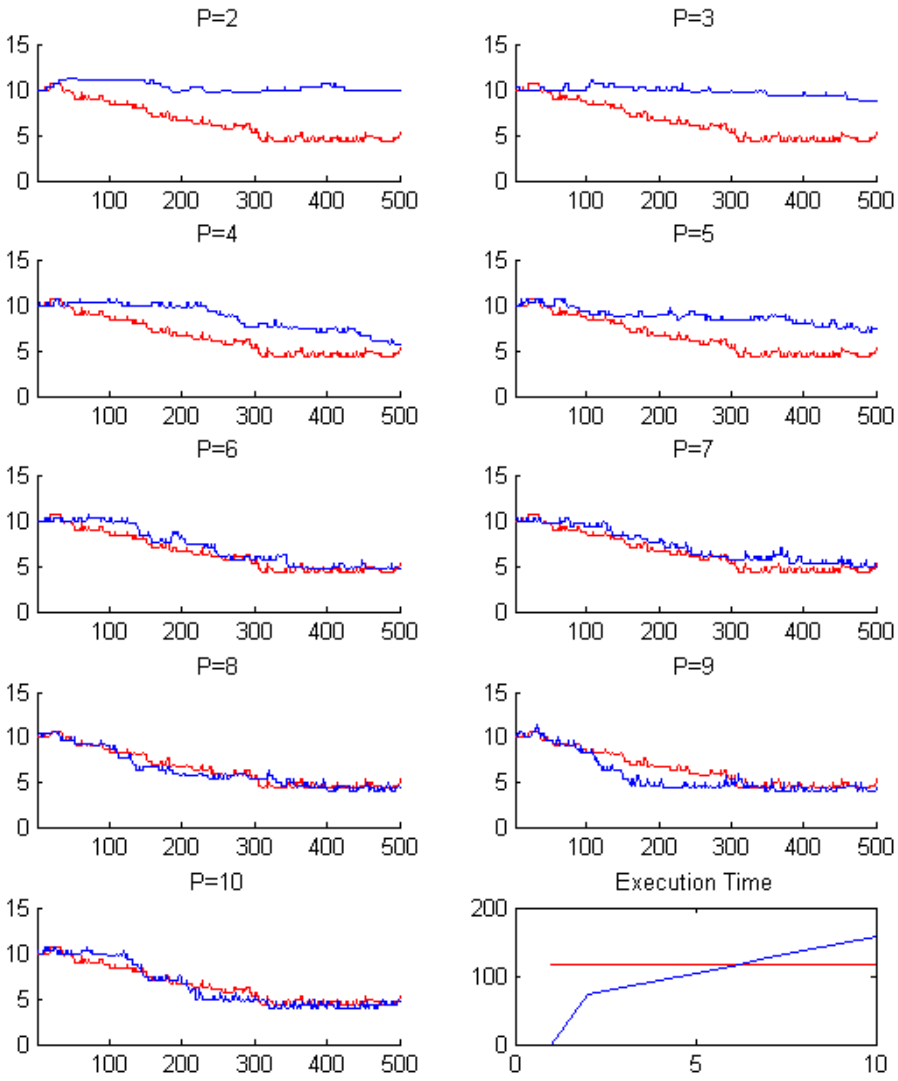


Figure 3.13: This plot compares the embedded HMM sampler (blue) with that of the beam sampler (red). The first 9 plots compare the mixing time, for different pool sizes P of the embedded HMM. The Y-axis of the plots represent the number of used states in the latent Markov chain at every iterations of the samplers; the number of states for the chain that generated the data is 4. The last plot shows the wall clock time in seconds for 500 iterations for different pool sizes P.

3.3 Alternatives to the iHMM

The iHMM is an attempt to solve the model selection problem for HMM's. It is a relatively new method for solving this problem and a number of existing statistical techniques

already exist. In this chapter we give a short overview of the most common methods and how they relate to the iHMM.

3.3.1 Model Selection

There is a large body of work on model selection techniques for the HMM (Cappé et al., 2005). The simplest technique uses maximum likelihood estimation to train HMM with a varying number of hidden states and then use some regularisation method to penalise models that overfit. Both AIC and BIC are common regularisation techniques used in this space.

A different family of methods uses a fully Bayesian analysis of the HMM and tries to estimate the marginal likelihood to perform model selection. One problem with this group of methods is that computing the marginal likelihood of the Bayesian HMM is intractable. We can distinguish a few practical approaches to approximate the marginal likelihood for the HMM. The first family of approximations are based on MCMC: examples are Annealed Importance Sampling (Neal, 2001) and Bridge Sampling (Fruhwirth-Schnatter, 2004). MCMC methods for estimating marginal likelihoods are computationally expensive and often don't mix well.

A second and very elegant approximation to the exact marginal likelihood is the approach developed by Stolcke and Omohundro (1993). Note that in the graphical model in figure 1.7, if the hidden states $s_{1:T}$ were observed, the parameters π and θ become independent and assuming that the prior and likelihood are conjugate, we can compute the marginal likelihood analytically. Stolcke and Omohundro (1993) propose to choose a good state sequence and integrating out the other parameters to compute an approximation to the marginal likelihood. They devise a state-merging algorithm based on this idea.

A third technique to approximate the marginal likelihood is based on *variational Bayesian* (VB) inference. VB computes a lower bound on the marginal likelihood; MacKay (1997) and Beal (2003) describe VB inference algorithms that bound the marginal likelihood of an HMM. VB generalises EM as it doesn't use a point estimate of the parameters π, θ but rather an approximate posterior of these parameters. Moreover, VB also generalises the idea in Stolcke and Omohundro (1993) as it doesn't use a point estimate of the state sequence $s_{1:T}$ but rather a full distribution on this quantity.

To illustrate the last technique we ran an experiment on a character prediction task. We take the first 1000 characters from *Alice's Adventures in Wonderland* as a training set and evaluate different models on the subsequent 4000 characters. On one hand we train the iHMM using the beam sampler using a burn-in of 1000 iterations and collecting 50 samples over the next 10,000 iterations. We chose the hyper priors for the iHMM to be $\gamma = 4$ and $\alpha = 1$. On the other hand we train a finite HMM using variational Bayes on a

model with $K = 1$ to 50 hidden states. For the HMM transition matrix prior we choose a symmetric Dirichlet with $4/K$ pseudo-counts for each state and a symmetric Dirichlet with 0.3 pseudo-count for the emission distribution. The predictive performances for the HMM and iHMM were averaged over 50 and 20 independent runs respectively.

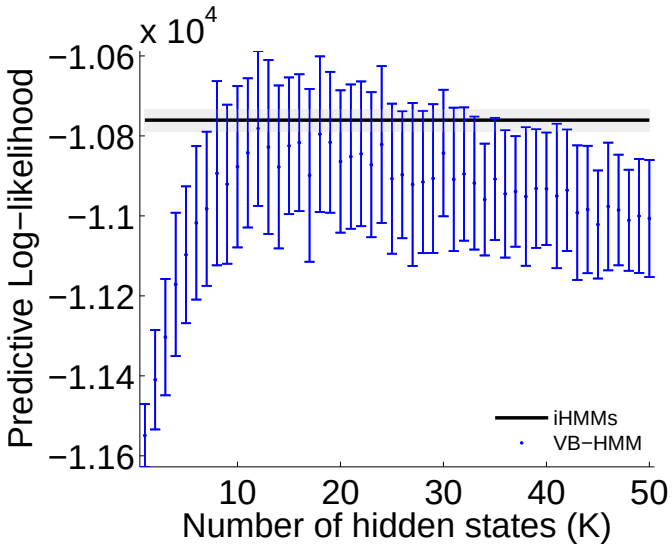


Figure 3.14: Comparing model selection with the iHMM on predicting characters for *Alice's Adventures in Wonderland*.

Figure 3.14 illustrates the estimated predictive log-likelihoods for the HMM and the iHMM. We find that the iHMM has slightly superior predictive power when compared to the HMM's, even when we select the best number of hidden states (around $K = 16$). The iHMM trained with the collapsed Gibbs sampler as well as the beam sampler both converged to a posterior distribution over hidden state sequences with around 16 states, showing that in terms of statistical performance, nonparametric Bayesian techniques are an effective alternative to Bayesian model selection. The advantage of using an iHMM is that we only trained one model.

3.3.2 Reversible Jump versus Nonparametric HMM

The model selection methods described above assume that one model with a fixed capacity is the right model. Reversible jump MCMC methods are a family of techniques which allow switching between models of varying capacity and hence take the uncertainty in model capacity seriously. These methods were pioneered in Green (1995) and extended to the context of HMM's by Robert et al. (2000).

Ryden (2008) performs a detailed analysis of the reversible jump MCMC algorithm on the HMM. The data $\{y_t\}$ consist of the 1700 daily log-returns from the S&P 500 stock index during the 1950's. The stochastic volatility model that is trained in Ryden (2008) is

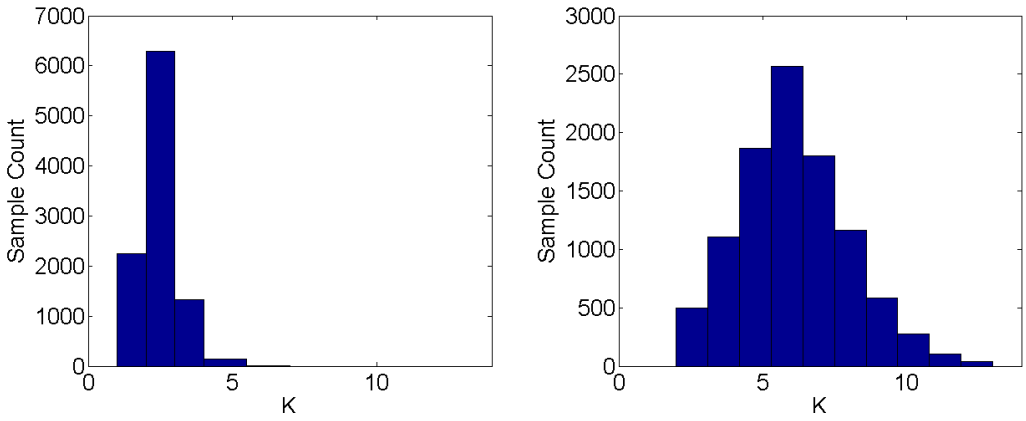


Figure 3.15: Comparing cluster occupancy numbers for the reversible jump MCMC (left) with the iHMM (right) for the last 10,000 iterations of the sampler.

an HMM with normally distributed likelihoods having zero mean and variance σ_i for state i. The prior for the standard deviations is $\sigma_i \sim \mathsf{Uniform}(0, \alpha)$ whith $\alpha \sim \mathsf{Exp}(5 \max_t y_t)$. Ryden trains models with $k \in \{1, \cdots, 8\}$ hidden states over which he puts a uniform prior. More details about the reversible jump algorithm and its implementation can be found in Ryden (2008).

We compare Ryden's Matlab implementation to the iHMM with exactly the same hyper parameters on σ_i. We chose vague hyper priors for the iHMM: $\alpha = 1$, $\gamma = 1$. We ran both samplers for 100,000 iterations and plot the histogram of cluster occupancy counts for the last 10,000 iterations in figure 3.15. Observe that the iHMM settles for a larger number of hidden states than the solution found by the reversible jump MCMC method. A closer inspection for the samples offers the following explanation. For each sample, almost all of the latent variables of the iHMM are in fact in one of three states; only very few latent variables are in other states. The reason is that the iHMM is continually exploring new states; this means that in every iteration a few latent variables will be assigned to a separate cluster. Most of the time, these new clusters disappear after one or two iterations, but then new clusters emerge for different latent variables. In that sense, our results are in close agreement with the findings in Ryden (2008). More interestingly though, we can compute the execution time of the iHMM versus the reversible jump MCMC method. We find that on the same machine the reversible jump MCMC method uses four times as much computational time than the iHMM.

3.3.3 Conclusion

There are a few other alternatives to the iHMM for model selection that we have not discussed here. However, we believe that compared to the most popular classical model selection approaches, the iHMM is a valid alternative which gives qualitatively very sim-

ilar results at a much lower computational cost.

3.4 Extensions

We conclude this chapter with an overview of some extensions to the basic iHMM model.

3.4.1 The Input Output iHMM

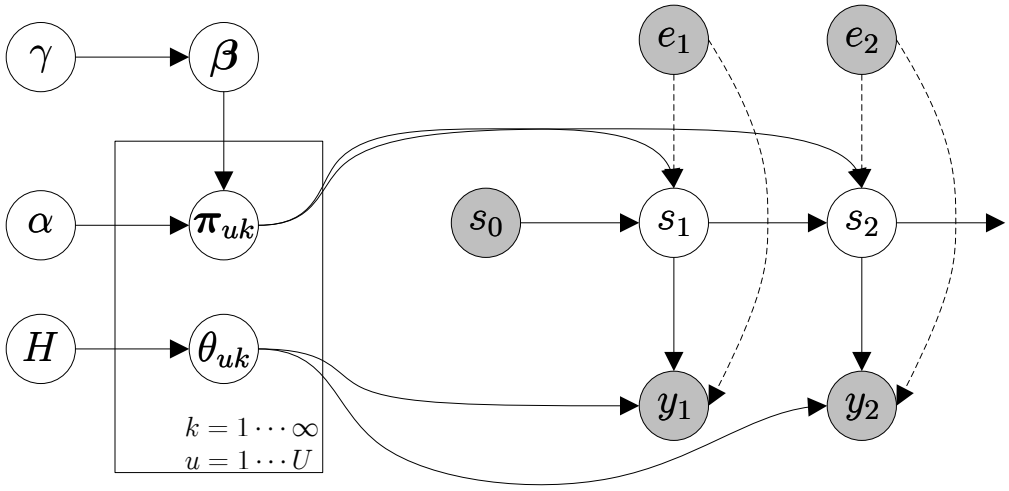


Figure 3.16: The graphical model for the IO-iHMM. The dotted lines denote the fact that we can choose to make either the Markov chain or the observations (or both) dependent on a set of inputs.

The iHMM, like the HMM assumes that the Markov chain evolves completely autonomously and the observations are conditionally independent given the Markov chain. In many real scenarios, the Markov chain can be affected by external factors: e.g. a robot is driving around in an interior environment while taking pictures. The sequence of pictures could be modelled by an iHMM where the latent chain represents the robot's location (e.g. a room index). If our robot doesn't perform a random walk through the environment but uses a particular policy, we can integrate the robot's actions as inputs to the iHMM. We call this model the Input-Output iHMM (IO-iHMM) in analogy to its finite counterpart the IO-HMM by Bengio and Frasconi (1995). Figure 3.16 illustrates the graphical model for the IO-iHMM. A different way to think about the IO-iHMM is as a conditional sequence model: it predicts the sequence of observations $y_{1:T}$ from inputs $e_{1:T}$ through a hidden representation $s_{1:T}$. This model can be useful for structured sequence prediction problems.

We can think of the IO-iHMM where the transition probabilities depend on the inputs e_t as an iHMM with a 3-dimensional transition matrix: for each input symbol e and each

previous state s, the probabilities for moving to the next state are $p(s_t|s_{t-1}, e_t) = \pi_{e_t,s_{t-1},s_t}$. The vectors $\pi_{e,s,\cdot}$ can be constructed similarly to the rows of the transition matrix in the iHMM, by drawing $\pi_{e,s,\cdot} \overset{iid}{\sim} \mathsf{Stick}(\alpha\beta)$. One effect of introducing a dependency on input variables is that the number of parameters increases. If we train a finite sequence of length T with the iHMM and find an effective number of states equal to K then we have to learn $\mathcal{O}(K^2)$ transition matrix parameters. If we train the same sequence with the IO-iHMM using an input sequence with E possible input symbols and find K effective states, then we have to learn $\mathcal{O}(EK^2)$ transition matrix parameters. Similarly, if we introduce a dependency of the observation model on the input sequence, $y_t \sim F(\theta_{s_t,e_t})$, we augment the number of parameters of the observation model with a factor of E. This suggests that either: a) more data will be needed to learn a model with the same capacity as the iHMM; b) with the same amount of data a model with smaller capacity will be learnt or c) the model complexity must be reduced by coupling the parameters.

3.4.2 The iHMM with Pitman-Yor Base Distribution

From the Polya urn scheme, we know that the base process (the oracle urn) behaves like an average distribution over the states. In other words, the base process influences the expected number of states visited for a given sequence length. The left plot in figure 3.17 plots the frequency of colours in a Polya urn for a Dirichlet process against the colour's rank. From this plot, we can see that the Dirichlet process is quite specific about the distribution implied in the Polya urn: because the right tail of the plot drops off sharply, we know that the number of colours that appear only once or twice is very small. A two parameter generalisation of the Dirichlet process, known as the Pitman-Yor process (Pitman (2006)) introduces more flexibility in this behaviour. The right plot in figure 3.17 shows that the frequency-rank for draws from a Pitman-Yor process *can* be more specific about the tails. More in particular, the Pitman-Yor process fits a Power-law distribution; Goldwater et al. (2006) and Teh (2006) showed how in the context of language modelling the Pitman-Yor distribution encodes more realistic priors. Analogous to the Dirichlet process, the Pitman-Yor process has a stick breaking constructing: the sticks for the Pitman-Yor distribution with parameters d and α can be constructed by drawing $\hat{\beta}_i \overset{iid}{\sim} \mathsf{Beta}(1-d, \alpha+id)$ and $\beta_i = \hat{\beta}_i \prod_{l=1}^{i-1}(1-\hat{\beta}_l)$. Note that the Pitman-Yor process with $d = 0$ is exactly the Dirichlet process.

