Exact slice sampler for HDPs

Anonymous Author(s)
Affiliation
Address
email

Abstract

We propose an exact slice sampler for Hierarchical Dirichlet process (HDP) and its associated mixture models [9]. Although there are existing MCMC algorithms for sampling the HDP, a slice sampler for HDP has been missing from the literature. Slice sampling is well-known for its desirable properties including its fast mixing time and its natural potential for parallelization. On the other hand, the hierarchical nature of HDPs poses challenges to adopting a full-fledged slice sampler that automatically truncates all the infinite measures involved without ad-hoc modifications. In this work, we adopt the powerful idea of Bayesian variable augmentation to address this challenge. By introducing new latent variables, we obtain a full factorization of the joint distribution that is suitable for slice sampling. Our algorithm has several appealing features such as (1) fast mixing; (2) remaining exact while allowing natural truncation of the underlying infinite-dimensional measures, as in [3], resulting in updates of only a finite number of necessary atoms and weights in each iteration; and (3) being naturally suited to parallel implementations. The underlying principle for joint factorization of the full likelihood is simple and can be applied to many other settings such as designing sampling algorithms for general DDP (dependent Dirichlet process) models.

1 Introduction

Hierarchical Dirichlet process (HDP)[9] is one of the popular Bayesian nonparametric models for modeling the hierarchy of groups of data. It has been widely applied in various learning tasks in statistics and machine learning such as in topic modeling [9, 6] and information retrieval [9]. It is often used as the mixing measure in a mixture model for modeling the cluster structure for groups of data [8]. Other applications include using HDP for modeling the transition probabilities between hidden states in a hidden Markov model[2]. Being able to sample efficiently from a HDP is crucial for making inference in HDP related models, and to do so, both approximating algorithms and sampling-based algorithms have been proposed in the literature. Approximating algorithms for HDP are mostly based on variational approaches. See for example [5], [7] and [6]. These algorithms can often scale to large data sets but suffer from some obvious draw backs: the variational posterior tends to underestimate the variability of the true posterior.

In this work we will focus on the sampling based methods. One of the prominent methods is the Chinese Restaurant Franchise (CRF)-based Gibbs sampler [9]. This algorithm, however, is known to mix slowly, therefore encouraging the search for more efficient sampling based approaches. Attempting that, [2] adopted a truncated approximation to the full posterior distribution for sampling HDP in a hidden Markov model, and [10] proposed a split/merge MCMC algorithm, which, however, according to [1] only shows a marginal improvement over the [9] sampler. More recently, [1] proposed an algorithm that also considers split/merge steps but has the advantage of allowing parallel sampling, which exhibited significantly improved convergence when compared to [10]. [4] proposes a online algorithms based on mini-batch ideas.

Submitted to 32nd Conference on Neural Information Processing Systems (NIPS 2018). Do not distribute.

One efficient approach to sample from DP mixtures is through slice sampling. This approach has the advantage of allowing natural truncation of the infinite-dimensional Dirichlet measures, as shown in the seminal work of [3], and is known to have great mixing properties. To our best knowledge, however, no slice samplers have been proposed for HDPs . We fill this gap with the development of an *exact slice sampler* for HDP, extending the ideas of the slice sampling for DP to the hierarchical setup. One difficulty in performing slice sampling for HDP arises from the hierarchical nature of the HDP model (cf. Section 2.1 for more details on the difficulty facing the hierarchical stick-breaking representation and the marginalization approach). To circumvent this, we adopt the powerful idea of Bayesian variable augmentation which allows a full factorization of the joint distribution. The slice sampler that we derive for HDPs retains the same advantage of being exact while truncating the underlying infinite-dimensional measures ($\boldsymbol{\beta}$ and $\boldsymbol{\gamma}_j$ in our notation), i.e., the sampler only updates the necessary atoms and weights in each iteration, whose number is guaranteed to be finite.

The slice sampler proposed here enjoys several key features: it is very simple and its updates are fairly intuitive. It is also naturally parallel. We provide a natural order of updates which is in general not trivial. Moreover, as a by-product, our approach for a complete factorization of the joint density for HDP could be of independent interest. The expression for the joint density allows one to easily verify the validity of the slice sampler updates; i.e., not much knowledge of Dirichlet processes and their intricacies is required, and the derivations are accessible to most practioners with a basic understanding of Bayesian statistics.

Our motivation for this new algorithm came from a network point of view, and the need to propose inference models for multiplex networks that can take into account potential dependency across different layers, particularly when the aim is community detection. In this work (that is currently in progress), we specify HDP as a natural random partition prior for the partitions across different layers in the multiplex network. Despite our original motivation, it is important to point out that our algorithm can be easily generalized to other models if one replaces the mixture part with a general likelihood, therefore immediately getting a nicer sampling scheme.