In the context of the iHMM, it is fairly straightforward to replace the Dirichlet process base distribution by a Pitman-Yor distribution allowing the iHMM to use more states. The collapsed Gibbs sampler works with only minor adjustments, however, it is critical that the beam sampler in this context be implemented with great care. As we mentioned in the discussion of the beam sampler, we need to consider all transitions ij where $\pi_{ij} > u_t$. Unfortunately, the quantity π_{ij} decreases much faster for the Pitman-Yor process

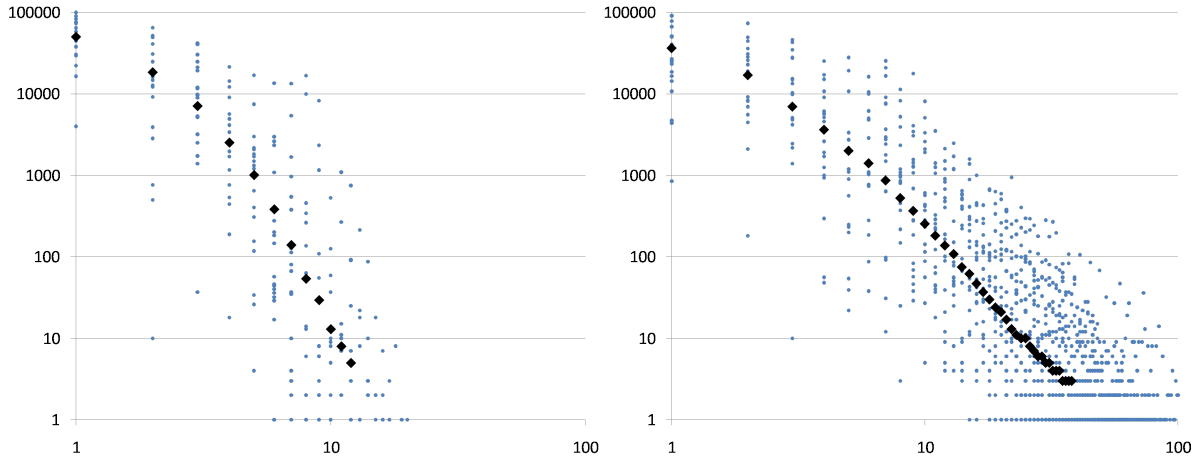


Figure 3.17: In this plot we show (on a log-log scale) the frequency versus rank of colours in the Polya urn for both the Dirichlet process (left) and the Pitman-Yor process (right). We see that the data from the Pitman-Yor process can be fitted with a straight line implying that it follows a power-law behavior; this is not the case for the Dirichlet process.

stick breaking construction than the Dirichlet process stick breaking construction. The advantage is that it allows for many small transition probabilities, unfortunately it also means that the quantity $\sum_l \pi_{il}$ decreases very slowly. Since the beam sampler must consider all $\pi_{ij} > u_t$, it will need to expand the transition matrix to K states so that $\sum_{l>K} \pi_{il} < u_t$ and it is sure it considered all possible states. This means when running the forward-filtering backward-sampling algorithm that we potentially end up with a very large transition matrix. Luckily most of the entries in π will be quite small (otherwise the remaining mass $\sum_{j>K} \pi_{ij}$ can't decrease slowly). Hence, while expanding the transition matrix we can keep a sorted list of matrix entries and when we run the forward-filtering step, at each time step we walk down this list until u_t becomes larger than the elements in the list. The embedded HMM from section 3.2.3 can easily be adapted to perform exact inference for the iHMM with Pitman-Yor base distribution as well. The major difference is that the active pool C is now sampled from the stick β which now follows a Pitman-Yor distribution.

3.4.3 The Sticky and Block Diagonal iHMM

In many applications, it is useful to explicitly model the time scale of transitions between states. For an HMM or iHMM, the weight on the diagonal of the transition matrix controls the frequency of state transitions. The probability that we stay in state i for g time steps is a geometric distribution $p(g) = \pi_{ii}^{g-1}(1 - \pi_{ii})$.

As we noted in the introduction to the iHMM, the original hierarchical Polya urn description in Beal et al. (2002) introduced a self transition hyper parameter in the

iHMM adding prior probability mass to the diagonal of the transition matrix π. Fox et al. (2009b) further elaborated this idea in the hierarchical Dirichlet process representation of the iHMM. They also developed a dynamic programming based inference algorithm for a truncated version of the model. They coined it the *sticky iHMM* (or sticky HDP-HMM). The sticky iHMM is particularly appropriate for segmentation problems where the number of segments is not known a priori. Fox et al. (2009b) show impressive results on a speaker diarization task.

Formally, the sticky iHMM still draws the base measure $\beta \sim \mathsf{Stick}$ but then draws the rows of the transition matrix from the following distribution

$$\pi_{i\cdot} \overset{iid}{\sim} \mathsf{Dirichlet}(\frac{\alpha}{\alpha+\kappa}\beta_1, \frac{\alpha}{\alpha+\kappa}\beta_2, \cdots, \frac{\alpha\beta_i+\kappa}{\alpha+\kappa}, \cdots). \tag{3.17}$$

The effect of this change is that for all states the diagonal entry carries more weight. The parameter κ controls the switching rate or length scale of the process.

A more general model which allows grouping of states into a block diagonal structure is given by the *block-diagonal infinite HMM* (BD-iHMM) introduce in Stepleton et al. (2009). When the blocks are of size one, this is a sticky iHMM, but larger blocks allow unsupervised clustering of states. The block-diagonal iHMM is used for unsupervised learning of view-based object models from video data, where each block or cluster of states corresponds to an object and the model assumptions capture the intuition that temporally contiguous video frames are more likely to correspond to different views of the same object than to different objects.

Another approach to introduce more control over the self-transition probability is to have an explicitly duration model for the time spent in a particular state. This type of model is known as a hidden semi-Markov model. Inference in these models is more costly than in HMM's and nonparametric versions have not yet been explored.

3.4.4 The Auto-Regressive iHMM & Switching Linear Dynamical Systems

Many time series are modelled using auto-regressive (AR) or linear dynamic systems (LDS). These models assume that the dynamics of the process are stationary and linear. More powerful nonlinear generalisations can be obtained by switching between a fixed set of stationary dynamics. A model where the switching is driven by an HMM is often called a switching linear dynamical systems [4]; Ghahramani and Hinton (2000) review a number of variants of the SSSM all of whom share the property that there are a fixed finite number of dynamics.

[4] Also known as a Markov-switching model, Markov jump system or Switching State Space Models

Fox et al. (2009a) extended the SSSM so that an arbitrary large number of dynamics can be learnt from the data by replacing the finite Markov chain underlying the SSSM with an iHMM. They explored two prototypical variants of the SSSM. The first model assumes the observations follow auto-regressive dynamics; this model is referred to as the AR-iHMM. The second model assumes that only part of the continuous variables are observed and the unobserved variables follow linear dynamics; this model is referred to as the infinite switching linear dynamical system. The inference in Fox et al. (2009a) is implemented on a finite truncation of the nonparametric model. In this incarnation, the model is similar to the Bayesian switching linear Gaussian model from Chiappa (2008) except that the nonparametric model introduces an additional "prior on the prior" for the rows of the transition matrix.

Figure 3.18 illustrates the graphical model for the LDS SSSM variant. Inference in both models can be done efficiently using the beam sampler. For the AR model, a collapsed Gibbs sampler is possible while for the LDS we are only aware of algorithms that explicitly represent the hidden state of the LDS.

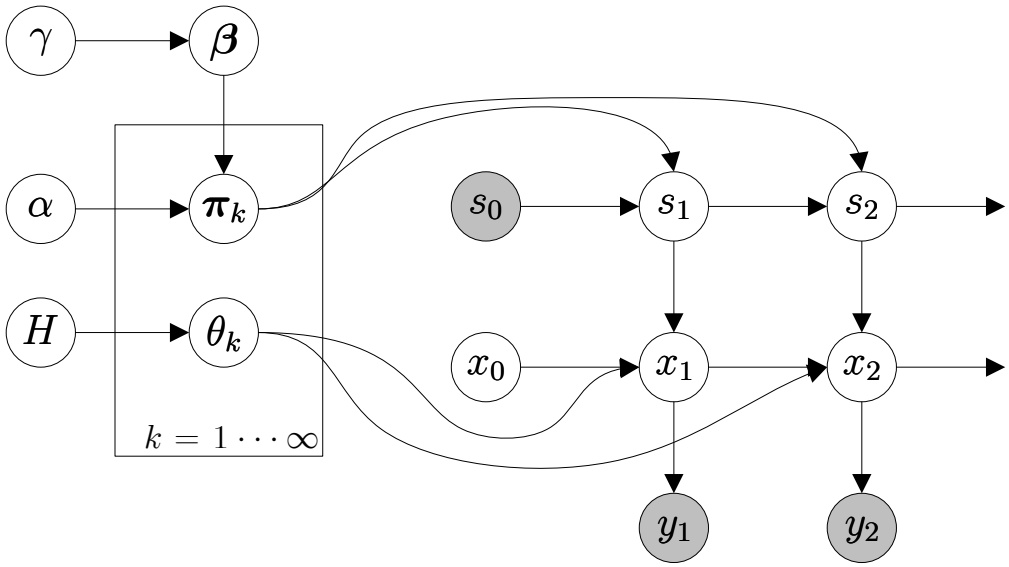


Figure 3.18: The graphical model for a switching state space model with a potential infinitely large state space.

3.5 Applications & Further Reading

The iHMM has been the foundation of a number of application in bio-informatics. Beal and Krishnamurthy (2006) describes how the iHMM can be used for clustering gene expression time course data. Their results show a clear advantage of the iHMM over model selection using the finite HMM on various different clustering metrics. Sohn and

Xing (2007) introduces a model that is equivalent to the iHMM called the *Hidden Markov Dirichlet Process*. They use the iHMM to model genetic recombinations where the number of founders can be arbitrary large. Their results also show a clear improvement over model selection using the HMM.

Change point and outlier detection in time series can often be modelled using Markov models. Van Gael et al. (2008a) perform segmentation of a one dimensional time series with outliers using the iHMM. Fox et al. (2008b) show impressive results using the block diagonal iHMM for a speaker diarization task. Paisley and Carin (2008) solve a music segmentation task using the iHMM; their additional contribution generalises the iHMM to use arbitrary stick breaking constructions. Pruteanu-Malinici and Carin (2008), Zhang et al. (2009) show how to model abnormal events in video sequences using the iHMM. Stepleton et al. (2009) performs gesture classification in video streams using the block diagonal iHMM.

Another interesting extension of the iHMM is the multi-task learning iHMM in Dunson (2007). In this model an additional nested Dirichlet process is used to couple different iHMM's for multi-task learning. More specifically in this model, dubbed the nDP-iHMM, there are J learning tasks, each modelled with its own iHMM. The base distribution for each of the J iHMM's is a draw from a nested Dirichlet process. In other words, the iHMM base distributions are shared between tasks. Learning is done using a blocked Gibbs sampler.

Doshi-Velez (2009) introduce a model for reinforcement learning based on the iHMM called the *infinite Partially Observable Markov Decision Process* or iPOMDP.

We conclude this section by pointing to chapter 4 for a more in-depth look at an application of the iHMM in natural language processing.

3.6 Discussion

We reviewed both the hierarchical Polya urn construction and the hierarchical Dirichlet process construction for the iHMM and showed a proof that both formalisations are equivalent. Building on these different representations of the iHMM, we described a collapsed Gibbs sampler and two dynamic programming based samplers. Both inference methods are only marginally harder to implement than their HMM based counterparts. We believe this makes the iHMM an attractive solution to the problems one faces when learning the number of states for an HMM.

As we have described, there are a number of interesting ways of extending iHMMs. We can allow inputs and outputs, sticky self transitions, block structured models, Pitman-Yor base distributions, auto regressive and switching linear structure. Given the important role of hidden Markov models in time series and sequence modelling, and the flexibility

of nonparametric approaches, there is great potential for many future applications and extensions of the infinite hidden Markov model.

Chapter 4

Unsupervised Part-of-Speech Tagging with Nonparametric Models

Many Natural Language Processing (NLP) tasks are commonly tackled using supervised learning approaches. These learning methods rely on the availability of labelled data sets which are usually produced by expensive manual annotation. For some tasks, we have the choice to use unsupervised learning approaches. While they do not necessarily achieve the same level of performance, they are appealing as unlabelled data is usually abundant. In particular, for the purpose of exploring new domains and languages, obtaining labelled material can be prohibitively expensive and unsupervised learning methods are a very attractive choice.

In this chapter we will look at a problem called Part-of-speech (PoS) tagging. PoS-tagging is a task in natural language processing with the goal of labelling words in a sentence with their appropriate syntactic class. E.g.

The	man	sat	on	the	moon	.
Determiner	Noun	Verb	Preposition	Determiner	Noun	.

PoS tagging is a standard component in many linguistic processing pipelines, so any improvement on its performance is likely to impact a wide range of tasks. Although there exist various approaches to PoS tagging, we will look more closely at PoS taggers based on HMM's. The history of HMM based taggers goes as far back as Church (1988) but it was the work of Kupiec (1992), Merialdo (1994), Weischedel et al. (1993) which introduced supervised training algorithms. The recent work of Johnson (2007), Goldwater and Griffiths (2007), Gao and Johnson (2008) explored the task of unsupervised part-of-speech tagging (PoS) using Hidden Markov Models.

A key issue that sets supervised and unsupervised learning apart is that a completely unsupervised learning method will discover the statistics of a data set *according to a particular model choice* but these statistics might not correspond exactly to our linguistic

intuition about PoS tags. Johnson (2007) and Gao and Johnson (2008) assume that words are generated by a hidden Markov model and find that the resulting states strongly correlate with POS tags. Nonetheless, *identifying* the HMM states with appropriate POS tags is hard. Because many evaluation methods often require POS tags (rather than HMM states) this identification problem makes unsupervised systems difficult to evaluate.

Moreover, not only is it hard to map HMM states to PoS tags, choosing the number of states of the HMM is a non-trivial problem. Since the HMM finds statistical properties of the data, it might find that two states map to the same PoS tag or two PoS tags might map to the same HMM state: e.g. nouns and proper nouns have similar statistical properties and hence might all cluster in the same HMM state. This means that in addition to finding a good mapping from HMM states to PoS tags, unsupervised PoS tagging methods that use HMM's need to take the model selection problem seriously. In previous work on unsupervised PoS tagging Johnson (2007) reports results for different numbers of hidden states but it is unclear how to make this choice a priori, while Goldwater and Griffiths (2007) leave this question as future work. In this chapter we will take an in-depth look at how nonparametric Bayesian hidden Markov models can be used to handle the model selection problem.

The chapter is structured as follows: first we introduce how HMM's can be used for PoS tagging and describe the previous work on using unsupervised learning. Next, we take an in depth tour on how to model the PoS-tagging task using the iHMM. Then we show how we can use the iHMM for unsupervised PoS tagging and describe some empirical results. This chapter is based on Van Gael et al. (2009).

4.1 Unsupervised PoS Tagging using the HMM

The basic idea for using first order HMM's for PoS-tagging is to assume the data is generated from the following generative process. For a sentence of length T, a sequence of tags $s_{1:T}$ is generated using a Markov chain with transition matrix π. Next, given the tags $s_{1:T}$ a sequence of words is generated using a set of emission distributions θ as $w_t \sim \theta_{s_t}$. This generative model can easily be generalised to higher order Markov chains or multiple state sequences. For simplicity, in this section we will assume a first order Markov and single state sequence.

One way to do unsupervised training of the HMM is to compute the maximum-likelihood estimator of the following model

$$\hat{\pi}, \hat{\theta} = \arg\max_{\pi,\theta} p(w_{1:T}|\pi, \theta) = \arg\max_{\pi,\theta} \sum_{s_{1:T}} p(w_{1:T}, s_{1:T}|\pi, \theta). \tag{4.1}$$

When a prior distribution on the parameters is available, a maximum-a-posteriori solution

can be found by optimising

$$\hat{\pi}, \hat{\theta} = \arg\max_{\pi,\theta} p(w_{1:T}|\pi,\theta)p(\pi,\theta) = \arg\max_{\pi,\theta} \sum_{s_{1:T}} p(w_{1:T}, s_{1:T}|\pi,\theta)p(\pi,\theta). \tag{4.2}$$

Both the maximum-likelihood and maximum-a-posteriori solutions can be computed efficiently using the EM algorithm. A careful analysis of the EM algorithm in Johnson (2007) shows that these point estimates suffer from overfitting. One particular form of overfitting is illustrated in figure 4.1. The figure shows various HMM based models that

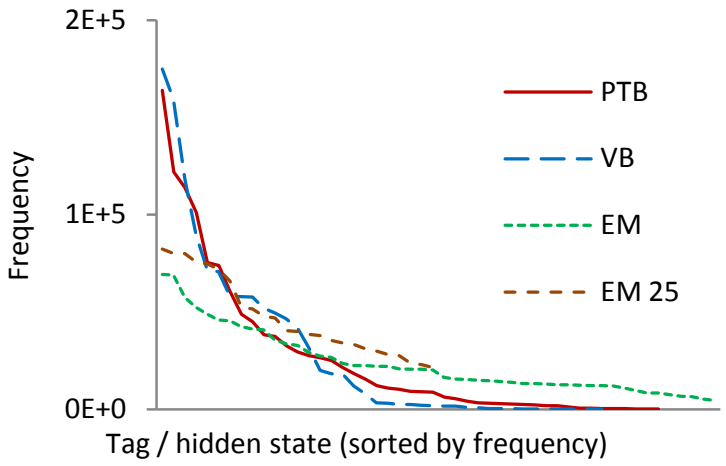


Figure 4.1: Comparison of unsupervised PoS taggers (with permission Johnson (2007)). The x-axis represents the tags in decreasing order of usage. The y-axis shows the actual number of words labeled with a particular tag. The red line denotes the empirical distribution from the Penn Treebank; the blue line represents the distribution from training an HMM with 50 states using variational Bayesian inference; the dotted green line represents the distribution of tags when training an HMM with 50 states using EM; the dotted red line represents the distributioin of tags when training an HMM with 25 states using EM.

were trained on the Penn Treebank corpus. In this labelled corpus, we can compute the average number of words labelled with each tag. As the red line in figure 4.1 illustrates, the empirical tag frequencies show a power-law type of behaviour: a few states label many words and many states label only a few words. When we compare the average number of words labelled in the maximum-likelihood solution we find a much flatter curve: all states label a lot of words. The underlying statistical problem is that the maximum-likelihood solution uses more states to find a better fit. Empirically it was found that a consequence of this behaviour is that the resulting HMM estimate has a dense transition matrix and dense output distributions. This is in contrast to what we know is true of natural language: e.g. we hardly ever see transitions from determiners to verbs.

In section 1.1 we argued that a Bayesian treatment of a probabilistic model can help solve this problem. By introducing an appropriate prior distribution over the transition

and emission matrices, we can encode our beliefs about natural language to find better solutions to the unsupervised PoS-tagging problem. In the HMM for PoS-tagging scenario, we would integrate over the parameters π, θ to find the posterior assignment of tag assignments

$$p(s_{1:T}|w_{1:T}) \propto \int p(w_{1:T}|s_{1:T}, \theta)p(s_{1:T}|\pi)p(\pi, \theta)d\pi d\theta, \tag{4.3}$$

or we would compute the posterior distribution over the parameters integrating out the tag assignments

$$p(\pi, \theta|w_{1:T}) = \sum_{s_{1:T}} p(w_{1:T}|s_{1:T}, \theta)p(s_{1:T}|\pi)p(\pi, \theta)d\pi d\theta). \tag{4.4}$$

We can use a Dirichlet distribution as the prior on π and θ and leverage the properties described in appendix A to encode our prior belief that the transition and emission distribution are sparse. This is exactly what was done in Johnson (2007), Goldwater and Griffiths (2007), Gao and Johnson (2008): by choosing a sparsity inducing Dirichlet prior, unsupervised training of the PoS-tagging task finds PoS assignments that are much closer to what we see in natural language. Figure 4.1 illustrates this quite nicely: the blue curve represents the word assignment distribution when training an HMM in a fully Bayesian way using variational Bayes. It is clear that the blue curve matches the empirical state-to-word assignment much better than the EM-based approaches.

We postpone our discussion of our evaluation methodology and experiments to section 4.3. At this point we point to the following question raised in Johnson (2007), Goldwater and Griffiths (2007): how do we choose the number of states for the iHMM?

4.2 Unsupervised PoS Tagging using the iHMM

Our discussion above suggests that the iHMM could be a good candidate for modelling PoS sequences. It essentially behaves in the same way as the HMM but it has the added flexibility to adapt the number of states to the data automatically. In the following subsections we will start from an initial base model and incrementally extend the model to make it more appropriate for tagging natural language.

4.2.1 The Baseline iHMM

Our baseline iHMM for PoS tagging builds directly on the HMM model defined in the previous section. Since we have discrete observations (the words) we use a discrete output distribution. The Dirichlet distribution is the conjugate prior to the discrete distribution and hence we use it as the base distribution H in our iHMM. The generative model for

this iHMM can be described as follows

$$\begin{aligned}
\beta|\gamma &\sim \mathsf{Stick}(\gamma) \\
\pi_{k\cdot}|\alpha,\beta &\overset{iid}{\sim} \mathsf{Stick}(\alpha\beta) \qquad \forall k \in \{1\cdots\infty\}, \\
s_t|s_{t-1},\pi &\sim \pi_{s_{t-1},\cdot} \qquad \text{with} \quad s_0 = 1, \\
\theta_k|\delta &\overset{iid}{\sim} \mathsf{Dirichlet}(\delta) \qquad \forall k \in \{1\cdots\infty\}, \\
w_t|s_t,\theta &\sim \mathsf{Discrete}(\theta_{s_t}).
\end{aligned}$$

4.2.2 The Pitman-Yor iHMM

As we discussed in section 3.4.2, the Dirichlet process defines a very specific, non power-law distribution over the number of states in the iHMM. Particularly in the context of natural language processing, Goldwater and Griffiths (2007) have shown that the Pitman-Yor distribution can more accurately capture power-law like distributions that frequently occur in natural language. Hence, a first extension to our baseline iHMM is to introduce a Pitman-Yor base distribution for β. More specifically, we choose to construct β from a Pitman-Yor stick breaking construction with discount parameter $0 \leq d < 1$ and concentration parameter $\gamma > -d$. The full probabilistic model is now

$$\begin{aligned}
\beta|\gamma,d &\sim \mathsf{PYStick}(d,\gamma) \\
\pi_{k\cdot}|\alpha,\beta &\overset{iid}{\sim} \mathsf{Stick}(\alpha\beta) \qquad \forall k \in \{1\cdots\infty\}, \\
s_t|s_{t-1},\pi &\sim \pi_{s_{t-1},\cdot} \qquad \text{with} \quad s_0 = 1, \\
\theta_k|\delta &\overset{iid}{\sim} \mathsf{Dirichlet}(\delta) \qquad \forall k \in \{1\cdots\infty\}, \\
w_t|s_t,\theta &\sim \mathsf{Discrete}(\theta_{s_t}).
\end{aligned}$$

4.2.3 The PoS-tagging iHMM

During our initial experiments with the Pitman-Yor iHMM we set δ, the hyper parameter of the Dirichlet prior on the multinomial output distributions, to induce sparsity in the output distributions of the iHMM. This solution is not entirely satisfying as it implies that we believe that all states will have an equal sparsity level whereas in natural language we expect tags such as nouns and verbs to have dense output distributions whereas tags such as determiners to have sparse output distributions.

The work of Nemenman et al. (2001) (reviewed in appendix A) addresses this problem: by moving from a Dirichlet to a mixture of Dirichlet distributions. One possible way to achieve this is by putting a prior over the Dirichlet parameter δ, say $\delta \sim \mathsf{Gamma}$. By integrating δ over the prior distribution it essentially follows an infinite mixture of Dirichlet distributions with varying levels of δ.

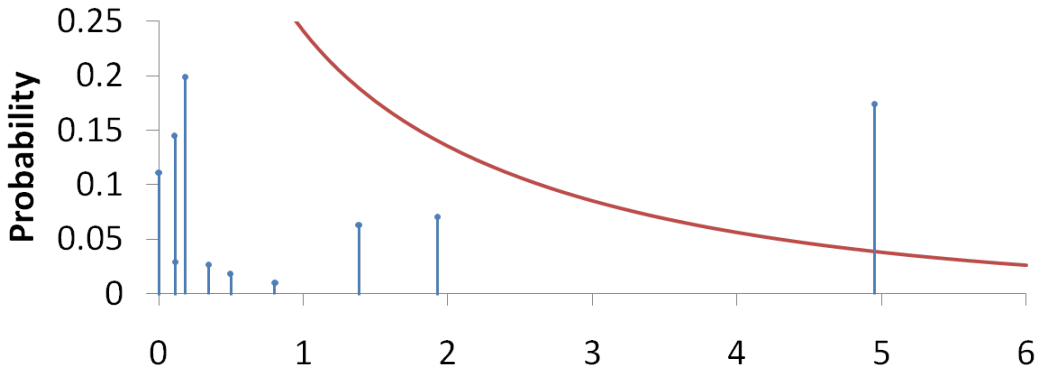


Figure 4.2: This plot shows the stick breaking construction of the Dirichlet Process with Gamma base measure. The blue lines represent the first 10 sticks at their respective locations in parameter space with their height denoting the probability. The red line represents the underlying $\mathsf{Gamma}(0.6, 0.3)$ distribution.

In our PoS-tagging application we take this idea one step further: instead of choosing a Gamma distribution as our mixing distribution, we choose to use a Dirichlet Process prior with Gamma base distribution. Figure 4.2 shows a sample from this prior distribution: there are a potentially infinite number of sparsity levels, but we expect some tag classes to exhibit similar sparsity structure.

With all the improvements listed above, we believe that the following model is a much better prior for unsupervised PoS tagging than the baseline iHMM. The generative model for our final PoS-tagging iHMM is

$$\begin{aligned}
\beta|\gamma, d &\sim \mathsf{PYStick}(d, \gamma) \\
\pi_{k\cdot}|\alpha, \beta &\overset{iid}{\sim} \mathsf{Stick}(\alpha\beta) \qquad \forall k \in \{1 \cdots \infty\}, \\
s_t|s_{t-1}, \pi &\sim \pi_{s_{t-1},\cdot} \qquad \text{with} \quad s_0 = 1, \\
H &\sim \mathsf{DP}(\mu, \mathsf{Gamma}(0.6, 0.3)), \\
\delta_k|H &\sim H \qquad \forall k \in \{1 \cdots \infty\}, \\
\theta_k|\delta &\overset{iid}{\sim} \mathsf{Dirichlet}(\delta_k) \qquad \forall k \in \{1 \cdots \infty\}, \\
y_t|s_t, \theta &\sim \mathsf{Discrete}(\theta_{s_t}).
\end{aligned}$$

The parameterization of the Gamma distribution is arguable; we selected this distribution so it had a mean of 2 and slightly larger standard deviation of roughly 2.5.

4.3 Evaluation

As we alluded to in section 4.1, evaluating unsupervised PoS tagging is difficult mainly due to the fact that the outputs of such systems are not actual PoS tags but HMM

state identifiers. Therefore it is impossible to evaluate performance against a manually annotated gold standard using accuracy. Recent work (Johnson, 2007, Goldwater and Griffiths, 2007, Gao and Johnson, 2008) on this task explored a variety of methodologies to address this issue.

Let C be a random variable denoting the class assignment in the gold standard and let K be a random variable denoting the clustering. Two basic measures to evaluate a clustering against a labelled data set is to compute the homogeneity and completeness. Homogeneity measures the conditional entropy of the class distribution in the gold standard given the clustering: $1 - H(C|K)/H(C)$. Completeness is the dual, it measures the conditional entropy of clustering given the class distribution in the gold standard: $1 - H(K|C)/H(K)$. To illustrate these two quantities consider the confusion matrix in table 4.1 The vertical axis corresponds to states in the clustering whereas the horizontal axis corresponds to ground truth labels. The entries in the matrix represent how many data points were labelled in a particular state and have an actual ground truth labelling corresponding to their column. Homogeneity measures whether the clustering is indicative of a single ground truth cluster. In other words, whether entries in a row all cluster in a single cell. E.g. state 1 always map onto nouns: it has maximal homogeneity. Completeness measures whether a ground truth label is indicative of a clustering state. In other words, whether entries in a column all cluster in a single cell. E.g. nouns map to both state 1 and state 2: completeness is not maximal.

	Noun	Determiner	Verb
State 1	2	0	0
State 2	2	1	0
State 3	0	1	4

Table 4.1: Confusion matrix for a toy clustering.

Although homogeneity and completeness are useful measures for unsupervised PoS tagging, it is common to summarise the quality of a clustering using a single number. The most common approach is to use the Variation of Information (VI) (Meila, 2007) which is defined as $VI = H(C|K) + H(K|C)$. As Johnson (2007) points out, VI is biased towards clusterings with a small number of clusters. A different evaluation measure that uses the same quantities but weighs them differently is the V-measure (Rosenberg and

Hirschberg, 2007), which is defined in Equation 4.5 by setting the parameter β to 1.

$$
\begin{aligned}
h &= 1 - \frac{H(C|K)}{H(C)} \\
c &= 1 - \frac{H(K|C)}{H(K)} \\
V_\beta &= \frac{(1+\beta)hc}{(\beta h) + c}
\end{aligned}
\tag{4.5}
$$

Vlachos et al. (2009) noted that V-measure favours clusterings with a large number of clusters. Both of these biases become crucial in our experiments, since the number of clusters (states of the iHMM) is not fixed in advance. Vlachos et al. proposed a variation of the V-measure, V-beta, that adjusts the balance between homogeneity and completeness using the parameter β in Eq. 4.5. More specifically, setting $\beta = |K|/|C|$ assigns more weight to completeness than to homogeneity in case $|K| > |C|$ since the former is harder to achieve and the latter is easier when the clustering solution has more clusters than the gold standard has classes. The opposite occurs when $|K| < |C|$. In case $|K| = |C|$ the score is the same as the standard V-measure.

It is worth mentioning that, unlike V-measure and V-beta, VI scores are not normalised and therefore they are difficult to interpret. Meila (2007) presented two normalisations, acknowledging the potential disadvantages they have. The first one normalises VI by $2\log(\max(|K|, |C|))$, which is inappropriate when the number of clusters discovered $|K|$ changes between experiments. The second normalisation involves the quantity $\log N$ which is appropriate when comparing different algorithms on the same data set (N is the number of instances). However, this quantity depends exclusively on the size of the data set and hence if the data set is very large it can result in normalised VI scores misleadingly close to 100%. This does not affect rankings, i.e. a better VI score will also be translated into a better normalised VI score. In our experiments, we report results only with the un-normalised VI scores, V-measure and V-beta.

All the evaluation measures mentioned so far evaluate PoS tagging as a clustering task against a manually annotated gold standard. While this is reasonable, it still does not provide means of assessing the performance in a way that would allow comparisons with supervised methods that output actual PoS tags. Even for the normalised measures V-measure and V-beta, it is unclear how their values relate to accuracy levels. Gao and Johnson (2008) partially addressed this issue by mapping states to PoS tags following two different strategies, cross-validation accuracy, and greedy 1-to-1 mapping, which both have shortcomings. We argue that since an unsupervised PoS tagger is trained without taking any gold standard into account, it is not appropriate to evaluate against a particular gold standard, or at least this should not be the sole criterion. The fact that different authors use different versions of the same gold standard to evaluate similar

experiments (e.g. Goldwater and Griffiths (2007) versus Johnson (2007)) supports this claim. Furthermore, PoS tagging is seldom a goal in itself, but it is a component in a linguistic pipeline.

In order to address these issues, we perform an *extrinsic evaluation* using a well-explored task that involves PoS tags. While PoS tagging is considered a pre-processing step in many natural language processing pipelines, the choice of task is restricted by the lack of real PoS tags in the output of our system. For our purposes we need a task that relies on discriminating between PoS tags rather than the PoS tag semantics themselves, in other words, a task in which knowing whether a word is tagged as noun instead of a verb is equivalent to knowing it is tagged as state 1 instead of state 2. Taking these considerations into account, in Section 4.4 we experiment with shallow parsing in the context of the CoNLL-2000 shared task (Sang and Buchholz, 2000) in which very good performances were achieved using only the words with their PoS tags. Our intuition is that if the iHMM (or any unsupervised PoS tagging method) has a reasonable level of performance, it should improve on the performance of a system that does not use PoS tags. Moreover, if the performance is very good indeed, it should get close to the performance of a system that uses real PoS tags, provided either by human annotation or by a good supervised method.

4.4 Experiments

In all our experiments, the Wall Street Journal (WSJ) part of the Penn Treebank (Marcus et al., 1994) was used. As explained in Section 4.3, we evaluate the output of the iHMM in two ways, as a clustering with respect to a gold standard and as direct replacement of the PoS tags in the task of shallow parsing. In each experiment, we obtain a sample from the iHMM over all the sections of WSJ. The states for sections 15-18 and 20 of the WSJ (training and testing sets respectively in the CoNLL shared task) are used for the evaluation based on shallow parsing, while the remaining sections are used for evaluation against the WSJ gold standard PoS tags using clustering evaluation measures.

We performed three runs with the iHMM: one run with DP prior and fixed γ, α, one with PY prior and fixed d, γ, α and one with DP prior but where we learn the hyper parameters γ, α from the data. Our inference algorithm uses 1000 burn-in iterations after which we collect a sample every 1000 iterations. Our inference procedure is annealed during the first 1000 burn-in and 2400 iterations by powering the likelihood of the output distribution with a number that smoothly increases from 0.4 to 1.0 over the 3400 first iterations. The numbers of iterations reported in the remainder of the section refer to the iterations after burn-in. We initialised the sampler by: a) sampling the hyper-parameters from the prior where applicable, b) uniformly assign each word one out of 20 iHMM

states. For the DP run with fixed parameters, we chose $\alpha = 0.8$ to encourage some sparsity in the transition matrix and $\gamma = 5.0$ to allow for enough hidden states. For the PY run with fixed parameters, we chose $\alpha = 0.8$ for similar reasons and $d = 0.1$ and $\gamma = 1.0$. We point out that one weakness of MCMC methods is that they are hard to test for convergence. We chose to run the simulations until they became prohibitively expensive to obtain a new sample.

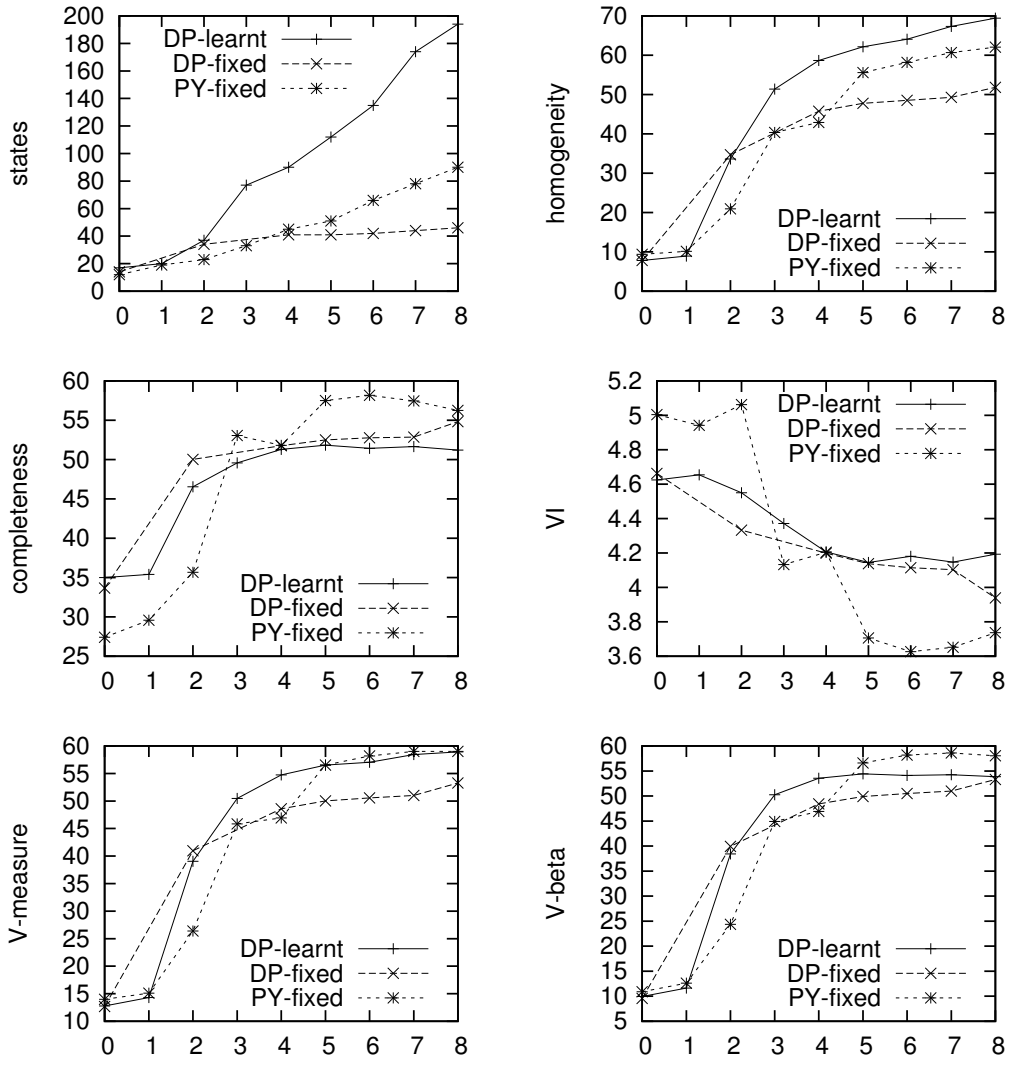


Table 4.2: Performance of the three iHMM runs according to clustering evaluation measures against number of iterations (in thousands).