2 HDP setup

2.1 Label-only HDP

We start by deriving a representation of the HDPs which is suitable for slice sampling. We assume familiarity with the setup of a HDP as in [9] and the associated metaphor of a Chinese restaurant franchise (CRF). Consider the HDP as defined in [9, Equ. (19)], using the same notation (except for minor tweaks) and the CRF metaphor,

$$\begin{aligned} \boldsymbol{\beta} \mid \gamma_0 &\sim \mathrm{GEM}(\gamma_0) \\ \boldsymbol{\pi}_j \mid \alpha_o, \boldsymbol{\beta} &\sim \mathrm{DP}(\alpha_0, \boldsymbol{\beta}), \quad z_{ji} \mid \boldsymbol{\pi}_j \sim \boldsymbol{\pi}_j \end{aligned}$$

where $i = 1, \ldots, n_j$ is the customer index and $j = 1, \ldots, J$ is the restaurant index. Note that we are ignoring the downstream mixture model for the moment, since this is the main part of the sampling problem. Thus, we assume that we observe the labels $\{z_{ji}\}$ where z_{ji} is the dish of customer i in restaurant j, and we want to estimate $\boldsymbol{\beta}$ and $\{\boldsymbol{\pi}_j\}$.

The stick-breaking representation of $\boldsymbol{\pi}_j$—see Equ. (21) in [9]—is not suitable for slice (or Gibbs) sampling due to the complicated dependence on $\boldsymbol{\beta}$. Another idea is to marginalized $\boldsymbol{\pi}_j$, but that would lead to distributions with ratios of Gamma functions as densities which are not easy to sample from. Instead, we just use the fact that $\boldsymbol{\pi}_j$ is itself a Dirichlet measure. That is, we can write the model as

$$\begin{aligned} \boldsymbol{\beta} \mid \gamma_0 &\sim \mathrm{GEM}(\gamma_0) \\ \boldsymbol{\gamma}_j \mid \alpha_0 &\sim \mathrm{GEM}(\alpha_0), \quad \boldsymbol{\gamma}_j = (\gamma_{jt}) \\ k_{jt} \mid \boldsymbol{\beta} &\sim \boldsymbol{\beta}, \quad t \in \mathbb{N} \\ \boldsymbol{\pi}_j &= \sum_{t=1}^{\infty} \gamma_{jt} \delta_{k_{jt}}, \quad z_{ji} \mid \boldsymbol{\pi}_j \sim \boldsymbol{\pi}_j. \end{aligned}$$

Note that $\boldsymbol{\gamma}_j$ and (k_{jt}) are independently drawn. Here, index "t" is interpreted as indexing the tables; k_{jt} is the dish (type) of table t in restaurant j, while γ_{jt} represents the fraction of the customers in restaurant j that would sit at table t (eventually).

This representation is still not suitable for sampling. Instead of sampling directly from $\boldsymbol{\pi}_j$, we sample from the weights $\boldsymbol{\gamma}_j$ first, i.e., we pick the table of customer i and then assign them the dish of the table. This gives us access to the last missing piece which is t_{ji}, the table of customer i in restaurant j. We can write the model equivalently as

$$\begin{aligned} \boldsymbol{\beta} \mid \gamma_0 &\sim \mathrm{GEM}(\gamma_0), \quad \boldsymbol{\beta} = (\beta_k) \\ \boldsymbol{\gamma}_j \mid \alpha_0 &\sim \mathrm{GEM}(\alpha_0), \quad \boldsymbol{\gamma}_j = (\gamma_{jt}) \\ k_{jt} \mid \boldsymbol{\beta} &\sim \boldsymbol{\beta}, \quad t \in \mathbb{N}, \quad t_{ji} \mid \boldsymbol{\gamma}_j \sim \boldsymbol{\gamma}_j, \quad i = 1, \ldots, n_j \\ z_{ji} \mid \boldsymbol{t}_j, \boldsymbol{k}_j &= k_{j,t_{ji}} \end{aligned} \tag{1}$$

where $\boldsymbol{k}_j = (k_{jt}, t \in \mathbb{N})$ is the collection of all the dishes at restaurant j. Note that we sample t_{ji} (which table to sit customer i in restaurant j) from the eventual distributions of customers among tables in restaurant j, i.e., $\boldsymbol{\gamma}_j$. The equation $z_{ji} = k_{j,t_{ji}}$ means that the dish of customer i in restaurant j, i.e. z_{ji}, is completely determined by looking at which table they are sitting at, t_{ji}, and what dish is presented at that table $k_{j,t_{ji}}$. Since given everything else, z_{ji} is deterministic, we only need to worry about sampling $\boldsymbol{\beta}$, $\boldsymbol{\gamma}_j$, $\boldsymbol{k}_j$ and $\boldsymbol{t}_j = (t_{ji})$. Let us define $F : [0,1]^{\mathbb{N}} \to [0,1]^{\mathbb{N}}$ by