First, we present results using clustering evaluation measures which appear in the figures of Table 4.2. The three runs exhibit different behaviour. The number of states reached by the iHMM with fixed parameters using the DP prior stabilises close to 50 states, while for the experiment with learnt hyper parameters the number of states grows more rapidly, reaching 194 states after 8,000 iterations. With the PY prior, the number

of states reached grows less rapidly reaching 90 states. Note that for the latter two, the number of states of the iHMM seems to still be growing at the end of our experiment. On one hand this is a disappointing result in that the computational complexity of our methods do not scale sufficiently to explore the stationary distribution of the Markov chain. On the other hand, we argue that *all* runs achieve better performances with respect to *all* measures used as the number of iterations grows. We argue that notwithstanding the fact that our chains have not yet mixed properly, these results suggest that the Markov chains explored so far exhibit appealing empirical properties.

It is interesting to notice how the measures exhibit different biases, in particular that VI penalises the larger numbers of states discovered in the DP run with learnt parameters as well as the run with the PY prior, compared to the more lenient scores provided by V-measure and V-beta. The latter though assigns lower scores to the DP run with learnt parameters because it takes into account that the high homogeneity is achieved using even more states.

The closest experiment to ours is the one by Gao and Johnson (2008) who run their Bayesian HMM over the whole WSJ and evaluate against the full gold standard, the only difference being is that we exclude the CoNLL shared task sections from our evaluation, which leaves us with 19 sections instead of 24. Their best VI score was 4.04 which they achieved using the collapsed, sentence-blocked Gibbs sampler with the number of states fixed to 50. The VI score achieved by the iHMM with fixed parameters using the PY prior reaches 3.73, while using the DP prior VI reaches 4.32 with learnt parameters and 3.93 with fixed parameters. These results, even if they are not directly comparable, are on par with the state-of-the-art, which encouraged us to proceed with the extrinsic evaluation.

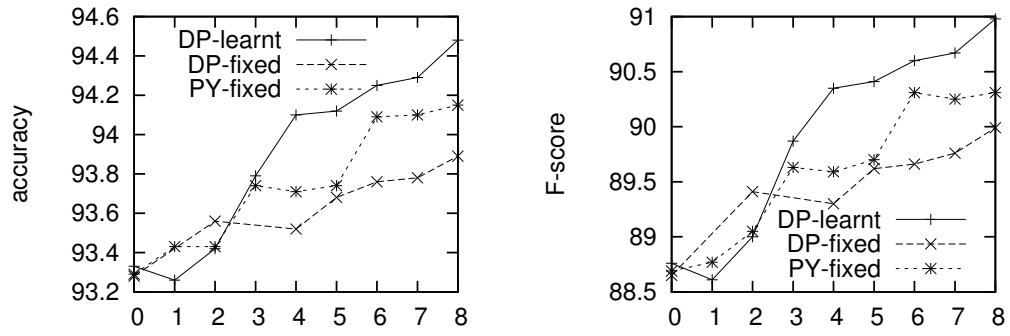


Table 4.3: Performance of the output of the three iHMM runs when used in shallow parsing against number of iteretions (in thousands).

For the experiments with shallow parsing we used the CRF++ toolkit[1] which has an efficient implementation of the model introduced by Sha and Pereira (2003) for this

[1]http://crfpp.sourceforge.net/

task. First we ran an experiment using the words and the PoS tags provided in the shared task data and the performances obtained were 96.07% accuracy and 93.81% F-measure. The PoS tags were produced using the Brill tagger (Brill, 1992) which uses transformation-based learning and was trained using the WSJ corpus. Then we ran an experiment removing the PoS tags altogether, and the performances were 93.25% accuracy and 88.58% F-measure respectively. This gave us some indication as to what the contribution of the PoS tags is in the context of the shallow parsing task at hand.

The experiments using the output of the iHMM as PoS tags for shallow parsing are presented in table 4.3. The best performance achieved was 94.48% and 90.98% in accuracy and F-measure, which is 1.23% and 2.4% better respectively than just using words, but worse by 1.57% and 2.83% compared to using the supervised PoS tagger output. Given that the latter is trained on WSJ we believe that this is a good result. Interestingly, this was obtained by using the last sample from the iHMM run using the DP prior with learnt parameters which has worse overall clustering evaluation scores, especially in terms of VI. This sample though has the best homogeneity score (69.39%). We believe that homogeneity is more important than the overall clustering score due to the fact that, in the application considered, it is probably worse to assign tokens that belong to different PoS tags to the same state, e.g. verb and adverbs, rather than generate more than one state for the same PoS. This is likely to be the case in tasks where we are interested in distinguishing between PoS tags rather than the actual tag itself. Also, clustering evaluation measures tend to score leniently consistent mixing of members of different classes in the same cluster. However, such mixing results in consistent noise when the clustering output becomes input to a machine learning method, which is harder to deal with.

4.5 Discussion

In the context of shallow parsing we saw that the performance of the iHMM does not match the performance of a supervised PoS tagger but does lead to a performance increase over a model using only words as features. Given that it was constructed without any need for human annotation, we believe this is a good result. At the same time though, it suggests that it is still some way from being a direct drop-in replacement for a supervised method. We also showed that the iHMM with Pitman-Yor base distribution outperforms the baseline iHMM on various measures. This is an interesting result since we don't know the finite marginal distribution of the Pitman-Yor distribution; it would thus be impossible to mimic this model using a finite HMM.

Our experiments also suggest that the number of states in a Bayesian non-parametric model can be rather unpredictable. On one hand, this is a strong warning towards

inference algorithms which perform finite truncation of non-parametric models. On the other hand, the remarkable difference in behaviour between the DP with fixed and learnt priors suggests that more research is needed towards understanding the influence of hyper parameters in Bayesian non-parametric models.

Also, the HMM and iHMM's are statistical models: they are not designed to correspond to our linguistic notion of PoS tags. They just happen to pick up very similar statistical properties. Great caution must be used when "peeking" inside a statistical model and interpreting the latent states. For future work we would suggest a semi-supervised approach to PoS tagging. In this model we let the transition matrix of the iHMM depend on annotated PoS tags. This model allows us to: a) use annotations whenever they are available and do unsupervised learning otherwise; b) use the power of non-parametric methods to possibly learn more fine grained statistical structure than tag sets created manually. This idea is similar to the context free grammar state-splitting ideas in Finkel et al. (2007), Liang et al. (2007).

Finally, another area for future investigation is doing unsupervised PoS tagging using web-scale data set. Although the WSJ corpus is reasonably sized, our computational methods do not currently scale to problems with one or two orders of magnitude more data. Our initial work (Bratières et al., 2010) has extended the beam sampler to work in a distributed environment using the map reduce paradigm.

Chapter 5

The Infinite Factorial Hidden Markov Model

We motivated the HMM in chapter 3 by introducing the part-of-speech tagging application. The assumption there was that a sentence can be produced by first generating a sequence of part-of-speech tags using a Markov chain and then generating the actual words conditioned on the part-of-speech tag. In other words, the hidden structure for a sentence was represented by one specific part-of-speech tag per word.

In this chapter we will look at some applications where having one latent variable per time step is too limited. Consider the so-called *cocktail party problem*. In this application we have people attending a cocktail party with all people chatting in little groups. Multiple people in separate groups might be talking at the same time. One or more microphones record all the conversations simultaneously and the goal is to extract each individuals speech signal from the recording. In this example, we will argue later on in the chapter that a good latent representation is to use multiple latent Markov models, each modelling the speech for one person. The latent Markov chains are then mixed together to form the observed microphone signal. This type of model has previously been called a factorial Markov model. In this chapter we introduce a Bayesian nonparametric Markov model with a factorial representation: it extends previous work by explicitly learning the number of latent chains. This chapter is based on work published in Van Gael et al. (2008b).

5.1 The Factorial Hidden Markov Model

The factorial hidden Markov model (FHMM), developed in Ghahramani and Jordan (1997), addresses the limited representational power of the hidden Markov model. The FHMM extends the HMM by representing the hidden state in a factored form. This way, information from the past is propagated in a distributed manner through a set of

$$
\begin{aligned}
&\forall m \in \{1 \cdots M\}, \\
&\quad \forall t \in \{1 \cdots T\}, \\
&\qquad s_0^{(m)} &&= 1 \\
&\qquad s_t^{(m)} | \boldsymbol{\pi} &&\sim \pi^{(m)}_{s_{t-1}^{(m)}} \\
&\qquad y_t | s_t, \boldsymbol{\theta} &&\sim F(y_t | \theta_{s_t^{(1:M)}})
\end{aligned}
$$

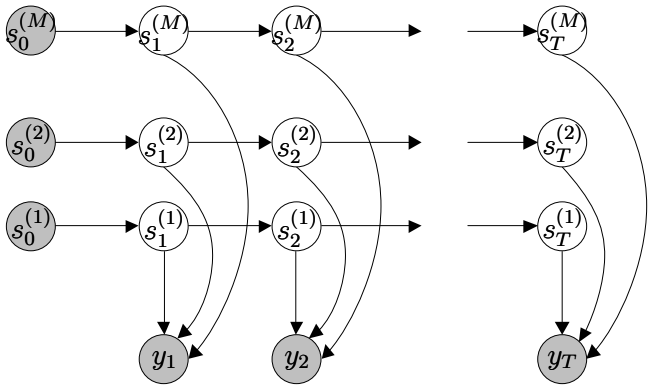


Figure 5.1: The graphical model for a Factorial Hidden Markov Model.

parallel Markov chains. The parallel chains can be viewed as latent features which evolve over time according to Markov dynamics. Formally, the FHMM defines a probability distribution over observations $y_1, y_2, \cdots y_T$ as follows: M latent chains $s^{(1)}, s^{(2)}, \cdots, s^{(M)}$ evolve according to Markov dynamics and at each time step t, the Markov chains generate an output y_t using some likelihood model F parameterised by a joint state-dependent parameter $\theta_{s_t^{(1:m)}}$. The graphical model of the FHMM in figure 5.1 shows how it is a special case of a dynamic Bayesian network.

Unfortunately, the dimensionality M of our factorial representation, or equivalently, the number of parallel Markov chains, is a free parameter for the FHMM which we would prefer learning from data rather than specifying beforehand. In this chapter we derive the basic building blocks for a Bayesian nonparametric factor model for time series which we call the Markov Indian Buffet Process (mIBP). Using this distribution we build a nonparametric extension of the FHMM which we call the Infinite Factorial Hidden Markov Model (iFHMM). This construction allows us to learn a factorial representation for time series.

5.2 The Markov Indian Buffet Process

Similar to the IBP, we define a distribution over binary matrices to model whether a feature at time t is on or off. In our representation rows correspond to time steps and columns to features or Markov chains. We want the distribution over matrices to satisfy the following two properties: (1) the potential number of columns (representing latent features) should be able to be arbitrary large; (2) the rows (representing time steps) should evolve according to a Markov process.

Below, we formally derive the mIBP distribution in two steps: first we describe a

distribution over binary matrices with a finite number of columns. We choose the hyper parameters carefully so we can easily integrate out the parameters of the model. In a second phase we take the limit as the number of features goes to infinity as we did in section 2.3.

5.2.1 A Finite Model

Let S represent a binary matrix with T rows (data points or time steps) and M columns (features) such that s_{tm} represents the hidden state at time t for Markov chain m. Each Markov chain evolves according to the transition matrix

$$W^{(m)} = \begin{pmatrix} 1 - a_m & a_m \\ 1 - b_m & b_m \end{pmatrix}, \tag{5.1}$$

where $W^{(m)}_{ij} = p(s_{t+1,m} = j | s_{tm} = i)$. We give the parameters of $W^{(m)}$ distributions $a_m \sim \mathsf{Beta}(\alpha/M, 1)$ and $b_m \sim \mathsf{Beta}(\gamma, \delta)$. Each chain starts with a dummy zero state $s_{0m} = 0$. The hidden state sequence for chain m is generated by sampling T steps from a Markov chain with transition matrix $W^{(m)}$. Summarising, the generative specification for this process is

$$\begin{aligned} \forall m \in \{1, 2, \cdots, M\} \quad &: \quad a_m \sim \mathsf{Beta}\left(\frac{\alpha}{M}, 1\right), b_m \sim \mathsf{Beta}(\gamma, \delta), \\ \forall t \in \{1, 2, \cdots, T\} \quad &: \quad s_{0m} = 0, s_{tm} \sim \mathsf{Bernoulli}(a_m^{1-s_{t-1,m}} b_m^{s_{t-1,m}}). \end{aligned} \tag{5.2}$$

Next, we evaluate the probability of the state matrix S with the transition matrix parameters $W^{(m)}$ marginalised out. We introduce the following notation, let $c_m^{00}, c_m^{01}, c_m^{10}, c_m^{11}$ be the number of $0 \to 0, 0 \to 1, 1 \to 0$ and $1 \to 1$ transitions in binary chain m (including the transition from the dummy state to the first state). We can then write

$$p(S|a, b) = \prod_{m=1}^{M} (1 - a_m)^{c_m^{00}} a_m^{c_m^{01}} (1 - b_m)^{c_m^{10}} b_m^{c_m^{11}}, \tag{5.3}$$

and integrate out a and b with respect to the conjugate priors defined in equation (5.2) to find

$$p(S|\alpha, \gamma, \delta) = \prod_{m=1}^{M} \frac{\frac{\alpha}{M}\Gamma(\frac{\alpha}{M} + c_m^{01})\Gamma(c_m^{00} + 1)\Gamma(\gamma + \delta)\Gamma(\delta + c_m^{10})\Gamma(\gamma + c_m^{11})}{\Gamma(\frac{\alpha}{M} + c_m^{00} + c_m^{01} + 1)\Gamma(\gamma)\Gamma(\delta)\Gamma(\gamma + \delta + c_m^{10} + c_m^{11})}. \tag{5.4}$$

5.2.2 Taking the Infinite Limit

Analogous to the IBP we compute the limit for $M \to \infty$ of the finite model in equation (5.4). Recall from our discussion in section 2.3 that the probability of a single matrix in the limit as $M \to \infty$ is zero. This is not a problem since we are only interested in the probability of a whole class of matrices, namely those matrices that can be

transformed into each other through column permutations. In other words, our factorial model is exchangeable in the columns as we don't care about the ordering of the features. Hence, we compute the infinite limit for left-ordered form equivalence classes.

Recall from section 2.3 that the left-ordered form of a binary S matrix can be defined as follows: we interpret one column of length T as encoding a binary number: column m encodes the number $2^{T-1}s_{1m} + 2^{T-2}s_{2m} + \cdots + s_{Tm}$. We called this number the history of the column. Then, we denote with M_h the number of columns in the matrix S that have the same history. We defined a matrix to be a left-ordered matrix if its columns are sorted in decreasing history values. If S is a left-ordered matrix then we denoted with $[S]$ the set of all matrices that can be transformed into S using only column permutations; we call $[S]$ the left-ordered equivalence class. We argued in section 2.3 that the number of elements in the left-ordered equivalence class of S is equal to $\frac{M!}{\prod_{h=0}^{2^T-1} M_h!}$. We thus find the probability of the equivalence class of S to be

$$\begin{aligned} p([S]) &= \sum_{S\in[S]} p(S|\alpha,\gamma,\delta) \\ &= \frac{M!}{\prod_{h=0}^{2^T-1} M_h!} \prod_{m=1}^{M} \frac{\frac{\alpha}{M}\Gamma(\frac{\alpha}{M}+c_m^{01})\Gamma(c_m^{00}+1)\Gamma(\gamma+\delta)\Gamma(\delta+c_m^{10})\Gamma(\gamma+c_m^{11})}{\Gamma(\frac{\alpha}{M}+c_m^{00}+c_m^{01}+1)\Gamma(\gamma)\Gamma(\delta)\Gamma(\gamma+\delta+c_m^{10}+c_m^{11})} \end{aligned} \quad (5.5)$$

This form allows us to compute a meaningful limit as $M \to \infty$. In order to do so, we need to distinguish between two types of Markov chains: those with all zero states ($c_m^{00} = T$) and those with nonzero states ($c_m^{00} < T$). Let us denote with M_0 the number of Markov chains with all zero states and with M_+ the number of Markov chains which have nonzero states; note that $M = M_0 + M_+$. We can rewrite the product in equation (5.5) based on this dichotomy

$$\begin{aligned} &\prod_{m=1}^{M} \frac{\frac{\alpha}{M}\Gamma(\frac{\alpha}{M}+c_m^{01})\Gamma(c_m^{00}+1)\Gamma(\gamma+\delta)\Gamma(\delta+c_m^{10})\Gamma(\gamma+c_m^{11})}{\Gamma(\frac{\alpha}{M}+c_m^{00}+c_m^{01}+1)\Gamma(\gamma)\Gamma(\delta)\Gamma(\gamma+\delta+c_m^{10}+c_m^{11})} \\ = &\left(\frac{\alpha\Gamma(\frac{\alpha}{M})\Gamma(T+1)}{M\Gamma(\frac{\alpha}{M}+T+1)}\right)^{M_0} \prod_{m=1}^{M_+} \frac{\frac{\alpha}{M}\Gamma(\frac{\alpha}{M}+c_m^{01})\Gamma(c_m^{00}+1)\Gamma(\gamma+\delta)\Gamma(\delta+c_m^{10})\Gamma(\gamma+c_m^{11})}{\Gamma(\frac{\alpha}{M}+c_m^{00}+c_m^{01}+1)\Gamma(\gamma)\Gamma(\delta)\Gamma(\gamma+\delta+c_m^{10}+c_m^{11})} \\ = &\left(\frac{\Gamma(\frac{\alpha}{M}+1)\Gamma(T+1)}{\Gamma(\frac{\alpha}{M}+T+1)}\right)^{M} \prod_{m=1}^{M_+} \frac{\Gamma(\frac{\alpha}{M}+T+1)\Gamma(\frac{\alpha}{M}+c_m^{01})\Gamma(c_m^{00}+1)\Gamma(\gamma+\delta)\Gamma(\delta+c_m^{10})\Gamma(\gamma+c_m^{11})}{\Gamma(\frac{\alpha}{M})\Gamma(T+1)\Gamma(\frac{\alpha}{M}+c_m^{00}+c_m^{01}+1)\Gamma(\gamma)\Gamma(\delta)\Gamma(\gamma+\delta+c_m^{10}+c_m^{11})} \\ = &\left(\frac{T!}{\prod_{t=1}^{T}(t+\frac{\alpha}{M})}\right)^{M} \left(\frac{\alpha}{M}\right)^{M_+} \prod_{m=1}^{M_+} \frac{\Gamma(\frac{\alpha}{M}+T+1)\prod_{i=1}^{c_m^{01}-1}(i+\frac{\alpha}{M})c_m^{00}!\Gamma(\gamma+\delta)\Gamma(\delta+c_m^{10})\Gamma(\gamma+c_m^{11})}{T!\Gamma(\frac{\alpha}{M}+c_m^{00}+c_m^{01}+1)\Gamma(\gamma)\Gamma(\delta)\Gamma(\gamma+\delta+c_m^{10}+c_m^{11})}. \end{aligned}$$

Substituting this equation back into equation (5.5) and using the results in appendix C we compute the final limit

$$\lim_{M\to\infty} p([S]) = \frac{\alpha^{M_+}\exp\{-\alpha H_T\}}{\prod_{h=1}^{2^T-1} M_h!} \prod_{m=1}^{M_+} \frac{(c_m^{01}-1)!c_m^{00}!\Gamma(\gamma+\delta)\Gamma(\delta+c_m^{10})\Gamma(\gamma+c_m^{11})}{(c_m^{00}+c_m^{01})!\Gamma(\gamma)\Gamma(\delta)\Gamma(\gamma+\delta+c_m^{10}+c_m^{11})}. \quad (5.6)$$

We call this distribution the Markov Indian buffet process. Note from equation (5.6) that our model is exchangeable in the columns and Markov exchangeable[1] in the rows.

5.2.3 The Stochastic Process

Next we derive the distribution in equation (5.6) through a stochastic process that is analogous to the Indian Buffet Process but slightly more complicated for the actors involved. In this stochastic process T customers enter an Indian restaurant with an infinitely long buffet of dishes organised in a line. The first customer enters the restaurant and takes a serving from each dish starting at the left of the buffet and stopping after a $\mathsf{Poisson}(\alpha)$ number of dishes as his plate becomes overburdened. A waiter stands near the buffet and takes notes as to how many people have eaten which dishes. The t'th customer enters the restaurant and starts at the left of the buffet. At dish m he looks at the customer in front of him to see whether he has served himself that dish.

- If so, he asks the waiter how many people have previously served themselves dish m when the person in front of them did (the waiter replies to him the number c_m^{11}) and how many people didn't serve themselves dish m when the person in front of them did (the waiter replies to him the number c_m^{10}). The customer then serves himself dish m with probability $(c_m^{11}+\delta)/(\gamma+\delta+c_m^{10}+c_m^{11})$.
- Otherwise, he asks the waiter how many people have previously served themselves dish m when the person in front of them did not (the waiters replies to him the number c_m^{01}) and how many people didn't serve themselves dish m when the person in front of them did not either (the waiter replies to him the number c_m^{00}). The customer then serves himself dish m with probability $c_m^{00}/(c_m^{00}+c_m^{01})$.

The customer then moves on to the next dish and does exactly the same. After the customer has passed all dishes people have previously served themselves from, he tries $\mathsf{Poisson}(\alpha/t)$ new dishes. If we denote with $M_1^{(t)}$ the number of new dishes tried by the t'th customer, the probability of any particular matrix being produced by this process is

$$p([S]) = \frac{\alpha^{M_+}\exp\{-\alpha H_T\}}{\prod_{t=1}^{T} M_1^{(t)}!}\prod_{m=1}^{M_+}\frac{\frac{\alpha}{M}\Gamma(\frac{\alpha}{M}+c_m^{01})\Gamma(c_m^{00}+1)\Gamma(\gamma+\delta)\Gamma(\delta+c_m^{10})\Gamma(\gamma+c_m^{11})}{\Gamma(\frac{\alpha}{M}+c_m^{00}+c_m^{01}+1)\Gamma(\gamma)\Gamma(\delta)\Gamma(\gamma+\delta+c_m^{10}+c_m^{11})}.$$

We can recover equation (5.6) by summing over all possible matrices that can be generated using the Markov Indian Buffet process that are in the same left-ordered equivalence class; there are exactly $\frac{\prod_{t=1}^{T} M_1^{(t)}!}{\prod_{h=1}^{2^T-1} M_h!}$ matrices in the same left-ordered equivalence class. Multiplying this by equation (5.2.3) we recover equation (5.6). This construction shows that the effective dimension of the model (M_+) follows a $\mathsf{Poisson}(\alpha H_T)$ distribution.

[1] A sequence is Markov exchangeable if its distribution is invariant under permutations of the transitions (Diaconis and Freedman, 1980).

5.2.4 The Stick Breaking Representation

Although the representation above is convenient for theoretical analysis it is not very practical for inference. Interestingly, we can adapt the stick breaking construction for the IBP (Teh et al., 2007) to the mIBP. This will be very important as it will allow us to use a combination of slice sampling and dynamic programming to do inference.

The first step in the stick breaking construction is to find the distribution of $a_{(1)} > a_{(2)} > \ldots$: the order statistics of the parameters a. Since the distribution on the variables a_m in our model is identical to the distribution of the feature parameters in the IBP model, we can use the result in Teh et al. (2007) that these variables have the distribution

$$\begin{aligned} a_{(1)} &\propto \mathsf{Beta}(\alpha, 1), \\ p(a_{(m)}|a_{(m-1)}) &= \alpha a_{(m-1)}^{-\alpha} a_{(m)}^{\alpha-1} \mathbb{I}(0 \leq a_{(m)} \leq a_{(m-1)}). \end{aligned} \tag{5.7}$$

Using a change of variables, it can be shown that this is equivalent to the following stick breaking construction

$$\nu_{(m)} \propto \mathsf{Beta}(\alpha, 1) \qquad a_{(m)} \propto \prod_{l=1}^{m} \nu_{(l)}. \tag{5.8}$$

The variables b_m are all independent draws from a $\mathsf{Beta}(\gamma, \delta)$ distribution which is independent of M. Hence if we denote with $b_{(m)}$ the b variable corresponding to the m'th largest a value (in other words: the b value corresponding to $a_{(m)}$) then it follows that $b_{(m)} \sim \mathsf{Beta}(\gamma, \delta)$.

5.2.5 Discussion

The Markov Indian buffet process introduced in this chapter is a first attempt at defining a nonparametric dynamic factor model. Its strength is that a stick breaking representation is readily available. One weakness is that the stationary distribution for the different chains don't have a nice analytic form: Miller et al. (2008b) addresses this issue.

5.3 The Infinite Factorial Hidden Markov Model

In this section, we explain how to use the mIBP as a building block in a full blown probabilistic model. The mIBP provides us with a matrix S which we interpret as an arbitrarily large set of parallel Markov chains. First we augment our binary representation with a more expressive component which can describe feature specific properties. We do this by introducing a base distribution H from which we sample a parameter $\theta_m \sim H$ for each Markov chain. Moreover, since we know the length of the mIBP sequence T, we can choose to make the base distribution H be a multivariate distribution over T

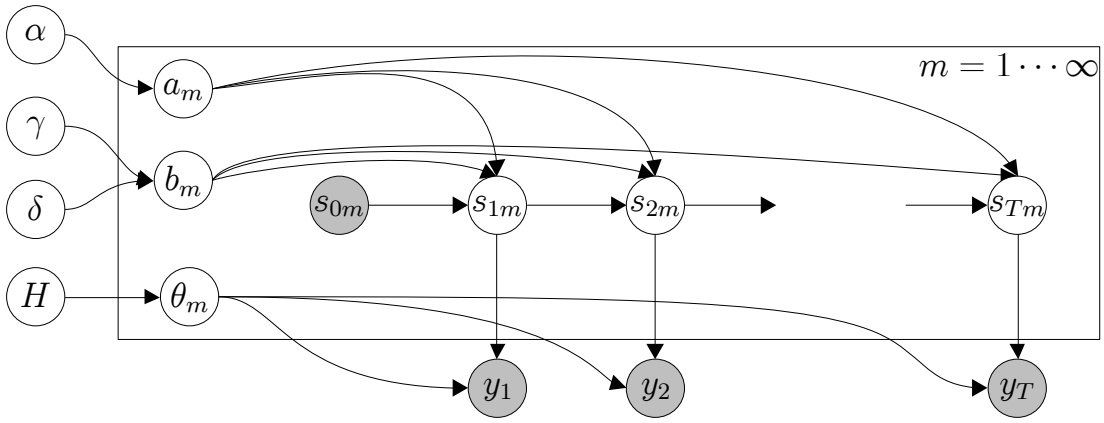


Figure 5.2: The graphical model for an infinite Factorial Hidden Markov Model.

variables. This is a rather flexible setup where the base distribution not only introduces a parameter for every chain but also for every time step. We will show an example of this type of model in section 5.5.

Now that we have a model with a more expressive latent structure, we want to add a likelihood model F which describes the distribution over the observations conditional on the latent structure. Formally, $F(y_t|\theta, s_{t,\cdot})$ describes the probability of generating y_t given the model parameters θ and the current latent feature state $s_{t,\cdot}$. We note that there are two important conditions which the likelihood must satisfy in order for the limit $M \to \infty$ to be valid: (1) the likelihood must be invariant to permutations of the features; (2) the likelihood cannot depend on θ_m if $s_{tm} = 0$. Figure 5.2 shows the graphical model for our construction: we call it the infinite Factorial Hidden Markov Model (iFHMM).

5.4 Inference

We conclude this chapter with an inference scheme for the iFHMM. As we discussed in chapter 3, inference for nonparametric models requires special treatment as the potentially unbounded dimensionality of the model makes it hard to use exact inference methods. Traditionally, in nonparametric factor models inference is done using Gibbs sampling, sometimes augmented with Metropolis Hastings steps to improve performance. As we argued in chapter 3 it is commonly known that naive Gibbs sampling in a time series model is notoriously slow due to potentially strong couplings between successive time steps. We will use a similar methodology as in chapter 3 where a slice sampler adaptively truncates the infinite dimensional model after which dynamic programming performs exact inference. Since a stick breaking construction for the iFHMM is readily available

we can use a very similar approach for the iFHMM. The central idea is the following: we introduce an auxiliary slice variable with the following distribution

$$\mu \sim \mathsf{Uniform}(0, \min_{m:\exists t, s_{tm}=1} a_m). \tag{5.9}$$

It is not essential that we sample from the uniform distribution; in fact, for some of our experiments we use the more flexible Beta distribution. The resulting joint distribution of the iFHMM with augmented auxiliary variables is

$$p(\mu, a, b, S) = p(\mu|a, S)p(a, b, S). \tag{5.10}$$

It is clear from the equation above that one recovers the original mIBP distribution when we integrate out μ. When we condition the joint distribution on μ we find

$$p(S|Y, \mu, a, b) \propto p(S|Y, a, b)\frac{\mathbb{I}(0 \leq \mu \leq \min_{m:\exists t, s_{tm}=1} a_m)}{\min_{m:\exists t, s_{tm}=1} a_m} \tag{5.11}$$

which forces all columns of S for which $a_m < \mu$ to be in the all zero state. Since there can only be a finite number of $a_m > \mu$ this effectively implies that we need only re-sample a finite number of columns of S.

Algorithm 5 Slice sampling algorithm for the iFHMM.

Initialise a, b, S, θ randomly.
loop
  Sample the transition matrix parameters $a|S, \alpha$ and $b|S, \gamma, \delta$
  Sample the auxiliary variable $\mu|S, a$
  Sample the state sequence $S|a, b, \mu, Y$
  Sample the parameters $\theta|S, H, Y$
end loop

Algorithm 5 shows the high level overview of the iFHMM sampler. One can think of the algorithm as a blocked Gibbs sampling algorithm with some step using dynamic programming to sample blocks of variables all at once. We now consider each of the sampling steps in more detail.

Sampling $a|S, \alpha$ and $b|S, \gamma, \delta$: in this step we estimate the transition parameters for each column of S. The easy case is for the columns that are switched on at some point in time. Using Bayes rule we can compute the posterior

$$\begin{aligned} a_k &\sim \mathsf{Beta}(c_m^{01}, c_m^{00}) \\ b_k &\sim \mathsf{Beta}(c_m^{11} + \gamma, c_m^{10} + \delta). \end{aligned}$$

We don't effectively sample the a and b for the unused features because there are an infinite number of them. As we will only need those a that are larger than μ we need

the posterior distribution of the a in decreasing order so we can retrospectively sample them (Papaspiliopoulos et al., 2008). The stick-breaking construction in equation (5.8) is exactly this.

Sampling $\mu|S, a$: as we described above, we sample

$$\mu \sim \mathsf{Uniform}(0, \min_{m:\exists t, s_{tm}=1} a_m). \tag{5.12}$$

As with the beam sampler for the iHMM the uniform distribution is not critical. We can improve the mixing time by biasing μ downward to let the algorithm consider more features or improve the computational complexity by decreasing the number of features considered.

Sampling $S|a, b, \mu, Y$: we only need to re-sample a finite number of columns m where $a_m > \mu$. Ghahramani and Jordan (1997) showed that it is computationally intractable to compute the full posterior for more than a few Markov chains at once. Hence we propose to fix all but a few chains and Gibbs sample them conditioned on the fixed chains. Currently the most attractive option is to use the forward-filtering backward-sampling algorithm (see appendix B) to resample the chains.

Sampling $\theta|S, H, Y$: The final step in the sampler for the iFHMM re-samples the parameters θ conditioned on the feature assignment matrix S and the base distribution H. When the base distribution H is conjugate to the likelihood model F there is a closed form for the posterior distribution $p(\theta|S, H, Y)$ from which to sample; otherwise one has to resort to custom inference schemes to sample from this posterior.

5.5 Blind Source Separation using the iFHMM

A common task in audio and signal processing is the extraction of different signals from one or more recordings or data streams. This is often called blind source separation. The *cocktail party problem* is an example of such a task: a number of people attend a cocktail party and are chatting in little groups. Multiple people in separate groups might be talking at the same time. One or more microphones record all the conversations simultaneously and the goal is to extract each individual's speech signal from the recording. We present one solution to this problem using a nonparametric time series model based on independent component analysis (ICA, Hyvärinen and Oja (2000)).

5.5.1 The Independent Component Analysis iFHMM

Independent Component Analysis means different things to different people. Originally invented as an algorithm to un-mix a signal into a set of independent signals, it will be more insightful for our purpose to think of ICA in terms of the probabilistic model which we describe below.

Assume that M signals are represented through the vectors x_m; grouping them together we can represent the signals using the matrix $\boldsymbol{X} = [x_1 x_2 \cdots x_M]$. Next, we linearly combine the signals using a mixing matrix W to generate the observed signal $Y = XW$. Additionally, we will assume additive i.i.d. $\mathsf{Normal}(0, \sigma_Y^2)$ noise: $Y = XW + \epsilon$.

A variety of fast algorithms exist which un-mix the observations Y and recover the signal X. Crucial to these algorithms is that the number of signals is known in advance. Knowles and Ghahramani (2007) used the IBP to design the infinite Independent Component Analysis (iICA) model which learns an appropriate number of signals from exchangeable data. Our ICA iFHMM model extends the iICA for time series; we will refer to it as the ICA iFHMM.

The ICA iFHMM generative model can be described as follows: we sample $S \sim \mathsf{mIBP}$ and point-wise multiply (denoted by $\odot$) it with a signal matrix X. Each entry in X is an i.i.d. sample from a $\mathsf{Laplace}(0, 1)$ distribution. One could choose many other distributions for X, but since we will model speech data, which is known to be heavy tailed, the Laplace distribution is a reasonable and convenient choice. Speakers will be speaking infrequently so point-wise multiplying a heavy tailed distribution with a sparse binary matrix achieves our goal of producing a sparse heavy tailed distribution, this model is also known as a *spike-and-slab model*. Next, we introduce a mixing matrix W which has a row for each signal in $\boldsymbol{S} \odot \boldsymbol{X}$ and a column for each observed dimension in Y. The entries for W are sampled i.i.d. from a $\mathsf{Normal}(0, \sigma_W^2)$ distribution. Finally, we combine the signal and mixing matrices as in the finite case to form the observation matrix $Y = (S \odot X)\boldsymbol{W} + \epsilon$ where ϵ is i.i.d. $\mathsf{Normal}(0, \sigma_Y^2)$. In terms of the general iFHMM model defined in the previous section, the base distribution H is a joint distribution over columns of X and rows of W. The likelihood F performs the point wise multiplication, mixes the signals and adds the noise. It can be checked that our likelihood satisfies the two technical conditions for proper iFHMM likelihoods described in section 5.3.

We perform inference using the algorithm described in section 5.4. We experimented with 3 different algorithms for re-sampling S in algorithm 5. The first, a naive Gibbs sampler did not perform well. This was to be expected for a time series model. The second algorithm, which we used for our experiments, is a blocked Gibbs sampler which fixes all but one column of S and runs a forward-filtering backward-sampling sweep on the remaining column. This allows us to analytically integrate out one column of X in the dynamic program and re-sample it from the posterior afterwards. In this setting, W can be sampled exactly conditional on X, S, Y. A third algorithm runs dynamic programming on multiple chains at once. We originally designed this algorithm as it has the potential to merge two features in one sweep. However, we found that because we cannot integrate out X and W in this setting, the mixing time was not faster than our second algorithm. Note that because the bulk of the computation is used for estimating

X and W the dynamic programming based algorithms are effectively as fast as the naive Gibbs sampler. Finally note that the rows of X are mutually independent given W, Y, S; hence we can sample them in parallel. Our implementation leverages this to speed up the computation on multi-core systems.