$$[F(\boldsymbol{x})]_1 := x_1, \quad [F(\boldsymbol{x})]_j := x_j \prod_{\ell=1}^{j-1} (1 - x_\ell) \tag{2}$$

where $\boldsymbol{x} = (x_j, j \in \mathbb{N})$. We note that both $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}_j$ have stick-breaking representations:

$$\begin{aligned} \gamma'_{jt} &\sim \mathrm{Beta}(1, \alpha_0), \quad \beta'_k \sim \mathrm{Beta}(1, \gamma_0), \\ \boldsymbol{\gamma}_j &= F(\boldsymbol{\gamma}'_j), \qquad\qquad \boldsymbol{\beta} = F(\boldsymbol{\beta}'), \end{aligned}$$

where $\boldsymbol{\gamma}'_j = (\gamma'_{jt})$ and $\boldsymbol{\beta}' = (\beta'_k)$. Let us write $x \mapsto b_{\alpha_0}(x)$ for the density of $\mathrm{Beta}(1, \alpha_0)$.

Note that $\mathbb{P}(t_{ji} = t) = \gamma_{jt},\ t \in \mathbb{N}$. Thus, we can write down the joint as

$$\begin{aligned} p(\boldsymbol{t}, \boldsymbol{k}, \boldsymbol{\gamma}', \boldsymbol{\beta}') &= \prod_{j=1}^{J} \Big[p(\boldsymbol{t}_j | \boldsymbol{\gamma}_j)\, p(\boldsymbol{\gamma}'_j)\, p(\boldsymbol{k}_j | \boldsymbol{\beta}) \Big] p(\boldsymbol{\beta}') \\ &= \prod_{j=1}^{J} \left(\prod_{i=1}^{n_j} \gamma_{j,t_{ji}} \prod_{t=1}^{\infty} b_{\alpha_0}(\gamma'_{jt}) \prod_{t=1}^{\infty} \beta_{k_{jt}} \right) \prod_{k=1}^{\infty} b_{\gamma_0}(\beta'_k). \end{aligned} \tag{3}$$

Interestingly, this decomposition works for any other stick-breaking distributions on $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}_j$. Using $\boldsymbol{\gamma}_j = F(\boldsymbol{\gamma}'_j)$ and $\boldsymbol{\beta} = F(\boldsymbol{\beta}')$, a more explicit formula is

$$p(\boldsymbol{t}, \boldsymbol{k}, \boldsymbol{\gamma}', \boldsymbol{\beta}') = \prod_{j=1}^{J} \left(\prod_{i=1}^{n_j} [F(\boldsymbol{\gamma}'_j)]_{t_{ji}} \prod_{t=1}^{\infty} b_{\alpha_0}(\gamma'_{jt}) \prod_{t=1}^{\infty} [F(\boldsymbol{\beta}')]_{k_{jt}} \right) \prod_{k=1}^{\infty} b_{\gamma_0}(\beta'_k) \tag{4}$$

which gives the complete joint density of of HDP in (1).

2.2 Mixture part

Finally, we can add in the mixture component as

$$\begin{aligned} f_k \mid \mathscr{F} &\sim \mathscr{F}, \quad \boldsymbol{f} = (f_k) \\ y_{ji} \mid z_{ji}, \boldsymbol{f} &\sim f_{z_{ji}}, \end{aligned} \tag{5}$$

where $\boldsymbol{f}$ is an infinite collection of possible mixture components, where each coordinate f_k is a density drawn from a distribution $\mathscr{F}$ on densities. We can assume $f_k = K(\cdot, \phi_k)$ for some kernel K, where $\phi_k \mid H \sim H$ to get back the more common parametric mixture model; the more general setup however is easier to work with conceptually. Since $p(\boldsymbol{y} | \boldsymbol{t}, \boldsymbol{k}, \boldsymbol{f}) = \prod_{j=1}^{J} \prod_{i=1}^{n_j} f_{z_{ji}}(y_{ji})$, the overall joint is

$$\begin{aligned} p(\boldsymbol{y}, \boldsymbol{f}, \boldsymbol{t}, \boldsymbol{k}, \boldsymbol{\gamma}', \boldsymbol{\beta}') &= p(\boldsymbol{y} \mid \boldsymbol{t}, \boldsymbol{k}, \boldsymbol{f})\, p(\boldsymbol{t}, \boldsymbol{k}, \boldsymbol{\gamma}', \boldsymbol{\beta}')\, p(\boldsymbol{f}) \\ &= \prod_{j=1}^{J} \left(\prod_{i=1}^{n_j} \left[f_{k_{j,t_{ji}}}(y_{ji})\, \gamma_{j,t_{ji}} \right] \prod_{t=1}^{\infty} b_{\alpha_0}(\gamma'_{jt}) \prod_{t=1}^{\infty} \beta_{k_{jt}} \right) \prod_{k=1}^{\infty} \left[b_{\gamma_0}(\beta'_k)\, \mathscr{F}(f_k) \right]. \end{aligned} \tag{6}$$

We note that this joint density is completely factorized over all its variables. The diagram of model is shown in Figure 1.

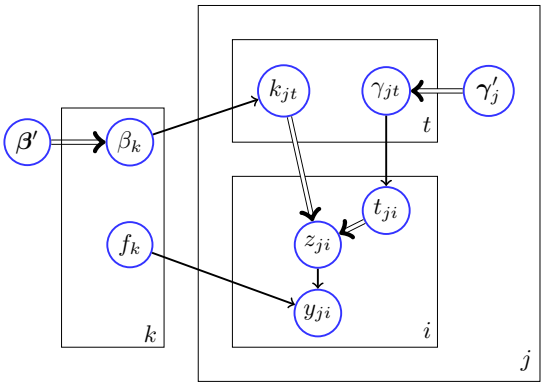

Figure 1: Schematic diagram of HDP mixture with latent variables introduced for sampling. Double arrows show deterministic relations.

3 Sampling

For the most part, when sampling we can ignore the mixture part. That is, for the most part it is enough to sample from (4). Only in sampling $\boldsymbol{t}$ and $\boldsymbol{k}$ the mixture part comes in. The factorized form of the density in (6) allows us to easily derive the Gibbs updates.

3.1 Usual (block) Gibbs sampling

Sampling $\boldsymbol{\gamma'} \mid \boldsymbol{t}, \boldsymbol{k}, \boldsymbol{\beta'}$. This factorizes over $\boldsymbol{\gamma}'_j$, and the posterior of $\boldsymbol{\gamma}'_j$ given the rest is proportional to $\prod_i [F(\boldsymbol{\gamma}'_j)]_{t_{ji}} \prod_t b_{\alpha_0}(\boldsymbol{\gamma}'_{jt})$. Using Lemma 1 (Appendix A) we have

$$\gamma'_{jt} \mid \boldsymbol{\gamma}'_{-jt}, \boldsymbol{t}, \boldsymbol{k}, \boldsymbol{\beta'} \sim \text{Beta}\left(n_t(\boldsymbol{t}_j) + 1, n_{>t}(\boldsymbol{t}_j) + \alpha_0\right). \tag{7}$$

(i.e., for a fixed j, we are applying the lemma to the factorization indexed by t.) Here, $n_t(\boldsymbol{t}_j) = \{i : t_{ji} = t\}|$ and $n_{>t}(\boldsymbol{t}_j) = |\{i : t_{ji} > t\}|$.

Sampling $\boldsymbol{\beta'} \mid \boldsymbol{t}, \boldsymbol{k}, \boldsymbol{\gamma'}$. This posterior is proportional to $\prod_j \prod_t [F(\boldsymbol{\beta'})]_{k_{jt}} \prod_k b_{\gamma_0}(\beta'_k)$. Applying Lemma 1 to the factorization over k, we obtain

$$\beta'_k \mid \boldsymbol{\beta}'_{-k}, \boldsymbol{\gamma'}, \boldsymbol{t}, \boldsymbol{k} \sim \text{Beta}\left(n_k(\boldsymbol{k}) + 1, n_{>k}(\boldsymbol{k}) + \gamma_0\right) \tag{8}$$

where $n_k(\boldsymbol{k}) = |\{(j,t) : k_{j,t} = k\}|$ and similarly for $n_{>k}(\boldsymbol{k})$.

Sampling $\boldsymbol{t} \mid \boldsymbol{k}, \boldsymbol{\gamma'}, \boldsymbol{\beta'}, \boldsymbol{y}, \boldsymbol{f}$. This posterior factorizes over j and i and is given by

$$t_{ji} \mid \boldsymbol{\gamma}'_j, \boldsymbol{k}_j, \boldsymbol{y}_j, \boldsymbol{f} \;\sim\; \left(f_{k_{jt}}(y_{ji})\, \gamma_{jt}\right)_{t \in \mathbb{N}}.$$