5.5.2 Experiments

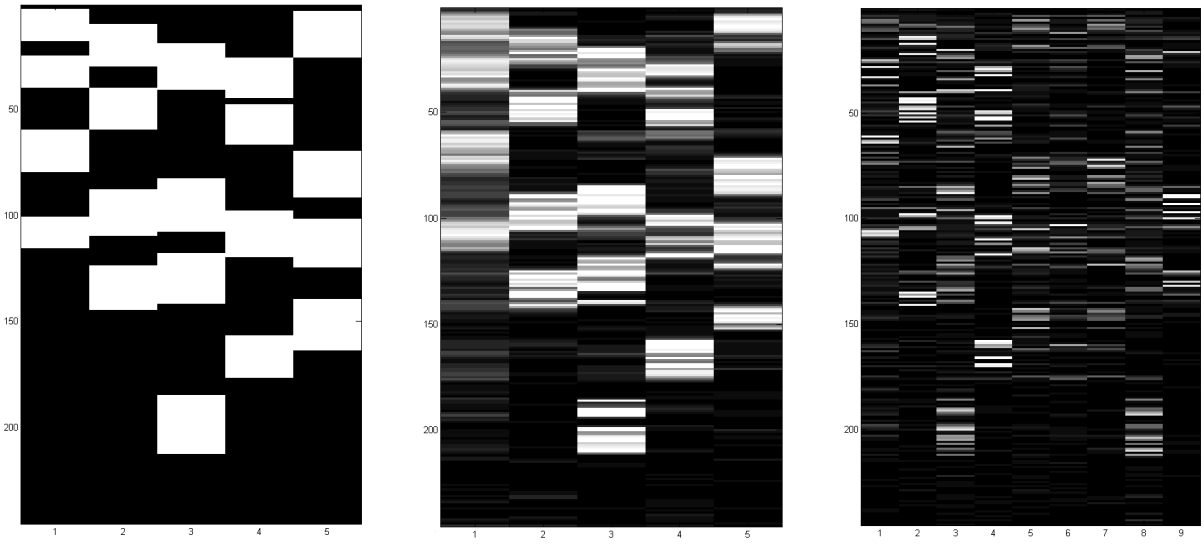
To test the ICA-iFHMM model and inference algorithms, we address the cocktail party problem. For this problem we record multiple people who are simultaneously speaking using a set of microphones. Given the mixed speech signals the goal is to separate out the individual speech signals. Key to our presentation is that we want to illustrate that using nonparametric methods, we can learn the number of speakers from a small amount of data. Our first experiment learns to recover the signals in a setting with more microphones then speakers, our second experiment uses less microphones then speakers.

The experimental setup was the following: we downloaded data from 5 speakers from the Speech Separation Challenge website[2]. The data for each speaker consists of 4 sentences which we appended with random pauses in between each sentence. The left plot in Figure 33 illustrates which person is talking at what point in time. Next, we artificially mix the data 10 times. Each mixture is a linear combination of each of the 5 speakers using $\mathsf{Uniform}(0, 1)$ mixing weights. We centred the data to have zero mean and unit variance and added i.i.d. $\mathsf{Normal}(0, \sigma_Y^2)$ noise with $\sigma_Y = 0.3$.

In our first experiment we compared the ICA iFHMM with the iICA model using all 10 microphones. We sub sample the data so we learn from 245 data points. We initialised the samplers for both models with an initial S matrix with 10 features and 5% random entries on. We use a $\mathsf{Gamma}(1.0, 4.0)$ prior on α. In both models, we use an $\mathsf{InverseGamma}(2.0, 1.0)$ prior for σ_Y and σ_W. Finally, for the iFHMM we chose a $\mathsf{Gamma}(10.0, 1.0)$ prior on γ and a $\mathsf{Gamma}(1.0, 1.0)$ prior on δ to encode our belief that people speak for larger stretches of time, say the time to pronounce a sentence. We ran the samplers for 5000 burn-in iterations and then gathered 20 samples every 20 iterations.

For both the ICA iFHMM and iICA models, we average the 20 samples and rearrange the features to have maximal overlap with the ground truth features. The middle plot in figure 5.3 shows that the ICA iFHMM model recognises that the data was generated from 5 speakers. Visual inspection of the recovered S matrix also shows that the model discovers who is speaking at what time. The right plot in figure 5.3 illustrates the results of the iICA model on the same data. Although the model discovers some structure in the data it fails to find the right number of speakers (it finds 9) and does a poor job in discovering which speaker is active at which time. We computed the average mutual information between the 5 columns of the true S matrix and the first 5 columns of the

[2] http://www.dcs.shef.ac.uk/m̃artin/SpeechSeparationChallenge.htm

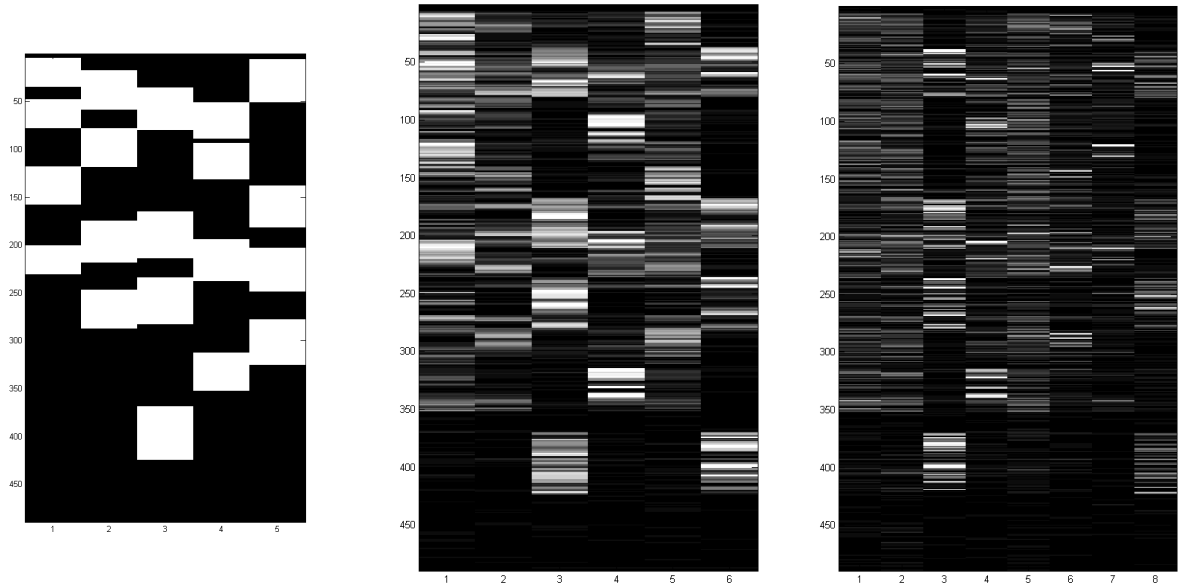


Ground Truth *ICA iFHMM* *iICA*

Figure 5.3: Blind speech separation experiment with 10 microphones; figures represent which speaker is speaking at a certain point in time: columns are speakers, rows are white if the speaker is talking and black otherwise. This corresponds to a visualisation of the S matrix in the iFHMM model.

recovered S matrices. We find that the iFHMM has an average mutual information of 0.296 bits compared to 0.068 bits for the iICA model. The difference between the two models is strictly limited to the difference between using the IBP versus mIBP as the prior distribution on S. We want to emphasise that although one could come up with ad-hoc heuristics to smooth the iICA results, the ICA iFHMM is a principled probabilistic model that does a good job at comparable computational cost.

In a second experiment, we chose to perform blind speech separation using only the first 3 microphones. We sub-sampled a noiseless version of the data to get 489 data points. We ran both the ICA iFHMM and iICA inference algorithms using exactly the same settings as in the previous experiment. The middle and right plots in figure 5.4 show the average of 20 samples rearranged to match the ground truth. In this setting both methods fail to identify the number of speakers although the ICA iFHMM clearly performs better. The ICA iFHMM finds one too many signal: the spurious signal is very similar to the third signal which suggests that the error is a problem of the inference algorithm and not so much of the model itself. The iICA on the other hand performs poorly: it is very hard to find any structure in the recovered S matrix. We compared the mutual information as described above and find that the iFHMM has a mutual information of 0.091 compared to 0.028 for the iICA model. To our knowledge, this is the first generative model which performs blind-source separation with an unknown number of speakers.



Ground Truth *ICA* *iFHMM* *iICA*

Figure 5.4: Blind speech separation experiment with 3 microphones; figures represent which speaker is speaking at a certain point in time: columns are speakers, rows are white if the speaker is talking and black otherwise. This corresponds to a visualisation of the S matrix in the iFHMM model.

5.6 Discussion

In this chapter we reviewed both parametric FHMM as well as introducing the Bayesian non-parametric analogue, the iFHMM. We showed how to construct a sampling based inference algorithm and used this to solve the cocktail party problem where the number of speakers is unknown.

In this chapter we focussed on deriving a Bayesian non-parametric model where each chain has a binary state space. As far as we know, there is no extension of this work to arbitrary finite state spaces such as in the FHMM. Even beyond this, one could imagine building a factorial model where each chain has a potentially infinite state space. Such a model would have the power to both learn the number of chains as well as the state space for each chain.

Chapter 6

Conclusion

In my thesis, I started from the assumption that nonparametric models have great potential in data modelling. Data sets become bigger every day and we need models which can adapt to the complexity progressively as more data becomes available.

The mathematics and statistics community have laid the foundation both in theory as well as in applications for Bayesian nonparametric models. I strongly believe that computer science can contribute a great deal of algorithmic expertise in taking Bayesian nonparametrics further.

In chapter 3 we built on the literature which introduced the iHMM and introduced the first dynamic programming-based algorithm, called the beam sampler, which can do inference for this Bayesian nonparametric model (based on work in Van Gael et al. (2008a)). In this chapter we introduced a second dynamic programming-based algorithm, called the embedded HMM sampler, which in comparison to the beam sampler allows more fine-grained control over the per iteration computational complexity. Since the introduction of the HDP in Teh et al. (2006a) it was unclear whether the HDP-HMM and iHMM define the same distribution over observations. In this chapter, we resolved this open problem and proved that they do define the same probability distribution. Both the embedded HMM sampler as well as the equivalence proof are both novel and unpublished contributions to the machine learning literature.

I believe that the beam sampler or the embedded HMM samplers are the methods of choice for inference for the iHMM. Our work shows that the dynamic programming based samplers mix faster than the Gibbs or collapsed Gibbs samplers that were known previously. In Fox et al. (2008a), the iHMM was truncated into a finite model making a standard dynamic programming based approach feasible. In my opinion the complexity of the beam sampler (less than 100 lines of code in Matlab) is so small that it does not warrant changing the model to make inference practical. We have released a prototype Matlab implementation on MLOSS Van Gael (2010) which we hope stimulates the adoption of the beam sampling algorithm.

In chapter 4, we used the iHMM to tackle an important task in natural language processing called unsupervised part-of-speech tagging (based on work in Van Gael et al. (2009)). The contribution of this work is twofold: first, we showed that the iHMM is a good alternative to the model selection approaches that were offered in previous work (Johnson, 2007, Goldwater and Griffiths, 2007, Gao and Johnson, 2008). Secondly, we believe one needs to be careful in reading into the latent representation. A Bayesian nonparametric model just captures the Markov correlations in the data; although this correlates with part-of-speech tags, they are not the same. Nonetheless, our work shows that the iHMM states can be successfully used as features in an NLP pipeline. In more recent work (Bratières et al., 2010) we have also explored using the Hadoop distributed computing platform for doing large scale unsupervised language processing using the iHMM.

An area of future work which I am currently exploring is to combine parametric and non-parametric Markov chains. The idea here would be to use an HMM as the input for an IO-iHMM. In a part-of-speech tagging context, we could train the HMM using tagged sequences while at the same time learning the IO-iHMM. This model learns more fine-grained structure based on the corpus defined tags resulting in better predictions. A second area of further investigation is to use the non-parametric models for language modelling rather than part-of-speech tagging. Non-parametric models such as the HDP and its extensions have shown great promise (Teh, 2006) for language modelling. We are currently exploring the performance of the iHMM for similar tasks as well.

In our final chapter 5, we introduced a new Bayesian nonparametric model called the infinite Factorial Hidden Markov Model (based on the work in Van Gael et al. (2008b)). The iFHMM and its underlying distribution, called the Markov Indian Buffet Process, allow for modelling parallel chains of latent random variables while keeping the number of parameters in the model small. We described an inference algorithm for the iFHMM and applied the model to the cocktail party problem where the number of speakers was unknown. Although our results show good performance on a small dataset, I have found that for larger problems the inference algorithm becomes a limitation. More generally, in my research I have found that any inference over IBP or mIBP-based models suffers from various problems, e.g. getting stuck in local optima, due to the combinatorial nature of the distribution. I believe that a search based inference algorithm, such as the one introduced in Daume (2007) for Dirichlet processes, is a promising alternative.

In the limited amount of experiments I have done, I have found that almost any problem in unsupervised learning which is set up as a parametric model can be transformed into a non-parametric model without any loss of performance. Moreover, the nonparametric equivalent requires less parameter tuning and output results faster than most model selection-based approaches.

After three years of research I strongly believe that Bayesian nonparametric models have great potential in both machine learning and statistics. I think the most promising future for Bayesian nonparametric models lies in defining distributions over more complicated objects such as trees and graphs and I am interested to explore these in the future.

Appendix A

The Dirichlet Distribution

The Dirichlet distribution is an important distribution in this thesis: e.g. it surfaces as the prior for the rows of the transition matrix in the HMM (chapter 1) and the emission distribution in various natural language processing problems (chapter 4). It is therefore instructive that we review a few of its key properties.

The Dirichlet distribution is a distribution over finite probability vectors; in other words, it is a "distribution over distributions". A Dirichlet distribution of dimension K is parametrised by K positive real numbers $\alpha_1, \cdots, \alpha_K$. We write

$$\pi \sim \mathsf{Dirichlet}(\alpha_1, \cdots, \alpha_K) \tag{A.1}$$

to denote that π is a sample of a Dirichlet distribution. The mean of the Dirichlet distribution is $\mathbb{E}[\pi_k] = \alpha_k/\alpha$ where $\alpha = \sum_k \alpha_k$. The variance of any component of the Dirichlet distribution can be written as

$$\mathbb{V}[\pi_k] = \frac{\alpha_k(\alpha - \alpha_k)}{\alpha^2(\alpha + 1)}. \tag{A.2}$$

It is common to work with a constrained version of the Dirichlet distribution called the symmetric Dirichlet distribution. This is the special case when all $\alpha_1 = \cdots = \alpha_K$. We write

$$\pi \sim \mathsf{Dirichlet}(\alpha) \tag{A.3}$$

to denote π is a sample from the symmetric Dirichlet distribution. The mean of any component reduces to $\mathbb{E}[\pi_k] = 1/K$ and the variance to

$$\mathbb{V}[\pi_k] = \frac{K - 1}{K^2(K\alpha + 1)}. \tag{A.4}$$

We can now see that when α is small, samples from the Dirichlet will have high variance while with α large, samples from the Dirichlet will have small variance. In the limit $\alpha \to \infty$, samples from a Dirichlet distribution tend to a uniform distribution. This

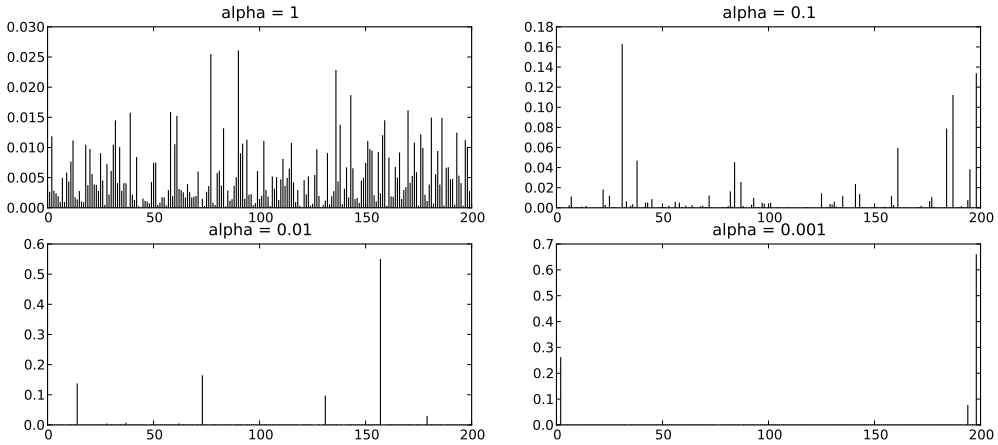


Figure A.1: Samples from a 200 dimensional Dirichlet distribution with decreasing parameter values. The x-axis shows the different dimensions of the Dirichlet samples. The y-axis shows the probability for each dimension.

behaviour is illustrated in figure A.1 for a Dirichlet with 200 dimensions: the bottom right plot has $\alpha = 0.001$ and hence a draw from this Dirichlet has only a few nonzero entries (hence high variance) while the top left plot has $\alpha = 1.0$ hence all entries of the sample have roughly the same magnitude (about 0.001).

Figure A.1 also makes clear why a Dirichlet distribution with $K \to \infty$ is nonsensical. This limit degenerates to a point mass on an arbitrary dimension.