Sampling $\boldsymbol{k} \mid \boldsymbol{t}, \boldsymbol{\gamma'}, \boldsymbol{\beta'}, \boldsymbol{y}, \boldsymbol{f}$. This posterior factorizes over j. Writing

$$f_{k_{j,t_{ji}}}(y_{ji}) = \prod_{t=1}^{\infty} \left[f_{k_{jt}}(y_{ji})\right]^{1\{t_{ji}=t\}}$$

the posterior for $\boldsymbol{k}_j$ given the rest is proportional to

$$\prod_{i=1}^{n_j} \left[f_{k_{j,t_{ji}}}(y_{ji})\right] \prod_{t=1}^{\infty} \beta_{k_{jt}} = \prod_{t=1}^{\infty} \left(\beta_{k_{jt}} \prod_{i=1}^{n_j} \left[f_{k_{jt}}(y_{ji})\right]^{1\{t_{ji}=t\}}\right) \tag{9}$$

which also factorizes over t. Thus, it is enough to sample (independently over j and t),

$$k_{jt} \mid \boldsymbol{\beta'}, \boldsymbol{t}_j, \boldsymbol{y}_j, \boldsymbol{f} \;\sim\; \left(\beta_k \prod_{i:\, t_{ij}=t} f_k(y_{ji})\right)_{k \in \mathbb{N}}.$$

Sampling $\boldsymbol{f} \mid \cdots$. Recalling $z_{ji} = k_{j,t_{ji}}$, we sample independently over k,

$$p(f_k \mid \cdots) \;\propto\; \mathscr{F}(f_k) \prod_{(i,j):\, z_{ji}=k} f_k(y_{ji}). \tag{10}$$

3.2 Slice sampling

We recall the basic idea of slice sampling, which itself is a form of variable augmentation: In order to sample from density $f(x)$, introduce the nonnegative variable u and look at the joint density $g(x,u) = 1\{u \le f(x)\}$ whose marginal over x is $f(x)$. Then perform Gibbs sampling on the joint g; in the end keep samples of x and discard those of u. This idea has been successfully employed in [3] to sample from the classical DP mixture. We now extend the ideas in [3] to sample from HDP mixtures. We augment the model by adding variables $\boldsymbol{u}_j = (u_{ji})$ and $\boldsymbol{v}_j = (v_{jt})$ and consider the joint density

$$p(\boldsymbol{y}, \boldsymbol{f}, \boldsymbol{t}, \boldsymbol{k}, \boldsymbol{\gamma}', \boldsymbol{u}, \boldsymbol{\beta}', \boldsymbol{v}) = \\ \prod_{j=1}^{J} \left(\prod_{i=1}^{n_j} f_{k_{j,t_{ji}}}(y_{ji}) 1\{u_{ji} \le \gamma_{j,t_{ji}}\} \prod_{t=1}^{\infty} b_{\alpha_0}(\gamma'_{jt}) \prod_{t=1}^{\infty} 1\{v_{jt} \le \beta_{k_{jt}}\} \right) \prod_{k=1}^{\infty} \left[b_{\gamma_0}(\beta'_k)\, \mathscr{F}(f_k) \right]. \tag{11}$$

Note that by integrating out the variables (u_{ji}) and (v_{jt}) we get back original joint density (6). The idea is that we sample $(\boldsymbol{\gamma}', \boldsymbol{u})$ jointly given the rest of variables, and similarly for $(\boldsymbol{\beta}', \boldsymbol{v})$.

Sampling $(\boldsymbol{\gamma}', \boldsymbol{u}) \mid \boldsymbol{t}, \boldsymbol{k}, \boldsymbol{\beta}', \boldsymbol{v}$. First we sample $(\boldsymbol{u} \mid \boldsymbol{\gamma}', \boldsymbol{t}, \boldsymbol{k}, \boldsymbol{\beta}', \boldsymbol{v})$ which factorizes and coordinate posteriors are $p(u_{ji} \mid \boldsymbol{\gamma}', \boldsymbol{t} \dots) \propto 1\{u_{ji} \le \gamma_{j,t_{ji}}\}$, that is

$$u_{ji} \mid \boldsymbol{\gamma}', \boldsymbol{t}, \boldsymbol{k}, \boldsymbol{\beta}', \boldsymbol{v} \;\sim\; \mathrm{Unif}(0, \gamma_{j,t_{ji}}).$$

Next we sample from $(\boldsymbol{\gamma}' \mid \boldsymbol{t}, \boldsymbol{k}, \boldsymbol{\beta}', \boldsymbol{v})$. This would be the same as (7).

Sampling $(\boldsymbol{\beta}', \boldsymbol{v}) \mid \boldsymbol{k}, \boldsymbol{t}, \boldsymbol{\gamma}', \boldsymbol{u}$. First we sample $(\boldsymbol{v} \mid \boldsymbol{\beta}', \boldsymbol{k}, \boldsymbol{t}, \boldsymbol{\gamma}', \boldsymbol{u})$ which factorizes and coordinate posteriors are $p(v_{jt} \mid \boldsymbol{\beta}', \boldsymbol{k} \dots) \propto 1\{v_{jt} \le \beta_{k_{jt}}\}$, that is

$$v_{jt} \mid \boldsymbol{\beta}', \boldsymbol{k}, \boldsymbol{t}, \boldsymbol{\gamma}', \boldsymbol{u} \;\sim\; \mathrm{Unif}(0, \beta_{k_{jt}}).$$

Next we sample from $(\boldsymbol{\beta}' \mid \boldsymbol{k}, \boldsymbol{t}, \boldsymbol{\gamma}', \boldsymbol{u})$. This would be the same as (8).

Sampling $\boldsymbol{t} \mid \cdots$. This posterior also factorizes over i, j. From (11), we have

$$\mathbb{P}(t_{ji} = t \mid \boldsymbol{t}_{-ji}, \boldsymbol{k}, \boldsymbol{\gamma}', \boldsymbol{u}, \boldsymbol{\beta}', \boldsymbol{v}) \;\propto\; f_{k_{jt}}(y_{ji}) 1\{u_{ji} \le \gamma_{jt}\} \tag{12}$$

Let $T_{ji} := T(\boldsymbol{\gamma}_j; u_{ji}) := \sup\{t :\; u_{ji} \le \gamma_{jt}\}$. According to the above t_{ji} given everything else will be distributed as

$$t_{ji} \mid \cdots \;\sim\; \big(f_{k_{jt}}(y_{ji}) \big)_{t \,\in\, [T_{ji}]}.$$

In CRF metaphor, T_{ji} is the maximum table index (t) that customer i in restaurant j can hop to at current iteration. Note that different customers are allowed different ranges of tables for their wandering. We note that the update of $\boldsymbol{t}$ is an instance of how the slice sampler truncates an infinite measure. Due to the presence of the indicator in (12) only values of t for which $\gamma_{jt} \ge u_{ji}$ lead to a nonzero probability, i.e., the support of the distribution (12) is contained in $[T_{ji}]$.

Sampling $\boldsymbol{k} \mid \cdots$. This posterior also factorizes over j, t. From (11), the posterior for $\boldsymbol{k}_j$ given the rest is proportional to the same expression (9) but with $\beta_{k_{jt}}$ replaced with $1\{v_{jt} \le \beta_{k_{jt}}\}$. Thus, we have

$$\mathbb{P}(k_{jt} = k \mid \cdots) \;\propto\; 1\{v_{jt} \le \beta_k\} \prod_{i:\, t_{ji}=t} f_k(y_{ji}).$$

Let $K_{jt} := K(\boldsymbol{\beta}; v_{jt}) := \sup\{k :\; v_{jt} \le \beta_k\}$. According to the above k_{jt} given everything else will be distributed as

$$k_{jt} \mid \cdots \;\sim\; \Big(\prod_{i:\, t_{ji}=t} f_k(y_{ji}) \Big)_{k \,\in\, [K_{jt}]}.$$

Algorithm 1 Slice sampler for HDP mixture

Initialize T_j^{cap} and K^{cap} to pre-specified values (say 10).
Initialize $\boldsymbol{t}_j$ and $\boldsymbol{k}_j$ to all-ones vectors.
Initialize (u_{ji}) and (v_{jt}) to independent uniform variables.
while not `CONVERGED`, nor maximum iterations reached **do**
  for $j = 1, \ldots, J$ **do**
    Sample $\gamma'_{jt} \sim \text{Beta}\left(n_t(\boldsymbol{t}_j) + 1, n_{>t}(\boldsymbol{t}_j) + \alpha_0\right)$ for all $t \in [T_j^{\text{cap}}]$.
    Let $[\boldsymbol{\gamma}_j]_{1:T_j^{\text{cap}}} \leftarrow [F(\boldsymbol{\gamma}'_j)]_{1:T_j^{\text{cap}}}$.
    Let $T_{ji} \leftarrow \max\{t : \ u_{ji} \le \gamma_{jt}\}, \forall i \in [n_j]$ and $T_j \leftarrow \max_{i=1,\ldots,n_j} T_{ji}$.
    `doubling`(T_j, T_j^{cap})
  end for
  Sample $\beta'_k \sim \text{Beta}\left(n_k(\boldsymbol{k}) + 1, n_{>k}(\boldsymbol{k}) + \gamma_0\right)$ for all $k \in [K^{\text{cap}}]$.
  Let $[\boldsymbol{\beta}]_{1:K^{\text{cap}}} \leftarrow [F(\boldsymbol{\beta}')]_{1:K^{\text{cap}}}$.
  Let $K_{jt} \leftarrow \max\{k : \ v_{jt} \le \beta_k\}, \forall t \in [T_j^{\text{cap}}], \ j \in [J]$.
  Let $K_j \leftarrow \max_t K_{jt}$, and $K \leftarrow \max_j K_j$.
  `doubling`(K, K^{cap})
  Sample f_k from density $p(f \mid \cdots) \ \propto \ \mathscr{F}(f) \prod_{(i,j):\ z_{ji}=k} f(y_{ji})$ for all $k \in [K^{\text{cap}}]$.
  for $j = 1, \ldots, J$ **do**
    Sample $k_{jt} \sim \left(\prod_{i:\ t_{ji}=t} f_k(y_{ji}) \right)_{k \in [K_{jt}]}$ for all $t \in [T_j^{\text{cap}}]$. Set $\boldsymbol{k}_j \leftarrow (k_{jt})$.
    Sample $v_{jt} \ \sim \ \text{Unif}(0, \beta_{k_{jt}})$ for all $t \in [T_j^{\text{cap}}]$.
    Sample $t_{ji} \sim \left(f_{k_{jt}}(y_{ji}) \right)_{t \in [T_{ji}]}$ for all $i \in [n_j]$. Set $\boldsymbol{t}_j \leftarrow (t_{ji})$.
    Sample $u_{ji} \ \sim \ \text{Unif}(0, \gamma_{j,t_{ji}})$ for all $i \in [n_j]$.
    Set $z_{ji} \leftarrow k_{j,t_{ji}}$.
  end for
end while
macro `doubling`(K, K^{cap})
  if $K < K^{\text{cap}}$, **then** continue **else** $K^{\text{cap}} \leftarrow 1.5K^{\text{cap}}$ and go to the previous iteration.