Nemenman et al. (2001) describe a fascinating analysis of the Dirichlet distribution that applies to the work in this thesis. Let us denote by S the entropy of a random variable $q \sim \mathsf{Dirichlet}(\beta)$. S is a random variable with distribution

$$p(S|\beta) = \int dq_1 dq_2 \cdots dq_K p(q|\beta) \delta \left[S + \sum_{k=1}^{K} q_k \log q_k \right]. \tag{A.5}$$

From our argument above, we can expect the entropy of a Dirichlet draw to be high when β is large. More in particular, it is upper bounded by $\log(K)$ when β approaches infinity and the Dirichlet distribution will approach a singular distribution at the uniform discrete distribution. When β approaches 0, a draw from a Dirichlet distribution approaches a delta peak on a random entry which is a distribution with entropy 1. Getting a full handle on the distribution of S is complicated but Nemenman et al. (2001) show how to compute the mean and variance of S. Figure A.2 illustrates the distribution of the mean and variance for the Dirichlet distribution with varying β and K. As this plot illustrates, for a particular value of β, the entropy of Dirichlet draws have small variance. This implies that as you change β, the entropy of the Dirichlet draws vary smoothly, however, because the variance of the entropy is very peaked, the a priori choice of β

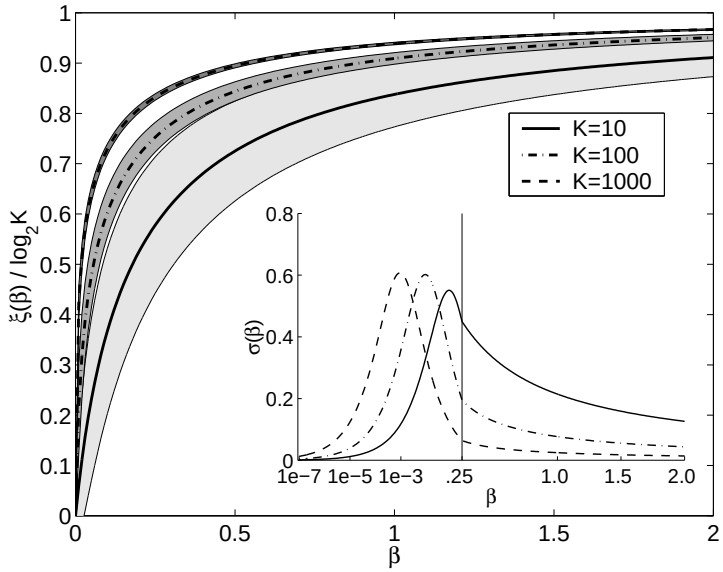


Figure A.2: The mean $\xi(\beta)$ and standard deviation $\sigma(\beta)$ of the entropy for a sample from a Dirichlet(β) distribution as a function of β. The grey bands denote one standard deviation away from the mean. Note the transition from logarithmic to linear scale at $\beta = 0.25$ in the insert. Figure reproduced from Nemenman et al. (2001).

almost completely fixes the entropy. This is problematic as it means that unless our distribution is sampled almost completely, the estimate of the entropy is dominated by the choice of our prior beta. In chapter 4 we propose one solution based on mixing over a potentially infinite set of β's using a DP mixture.

Appendix B

The Forward-Filtering Backward-Sampling Algorithm

The forward-filtering backward-sampling (FF-BS) algorithm is a Markov Chain Monte Carlo technique for sampling the state sequence $s_{1:T}$ in a hidden Markov model using dynamic programming. In contrast to Gibbs sampling, the FF-BS samples the state sequence $s_{1:T}$ in one iteration whereas the Gibbs sampling algorithm iteratively samples each s_t. For time series models, it is commonly known that this algorithm improves mixing time dramatically (Scott, 2002).

In this section we give a short overview of the algorithm. Let us denote with π the K by K transition matrix of the HMM and with $F(x|s)$ the emission likelihood. Algorithm 6 describes the sampling procedure in more detail.

Algorithm 6 The forward-filtering backward-sampling algorithm.

Initialise a K by T table to store $p(s_t = k|x_{1:t})$
for $t = 1 \dots T$ **do**
  Compute the quantity $p(s_t = k|x_{1:t}) = p(x_t|s_t = k) \sum_l \pi_{k,l} p(s_{t-1}|x_{1:t-1})$
end for
Sample $p(s_T|x_{1:T})$
for $t = T - 1 \dots 1$ **do**
  Sample the quantity $p(s_t|x_{1:T}, s_{t+1}) \propto p(s_t|x_{1:t})\pi_{s_t,s_{t+1}}$
end for

The core idea of the FF-BS algorithm is to use dynamic programming to compute the condition probability $p(s_t|x_{1:t})$ for each state s_t. When we compute $p(s_T|x_{1:t})$, note that it is conditioned on all the data. Hence, we can sample this variable where we integrated out all other state variables and conditioned on all observations. The next step is to then recursively sample the other states backwards by using Bayes rule $p(s_t|x_{1:T}, s_{t+1}) \propto p(s_t|x_{1:t})\pi_{s_t,s_{t+1}}$. The crucial observation is that we sample the whole state sequence in

one iteration. Note that the FF-BS algorithm can easily be used as an internal "block" for a blocked Gibbs sampler.

Appendix C

Markov IBP Computation

In equation (5.6) in chapter 5 we need simplified versions of combinatorial expressions. For completeness we've give the full derivation below.

$$
\begin{aligned}
\lim_{M\to\infty} \frac{M!}{M_0! M^{M_+}} &= \lim_{M\to\infty} \frac{\prod_{m=1}^{M_+}(M-m+1)}{M^{M_+}} \\
&= \lim_{M\to\infty} \left(\frac{M^{M_+} - \frac{(M_+-1)M_+}{2} M^{M_+-1} + \cdots + (-1)^{M_+-1}(M_+-1)!M}{M^{M_+}} \right) \\
&= \lim_{M\to\infty} \left(1 - O\left(\frac{c}{M}\right)\right) = 1, \qquad \text{for some constant } c \\
\lim_{M\to\infty} \prod_{i=1}^{c_m^{01}-1} (i + \frac{\alpha}{M}) &= \lim_{M\to\infty} \left((c_m^{01}-1)! + O\left(\frac{c}{M}\right)\right) = (c_m^{01}-1)!, \qquad \text{for some constant } c \\
\lim_{M\to\infty} \left(\frac{T!}{\prod_{t=1}^{T}(t+\frac{\alpha}{M})} \right)^M &= \lim_{M\to\infty} \prod_{t=1}^{T} \left(\frac{1}{1+\frac{\alpha\frac{1}{t}}{M}} \right)^M \\
&= \exp\left(-\alpha \sum_{t=1}^{T} \frac{1}{t} \right) \\
&= \exp(-\alpha H_T).
\end{aligned}
$$

Bibliography

Data, data everywhere, 2010.

M. Aitkin. Likelihood and Bayesian analysis of mixtures. *Statistical Modelling*, 1(4): 287–304, December 2001. ISSN 14770342. doi: 10.1191/147108201128212. URL http://stat.uibk.ac.at/smij/1.4-Aitkin-Abstract.php.

H. Akaike. A new look at the statistical model identification. *IEEE transactions on automatic control*, 19(6):716–723, 1974. ISSN 0018-9286. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1100705.

D.J. Aldous. Exchangeability and related topics. *Ecole dété de probabilités de Saint-Flour, XIII*, 1117:1–198, 1983. URL http://www.springerlink.com/index/c31v17440871210x.pdf.

C.E. Antoniak. Mixtures of Dirichlet processes with applications to Bayesian nonparametric problems. *The annals of statistics*, 2(6):1152–1174, November 1974. ISSN 00905364. URL http://www.jstor.org/stable/2958336.

HB Barlow. Unsupervised learning. *Neural Computation*, 3176:72–112, 1989. URL http://www.mitpressjournals.org/doi/abs/10.1162/neco.1989.1.3.295.

L.E. Baum and T. Petrie. Statistical inference for probabilistic functions of finite state Markov chains. *The Annals of Mathematical Statistics*, 37(6):1554–1563, December 1966. ISSN 00034851. URL http://www.jstor.org/stable/2238772.

Leonard E. Baum, George Soules, Norman Weiss, and Ted Petrie. A Maximization Technique Occurring in the Statistical Analysis of Probabilistic Functions of Markov Chains. *The Annals of Mathematical Statistics*, 41(1):164–171, February 1970. ISSN 00034851. URL http://www.jstor.org/stable/2239727.

M.J. Beal. *Variational algorithms for approximate Bayesian inference*. Citeseer, 2003. URL http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Variational+Algorithms+for+Approximate+Bayesian+Inference#0.

M.J. Beal and P. Krishnamurthy. Clustering gene expression time course data with countably infinite Hidden Markov Models. In *Proceedings of the Twenty-Second Conference on Uncertainty in Artificial Intelligence*, volume 16, pages 13–16, 2006. URL http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Clustering+gene+expression+time+course+data+with+countably+infinite+Hidden+Markov+Models#0.

M.J. Beal, Z. Ghahramani, and C.E. Rasmussen. The infinite hidden Markov model. *Advances in Neural Information Processing Systems*, 1:577–584, 2002. URL http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:The+infinite+hidden+Markov+model#0.

Y. Bengio and P. Frasconi. An input output HMM architecture. *Advances in neural information processing systems*, 7:427–434, 1995. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.78.1146&rep=rep1&type=pdf.

C.M. Bishop. *Pattern recognition and machine learning*. Springer New York:, 2006. URL http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Pattern+Recognition+and+Machine+Learning#0.

D. Blackwell and J.B. MacQueen. Ferguson distributions via Polya urn schemes. *The annals of statistics*, 1(2):353–355, 1973. URL http://www.jstor.org/stable/2958020.

D.M. Blei and M.I. Jordan. Variational inference for Dirichlet process mixtures. *Bayesian Analysis*, 1(1):121–144, 2006. URL http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Variational+inference+for+Dirichlet+process+mixtures#0.

S. Bratières, J. Van Gael, A. Vlachos, and Z. Ghahramani. Scaling the iHMM: Parallelization versus Hadoop. In *2010 10th IEEE International Conference on Computer and Information Technology*, pages 1235–1240. IEEE, 2010. URL http://www.computer.org/portal/web/csdl/doi/10.1109/CIT.2010.223.

E. Brill. A simple rule-based part of speech tagger. *Proceedings of the third conference on Applied Natural Language Processing Conference*, 1992. URL http://portal.acm.org/citation.cfm?id=974526.

O. Cappé, E. Moulines, and T. Rydén. Inference in hidden Markov models. 2005. URL http://books.google.com/books?hl=en&lr=&id=-3_A3_l1yssC&oi=fnd&pg=PR5&dq=Inference+in+Hidden+Markov+Models&ots=sII5yPS9Wr&sig=lEyP-AXFP-uyyRGVWBX0qhupKTY.

CERN. LHC Computing, 2010. URL http://public.web.cern.ch/Public/en/LHC/Computing-en.html.

S. Chiappa. A Bayesian Approach to Switching Linear Gaussian State-Space Models for Unsupervised Time-Series Segmentation. In *2008 Seventh International Conference on Machine Learning and Applications*, pages 3–9. IEEE Computer Society, December 2008. ISBN 978-0-7695-3495-4. doi: 10.1109/ICMLA.2008.109. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4724948.

K.W. Church. A stochastic parts program and noun phrase parser for unrestricted text. *Proceedings of the second conference on Applied natural language processing*, 1988. URL http://portal.acm.org/citation.cfm?id=974260&dl=#url.dl.

D.B. Dahl. An improved merge-split sampler for conjugate Dirichlet process mixture models. *Department of Statistics, University of Wisconsin Technical Report*, 1086, 2003. URL http://www.stat.tamu.edu/~dahl/papers/sams/tr1086.pdf.

P. F. Dahl. *Flash of the cathode rays: a history of J.J. Thomson's electron*. CRC Press, 1997. ISBN 0750304537. URL http://books.google.com/books?id=xUzaWGocMdMC&pgis=1.

H. Daume. Fast search for Dirichlet process mixture models. In *Conference on AI and Statistics*, 2007. URL http://arxiv.org/pdf/0907.1812.

A.P. Dempster, N.M. Laird, D.B. Rubin, and Others. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977. ISSN 00359246. URL http://www.ams.org/leavingmsn?url=http://links.jstor.org/sici?sici=0035-9246(1977)39:1<1:MLFIDV>2.0.CO;2-Z&origin=MSN.

P. Diaconis and D. Freedman. de Finetti's theorem for Markov chains. *The Annals of Probability*, 8(1):115–130, February 1980. ISSN 00911798. URL http://www.jstor.org/stable/2243063.

F. Doshi-Velez. The Infinite Partially Observable Markov Decision Process. In Bengio Y., D. Schuurmans, J.D. Lafferty, C.K.I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems*, volume 22, pages 477–485, 2009. URL http://people.csail.mit.edu/finale/papers/finale_ipomdp_nips09.pdf.

F. Doshi-Velez and J. Van Gael. Machine Learning for Bug Discovery, 2008. URL http://undirectedgrad.blogspot.com/2008/06/machine-learning-for-bug-discovery.html.

F. Doshi-Velez, K.T. Miller, J. Van Gael, and Y.W. Teh. Variational inference for the Indian buffet process. In *Proceedings of the Intl. Conf. on Artificial Intelligence and Statistics*, volume 12, pages 137–144. Citeseer, 2009. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.157.2795&rep=rep1&type=pdf.

D. Dunson. Multi-Task Learning for Sequential Data via iHMMs and the Nested Dirichlet Process. *Citeseer*, pages 689–696, 2007. URL http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Multi-Task+Learning+for+Sequential+Data+via+iHMMs+and+the+Nested+Dirichlet+Process#0.

M.D. Escobar. Estimating normal means with a Dirichlet process prior. *Journal of the American Statistical Association*, 1994. URL http://www.jstor.org/stable/2291223.

M.D. Escobar and M. West. Bayesian Density Estimation and Inference Using Mixtures. *Journal of the american statistical association*, 90(430):577–588, June 1995. ISSN 01621459. URL http://www.questia.com/PM.qst?a=o&se=gglsc&d=5002233859.

T.S. Ferguson. A Bayesian analysis of some nonparametric problems. *The annals of statistics*, 1(2):209–230, March 1973. ISSN 00905364. URL http://www.jstor.org/stable/2958008.

J.R. Finkel, T. Grenager, and C.D. Manning. The infinite tree. *Meeting of the Association for Computational Linguistics*, 2007. URL http://acl.ldc.upenn.edu/P/P07/P07-1035.pdf.

E.B. Fox, E.B. Sudderth, M.I. Jordan, and A.S. Willsky. An HDP-HMM for systems with state persistence. In *Proceedings of the 25th international conference on Machine learning*, volume 25, pages 312–319, Helsinki, 2008a. ACM. URL http://portal.acm.org/citation.cfm?id=1390196.

E.B. Fox, E.B. Sudderth, M.I. Jordan, and A.S. Willsky. An HDP-HMM for systems with state persistence. In *Proceedings of the 25th international conference on Machine learning*, volume 25, pages 312–319, Helsinki, 2008b. ACM. URL http://portal.acm.org/citation.cfm?id=1390196.

E.B. Fox, E.B. Sudderth, M.I. Jordan, and A.S. Willsky. Nonparametric Bayesian Learning of Switching Linear Dynamical Systems. In *Neural Information Processing Systems 21*, volume 21. MIT Press, 2009a. URL http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Nonparametric+Bayesian+Learning+of+Switching+Linear+Dynamical+Systems#0.

E.B. Fox, E.B. Sudderth, M.I. Jordan, and A.S. Willsky. The Sticky HDP-HMM: Bayesian Nonparametric Hidden Markov Models with Persistent States. *Arxiv preprint arXiv:0905.2592*, 2009b. URL http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:The+Sticky+HDP-HMM:+Bayesian+Nonparametric+Hidden+Markov+Models+with+Persistent+States#0.

S. Fruhwirth-Schnatter. Estimating marginal likelihoods for mixture and Markov switching models using bridge sampling techniques. *Econometrics Journal*, 7(1):143–167, 2004.

J. Gao and M. Johnson. A comparison of Bayesian estimators for unsupervised Hidden Markov Model POS taggers. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 344–352. Association for Computational Linguistics, 2008. URL http://portal.acm.org/citation.cfm?id=1613761.

J. Gao, G. Andrew, M. Johnson, and K. Toutanova. A comparative study of parameter estimation methods for statistical natural language processing. In *Meeting of the Association for Computational Linguistics*, volume 45, page 824, 2007. URL http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:A+comparative+study+of+parameter+estimation+methods+for+statistical+natural+language+processing#0.

A. Gelman, J.B. Carlin, H.S. Stern, and D.B. Rubin. Bayesian data analysis. *London, Glasgow, et al*, page 668, 1995. URL http://aje.oxfordjournals.org/cgi/reprint/146/4/367.pdf.

Z. Ghahramani. Factorial learning and the EM algorithm. In *Advances in neural information processing systems*, number Mdl, pages 617–624. Citeseer, 1995. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.13.7118&rep=rep1&type=pdf.

Z. Ghahramani and G.E. Hinton. Variational learning for switching state-space models. *Neural Computation*, 12(4):831–864, April 2000. URL http://www.mitpressjournals.org/doi/abs/10.1162/089976600300015619.

Z. Ghahramani and M.I. Jordan. Factorial hidden Markov models. *Machine learning*, 29 (2):245–273, 1997. URL http://www.springerlink.com/index/W3523227075K34T4.pdf.

S. Goldwater and T. Griffiths. A fully Bayesian approach to unsupervised part-of-speech tagging. In *ANNUAL MEETING-ASSOCIATION FOR COMPUTATIONAL LINGUISTICS*, volume 45, page 744, 2007. URL http://acl.ldc.upenn.edu/P/P07/P07-0094.pdf.

S. Goldwater, T. Griffiths, and M. Johnson. Interpolating between types and tokens by estimating power-law generators. *Advances in Neural Information Processing Systems*, 18:459, 2006. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.70.9842&rep=rep1&type=pdf.

D. Görür, F. Jäkel, and C.E. Rasmussen. A choice model with infinitely many latent features. In *Proceedings of the 23rd international conference on Machine learning - ICML '06*, pages 361–368, New York, New York, USA, 2006. ACM Press. ISBN 1595933832. doi: 10.1145/1143844.1143890. URL http://portal.acm.org/citation.cfm?doid=1143844.1143890.

P.J. Green. Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika*, 82(4):711, 1995. URL http://biomet.oxfordjournals.org/cgi/content/abstract/82/4/711.

T. Griffiths and Z. Ghahramani. Infinite latent feature models and the Indian buffet process. *Advances in Neural Information Processing Systems*, 18:475, 2006. URL http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Infinite+latent+feature+models+and+the+Indian+buffet+process#0.

A. Halevy, P. Norvig, and F. Pereira. The unreasonable effectiveness of data. *IEEE Intelligent Systems*, 24(2):8–12, 2009. ISSN 1541-1672. doi: http://doi.ieeecomputersociety.org/10.1109/MIS.2009.36. URL http://portal.acm.org/citation.cfm?id=1525642.1525689.

T. Hastie, R. Tibshirani, and J. H. Friedman. *The Elements of Statistical Learning*. Springer, 2003. ISBN 0387952845. URL http://www.amazon.com/Elements-Statistical-Learning-T-Hastie/dp/0387952845.

G.E. Hinton and R.S. Zemel. Autoencoders, Minimum Description Length and Helmholtz Free Energy. *Advances in Neural Information Processing Systems 6*, 114(9):3–10, 1994. ISSN 15205207. doi: 10.1021/jp906511z. URL http://eprints.kfupm.edu.sa/27105.