In CRF metaphor, K_{jt} is the maximum dish index (k) available for substitution at table t in restaurant j at current iteration. Again different tables in the same restaurant have potentially different options for dish exchange. The update of $\boldsymbol{k}$ is another instance where the slice sampler is truncating the infinite measures involved.

Sampling $\boldsymbol{f} \mid \cdots$. This will be the same as (10).

3.2.1 How many atoms to keep?

Let us define

$$T_j := \max_{i=1,\ldots,n_j} T_{ji}, \quad K_j := \max_{t=1,\ldots,T_j} K_{jt}, \quad K := \max_j K_j$$

so that T_j determines the maximum table index "t" which we need to keep track of for restaurant j. Given T_j, one can compute K_j which is the maximum number of dish index "k" we need to keep track of in restaurant j. Note that quantities T_j and K_j will be finite and random; they depend on $\boldsymbol{\gamma}, \boldsymbol{\beta}, \boldsymbol{u}, \boldsymbol{v}$ and get updated in each iteration.

This completes the description of the slice sampler which is summarized in Algorithm 1. The implementation however uses a few other ideas besides the update equations derived earlier. The difficulty is that the count parameters T_{ji}, T_j, K_{jt}, K_j and K are interrelated among themselves and

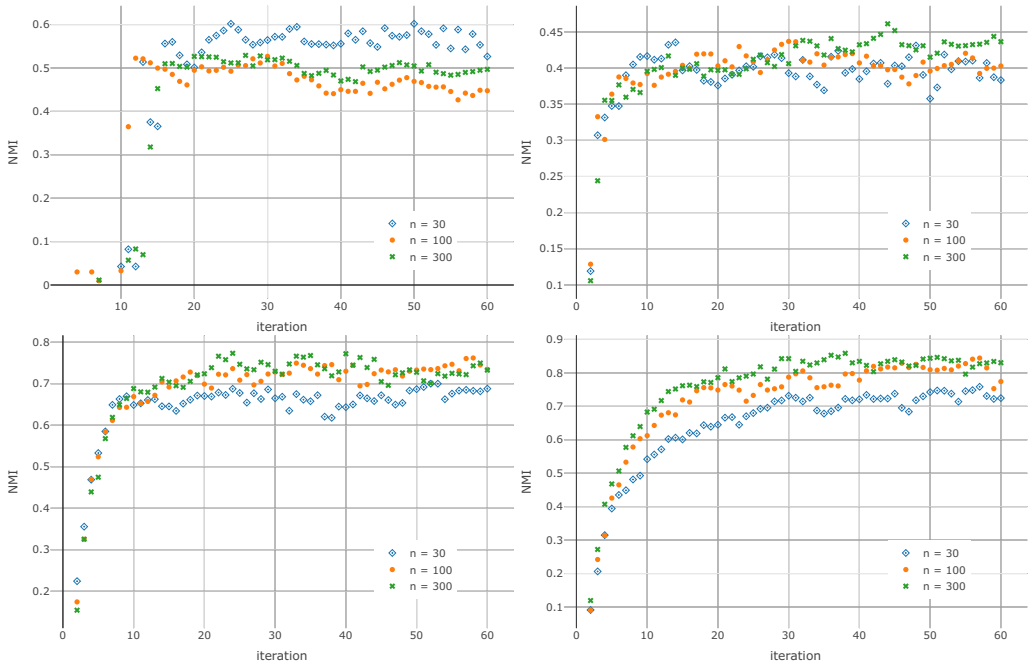

Figure 2: Typical Mixing behavior of the slice sampler for multinomial HDP-mixtures for various sample sizes (Section 3.4). From top-left clockwise: $J = W = 10, 20, 50, 200$. Each plot shows the aggregate normalized mutual information (NMI) at each iteration. The NMI is computed for the labels from the posterior against the "true" labels.