N.L. Hjort, C. Holmes, P. Müller, and S.G. Walker, editors. *Bayesian Nonparametrics*. Cambridge University Press, Cambridge, UK, 1 edition, 2010. ISBN 9780521513463. URL http://www.cambridge.org/catalogue/catalogue.asp?isbn=9780521513463.

A. Hyvärinen and E. Oja. Independent component analysis: algorithms and applications. *Neural networks*, 13(4-5):411–430, 2000. URL http://linkinghub.elsevier.com/retrieve/pii/S0893608000000265.

H. Ishwaran and L.F. James. Gibbs Sampling Methods for Stick-Breaking Priors. *Journal of the American Statistical Association*, 96(453):161–173, March 2001. ISSN 01621459. URL http://www.questia.com/PM.qst?a=o&se=gglsc&d=5002393290.

S. Jain and R.M. Neal. A split-merge Markov chain Monte Carlo procedure for the Dirichlet process mixture model. *Journal of Computational and Graphical Statistics*, 13(1): 158–182, 2004. URL http://pubs.amstat.org/doi/abs/10.1198/1061860043001.

L.F. James, A. Lijoi, and I. Prünster. Conjugacy as a distinctive feature of the Dirichlet process. *Scandinavian Journal of Statistics*, 33(1):105–120, 2006. URL http://www3.interscience.wiley.com/journal/118610760/abstract.

M. Johnson. Why doesnt EM find good HMM POS-taggers. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 296–305, 2007. URL http://acl.ldc.upenn.edu/D/D02/D02-1031.pdf.

M. Johnson, P. Blunsom, T. Cohn, and S. Goldwater. A Note on the Implementation of Hierarchical Dirichlet Processes. Singapore, 2009.

D. Jurafsky and J.H. Martin. *Speech and language processing*. Prentice Hall New York, 2000. ISBN 0131873210. URL http://www.cs.colorado.edu/~martin/SLP/Updates/1.pdf.

M. Kalli, S.G. Walker, and J.E. Griffin. Slice Sampling the Mixture of Dirichlet Process Model: A Comparative Study and New Ideas. Technical report, University of Kent, 2008.

J.F.C. Kingman. *Poisson processes*. Oxford University Press, USA, 1993. ISBN 0585200084. URL http://books.google.com/books?hl=en&lr=&id=VEiM-OtwDHkC&oi=fnd&pg=PA1&dq=Poisson+Processes&ots=zUEzPWME0O&sig=8ZCjVGJ1PKOaftH9bnM4mTF9NLo.

D. Knowles and Z. Ghahramani. Infinite sparse factor analysis and infinite independent components analysis. *Lecture Notes in Computer Science*, 4666:381, 2007. URL http://www.springerlink.com/index/h3w36r4587256416.pdf.

J. Kupiec. Robust part-of-speech tagging using a hidden Markov model. *Computer speech & language*, 6(3):225–242, 1992. URL http://linkinghub.elsevier.com/retrieve/pii/088523089290019Z.

K. Kurihara, M. Welling, and Y.W. Teh. Collapsed variational Dirichlet process mixture models. In *Proceedings of the International Joint*

Conference on Artificial Intelligence, volume 20, page 19, 2007a. URL http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Collapsed+Variational+Dirichlet+Process+Mixture+Models#0.

K. Kurihara, M. Welling, and N. Vlassis. Accelerated variational dirichlet process mixtures. *Advances in Neural Information Processing Systems*, 19:761, 2007b. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.73.1206&rep=rep1&type=pdf.

P. Liang, S. Petrov, M. Jordan, and D. Klein. The infinite PCFG using hierarchical Dirichlet processes. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 688–697, 2007. URL http://acl.ldc.upenn.edu/D/D07/D07-1072.pdf.

S.N. MacEachern. Estimating normal means with a conjugate style Dirichlet process prior. *Communications in Statistics - Simulation and Computation*, 1994. URL http://www.informaworld.com/index/780079038.pdf.

S.N. MacEachern and P. Müller. Estimating mixture of Dirichlet process models. *Journal of Computational and Graphical Statistics*, 1998. URL http://www.jstor.org/stable/1390815.

D.J.C. MacKay. Ensemble learning for hidden Markov models. Technical report, University of Cambridge, 1997. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.52.9627&rep=rep1&type=pdf.

M.P. Marcus, B. Santorini, and M.A. Marcinkiewicz. Building a large annotated corpus of English: The Penn Treebank. *Computational linguistics*, 19(2):313–330, June 1994. ISSN 0891-2017. URL http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Building+a+large+annotated+corpus+of+English:+the+Penn+treebank#0.

J.D. McAuliffe, D.M. Blei, and M.I. Jordan. Nonparametric empirical Bayes for the Dirichlet process mixture model. *Statistics and Computing*, 16(1):5–14, 2006. URL http://www.springerlink.com/index/T4Q7X4V353V4N264.pdf.

E. Meeds, Z. Ghahramani, R.M. Neal, and S.T. Roweis. Modeling dyadic data with binary latent factors. *Advances in Neural Information Processing Systems*, 19:977, 2007. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.70.477&rep=rep1&type=pdf.

M. Meila. Comparing clusterings–an information based distance. *Journal of Multivariate Analysis*, 2007. URL http://linkinghub.elsevier.com/retrieve/pii/S0047259X06002016.

B. Merialdo. Tagging English text with a probabilistic model. *Computational linguistics*, 1994. URL http://portal.acm.org/citation.cfm?id=972525.972526.

K.T. Miller, T.L. Griffiths, and M.I. Jordan. The Phylogenetic Indian buffet Process: A Non-Exchangeable Nonparametric Prior for Latent Features. In *Uncertainty in Artificial Intelligence*, Helsinki, 2008a.

K.T. Miller, T.L. Griffiths, and M.I. Jordan. Variations on Non-Exchangeable Nonparametric Priors for Latent Feature Models. In *Nonparametric Bayes Workshop at ICML/UAI*. Berkeley, CA, USA, 2008b.

T.P. Minka and Z. Ghahramani. Expectation propagation for infinite mixtures. In *NIPS Workshop on Nonparametric Bayesian Methods and Infinite Models*, volume 19. Citeseer, 2003. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.66.5681&rep=rep1&type=pdf.

P. Müller and F.A. Quintana. Nonparametric Bayesian data analysis. *Statistical science*, 19(1):95–110, 2004. URL http://www.jstor.org/stable/4144375.

R.M. Neal. Bayesian mixture modeling. In *Maximum Entropy and Bayesian Methods: Proceedings of the 11th International Workshop on Maximum Entropy and Bayesian Methods of Statistical Analysis*, pages 197–211, 1991. URL http://books.google.com/books?hl=en&lr=&id=WmkHBee7FyYC&oi=fnd&pg=PA197&dq=Bayesian+Mixture+Modeling&ots=tEVJKUbMaJ&sig=WDYlku6ReLajh7Hnp3FbvTPMN8I.

R.M. Neal. Markov chain sampling methods for Dirichlet process mixture models. *Journal of computational and graphical statistics*, 9(2):249–265, 2000. URL http://www.jstor.org/stable/1390653.

R.M. Neal. Annealed importance sampling. *Statistics and Computing*, 11(2):125–139, 2001. URL http://www.springerlink.com/index/X40W6W4R1142651K.pdf.

R.M. Neal. Slice sampling. *The annals of statistics*, 31(3):705–741, 2003. URL http://www.jstor.org/stable/3448413.

R.M. Neal, M.J. Beal, and S.T. Roweis. Inferring state sequences for non-linear systems with embedded hidden Markov models. In *Advances in Neural Information Processing Systems 16: Proceedings of the 2003 Conference*, volume 16, page 401. The

MIT Press, 2004. URL http://books.google.com/books?hl=en&lr=&id=0F-9C7K8fQ8C&oi=fnd&pg=PA401&dq=Inferring+State+Sequences+for+Non-linear+Systems+with+Embedded+Hidden+Markov+Models&ots=TFJrlVVc85&sig=yJ1tkk0PXbb5sPLJ2lXoGR_dJ9g.

I. Nemenman, F. Shafee, and W. Bialek. Entropy and inference, revisited. *Arxiv preprint physics/0108025*, 2001. URL http://arxiv.org/abs/physics/0108025.

J. Paisley and L. Carin. Hidden Markov Models with Stick Breaking Priors. *submitted to IEEE Trans. on Signal Processing*, 2008. URL http://people.ee.duke.edu/~jwp4/Papers/SB-HMM.pdf.

O. Papaspiliopoulos, G. O. Roberts, and G.O. Roberts. Retrospective Markov chain Monte Carlo methods for Dirichlet process hierarchical models. *Biometrika*, 95(1):169–186, 2008. URL http://biomet.oxfordjournals.org/cgi/reprint/asm086v1.pdf.

U. Paquet. *Bayesian Inference for Latent Variable Models*. PhD thesis, University of Cambridge, 2007.

J. Pitman. *Combinatorial stochastic processes*, volume 1875 of *Lecture Notes in Mathematics*. Springer-Verlag, Berlin, 2006. ISBN 978-3-540-30990-1; 3-540-30990-X. doi: 10.1007/b11601500. URL http://bibserver.berkeley.edu/csp/april05/bookcsp.pdf.

J. Pitman and M. Yor. The two-parameter Poisson-Dirichlet distribution derived from a stable subordinator. *The Annals of Probability*, 25(2):855–900, 1997. URL http://www.projecteuclid.org/GetRecord?id=euclid.aop/1024404422.

I. Porteous, A. Ihler, P. Smyth, and M. Welling. Gibbs sampling for (coupled) infinite mixture models in the stick-breaking representation. In *Proceedings of UAI*, volume 22, 2006. URL http://www.datalab.uci.edu/papers/ddp_uai06_v8.pdf.

I. Pruteanu-Malinici and L. Carin. Infinite hidden Markov models for unusual-event detection in video. *IEEE transactions on image processing: a publication of the IEEE Signal Processing Society*, 17(5):811, May 2008. ISSN 1057-7149. doi: 10.1109/TIP.2008.919359. URL http://www.ncbi.nlm.nih.gov/pubmed/18390385.

L. R. Rabiner. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989. URL http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=18626.

C.E. Rasmussen. The infinite Gaussian mixture model. *Advances in neural information processing systems*, 12:554–560, 2000. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.111.993&rep=rep1&type=pdf.

C. P. Robert, T. Ryden, and D. M. Titterington. Bayesian inference in hidden Markov models through the reversible jump Markov chain Monte Carlo method. *Journal of the Royal Statistical Society. Series B, Statistical Methodology*, pages 57–75, 2000.

A. Rosenberg and J. Hirschberg. V-measure: A conditional entropy-based external cluster evaluation measure. *EMNLP'07*, 2007. URL http://acl.ldc.upenn.edu/D/D07/D07-1043.pdf.

J.J.K.Ó. Ruanaidh and W.J. Fitzgerald. *Numerical Bayesian methods applied to signal processing*. Springer Verlag, February 1996. ISBN 0387946292. URL http://books.google.com/books?hl=en&lr=&id=74ky3bnm2gAC&oi=fnd&pg=PR6&dq=Numerical+Bayesian+Methods+Applied+to+Signal+Processing&ots=i805RE0isY&sig=LFOXN1u80kmLq_aV6enaQztybOQ.

T. Ryden. EM versus Markov chain Monte Carlo for estimation of hidden Markov models: a computational perspective. *Bayesian Analysis*, 3(4):659–688, 2008.

E.F.T.K. Sang and S. Buchholz. Introduction to the CoNLL-2000 shared task: Chunking. *Computing Research Repository*, 2000. URL http://acl.ldc.upenn.edu/W/W00/W00-0726.pdf.

G. Schwarz. Estimating the dimension of a model. *The annals of statistics*, 6(2):461–464, March 1978. ISSN 00905364. URL http://www.jstor.org/stable/2958889.

S.L. Scott. Bayesian Methods for Hidden Markov Models: Recursive Computing in the 21st Century. *Journal of the American Statistical Association*, 97(457):337–351, March 2002. ISSN 01621459. URL http://www.jstor.org/stable/3085787.

J. Sethuraman. A constructive definition of Dirichlet priors. *Statistica Sinica*, 4(2):639–650, 1994. URL http://www.yaroslavvb.com/papers/sethuraman-constructive.pdf.

F. Sha and F. Pereira. Shallow parsing with conditional random fields. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 134–141. Association for Computational Linguistics, 2003. URL http://portal.acm.org/citation.cfm?id=1073473&dl=.

K. Sohn and E.P. Xing. Hidden Markov Dirichlet process: modeling genetic recombination in open ancestral space. In *Advances in Neural Information Processing Systems*, volume 19, page 1305. Citeseer, 2007. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.72.9335&rep=rep1&type=pdf.

T. Stepleton, Z. Ghahramani, G. Gordon, and T.S. Lee. The block diagonal infinite hidden markov model. In *AISTATS*, volume 5, pages 552–559. Citeseer, 2009. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.150.3702&rep=rep1&type=pdf.

A. Stolcke and S. Omohundro. Hidden Markov model induction by Bayesian model merging. *Advances in neural information processing systems*, 5:11–11, 1993. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.43.470&rep=rep1&type=pdf.

Y.W. Teh. A hierarchical Bayesian language model based on Pitman-Yor processes. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, page 992. Association for Computational Linguistics, 2006. URL http://portal.acm.org/citation.cfm?id=1220299.

Y.W. Teh. *Dirichlet Processes*. Springer, 2010.

Y.W. Teh, M.I. Jordan, M.J. Beal, and D.M. Blei. Hierarchical dirichlet processes. *Journal of the American Statistical Association*, 101(476):1566–1581, 2006a. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.61.4501&rep=rep1&type=pdf.

Y.W. Teh, M.I. Jordan, M.J. Beal, and D.M. Blei. Hierarchical dirichlet processes. *Journal of the American Statistical Association*, 101(476):1566–1581, 2006b. URL http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Hierarchical+Dirichlet+processes#0.

Y.W. Teh, D. Görür, and Z. Ghahramani. Stick-breaking construction for the Indian buffet process. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, volume 11. Citeseer, 2007. URL http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Stick-breaking+construction+for+the+Indian+buffet+process#0.

Y.W. Teh, K. Kurihara, and M. Welling. Collapsed variational inference for HDP. *Advances in Neural Information Processing Systems*, 20:1481–1488, 2008. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.96.1165&rep=rep1&type=pdf.

R. Thibaux and M.I. Jordan. Hierarchical beta processes and the indian buffet process. In *International Conference on Artificial Intelligence and Statistics*. Citeseer, 2007. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.69.2569&rep=rep1&type=pdf.

M. Titsias. The infinite gamma-poisson feature model. In *International Conference on Neural Information Processing Systems*. Citeseer, 2007. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.68.6027&rep=rep1&type=pdf.

J. Van Gael. The infinite hidden markov model, 2010. http://mloss.org/software/view/205/.

J. Van Gael, Y. Saatci, Y.W. Teh, and Z. Ghahramani. Beam sampling for the infinite hidden markov model. In *Proceedings of the 25th international conference on Machine learning*, volume 25, pages 1088–1095, Helsinki, 2008a. ACM. URL http://portal.acm.org/citation.cfm?id=1390156.1390293.

J. Van Gael, Y.W. Teh, and Z. Ghahramani. The infinite factorial hidden Markov model. In D. Koller, D. Schuurmans, L. Bottou, and Y. Bengio, editors, *Advances in Neural Information Processing Systems*, volume 21, pages 1697–1704. MIT Press, 2008b. URL http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:The+infinite+factorial+hidden+Markov+model#0.

J. Van Gael, A. Vlachos, and Z. Ghahramani. The infinite HMM for unsupervised PoS tagging. *Empirical Methods in Natural Language Processing*, 2009. URL http://140.116.245.248/ACL-IJCNLP-2009/EMNLP/pdf/EMNLP071.pdf.

A. Vlachos, A. Korhonen, and Z. Ghahramani. Unsupervised and constrained dirichlet process mixture models for verb clustering. In *Proceedings of the workshop on geometrical models of natural language semantics*, pages 74–82. Association for Computational Linguistics, 2009. URL http://portal.acm.org/citation.cfm?id=1705425.

S.G. Walker. Sampling the Dirichlet mixture model with slices. *Communications in Statistics-Simulation and Computation*, 36(1-3):45–54, 2007. ISSN 0361-0918. doi: 10.1080/03610910601096262. URL http://www.icer.it/docs/wp2006/ICERwp16-06.pdf.

L. Wasserman. *All of nonparametric statistics*. Springer, 2006. URL http://books.google.co.uk/books?hl=en&lr=&id=neHDoLq9jycC&oi=fnd&pg=PA1&dq="all+of+nonparametric+statistics"+wasserman&ots=gfIfiLcZpk&sig=oZZwtdhAaZh6dX0XRzTDGwbIcnE.

R. Weischedel, R. Schwartz, J. Palmucci, M. Meteer, and L. Ramshaw. Coping with ambiguity and unknown words through probabilistic models. *Computational Linguistics*, 19(2):361–382, 1993. ISSN 0891-2017. URL http://portal.acm.org/citation.cfm?id=972477.

F. Wood and T.L. Griffiths. Particle filtering for nonparametric Bayesian matrix factorization. *Advances in Neural Information Processing Systems*, 19:1513, 2007. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.71.1901&rep=rep1&type=pdf.

F. Wood, T.L. Griffiths, and Z. Ghahramani. A non-parametric Bayesian method for inferring hidden causes. In *Conference on Uncertainty in Artificial Intelligence*, 2006. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.86.5769&rep=rep1&type=pdf.

Y. Xu, K.A. Heller, and Z. Ghahramani. Tree-based inference for dirichlet process mixtures. In *Proceedings of the 12th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 16–18. Citeseer, 2009. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.160.7101&rep=rep1&type=pdf.

X.X. Zhang, H. Liu, Y. Gao, and D.H. Hu. Detecting Abnormal Events via Hierarchical Dirichlet Processes. In *Proceedings of the 13th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining*, page 289. Springer, 2009. URL http://www.springerlink.com/index/1g35626p34l71114.pdf.

O. Zobay. Mean field inference for the Dirichlet process mixture model. *Electronic Journal of Statistics*, 3:507–545, 2009. URL http://www.projecteuclid.org/DPubS/Repository/1.0/Disseminate?handle=euclid.ejs/1244726600&view=body&content-type=pdfview_1.