with other latent parameters of the model. Updating some of the parameters while keeping others fixed would create a chain of dependencies which is hard to track. The easiest way to assure that we always have sufficiently enough atoms from all infinite measures is to put caps on their numbers, i.e., T_j^{cap} and K^{cap} in Algorithm 1, and increase the cap whenever we hit it, and repeat the previous iteration (for which we need to keep record of the state of the chain one-step into the past). This is reflected in the `doubling` macro in Algorithm 1. This procedure creates a little bit of redundancy, but after a few steps, the chain will remain within the caps, and one avoids resampling for all but a few early updates. The algorithm guarantees that we always have $T_j < T_j^{\text{cap}}$ and $K < K^{\text{cap}}$ and so the chain is sampling exactly.

An advantage of the slice sampler is that all the updates in each step can be done in parallel over the underlying coordinates. Even updates at multiple steps involving disjoint sets of parameters can be performed in parallel.

Using the more common notation $f_k(y) = K(y; \phi_k)$, step 16 can be written as: Sample ϕ_k from

$$p(\phi \mid \cdots) \;\propto\; H(\phi) \prod_{(i,j):\, z_{ji}=k} K(y_{ji}; \phi) \qquad \text{for all } k \in [K^{\text{cap}}] \tag{13}$$

and set $f_k = K(\,\cdot\,; \phi_k)$. Note that in step 16, if the set $\{(i,j) : z_{ji} = k\}$ is empty for some $k \in [K]$—which could happen since z_{ji} has not yet been updated from the previous iteration while K has just been updated—then the product evaluates to 1, and we draw f_k from the prior $\mathscr{F}(f)$ itself.

We also note that in updating k_{jt} in step 18, if the set $\{i : \; t_{ji} = t\}$ is empty, it means that customers are no longer sitting at table t; as before we interpret products over empty sets as evaluating to 1, hence the dish of the vacant table is updated uniformly at random.

3.3 Examples

Let us consider a few examples of the mixture densities $f_k = K(\cdot; \phi_k)$. As a first example, consider a hierarchical Gaussian mixture: We assume that $\phi_k \mid H \sim H = N(0, I_d/\tau_y^2)$ and that $y_{ji} \mid z_{ji} \sim N(\phi_{z_{ji}}, I_d/\tau_y^2)$ where τ_ϕ^2 and τ_y^2 are the precision parameters of the prior and the likelihood

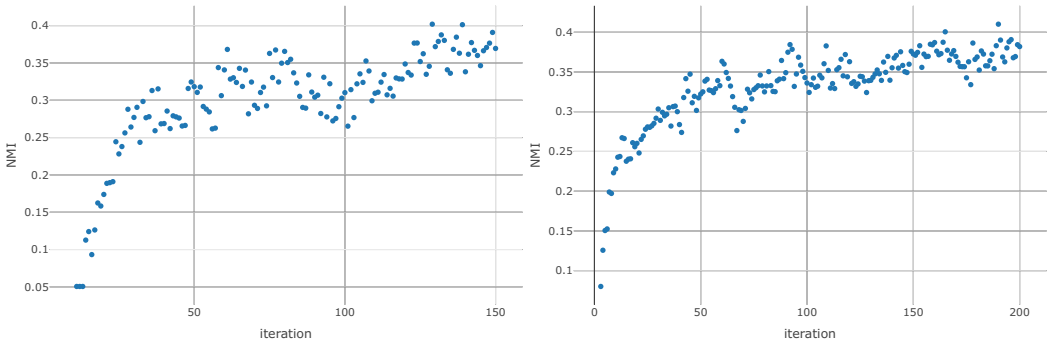

Figure 3: Results for a real data experiment. The NMI (relative to true labels) versus iteration for the real-world paper-title network. (left) performance on a random subset of size $J = 100$ of the data (right) on the whole dataset $J = 894$.

respectively. For this model, it is not hard to see that the posterior update in step 16 which are given in equation (14) in Appendix A.

Consider another example in a topic modeling setup, where y_{ji} represents the ith word of document j, and we assume $y_{ji} \in [W]$ where $[W]$ is a vocabulary of W words. Each atom $\phi \in [0,1]^W$ in this case represents a probability distribution over words in the vocabulary $[W]$. A natural prior on ϕ is $\text{Dir}((\alpha_w))$, i.e., $H(\phi) \propto \prod_{w=1}^{W} \phi_w^{\alpha_w - 1}$, and the likelihood is $y_{ji} \mid z_{ji} \sim \text{Mult}(\phi_{z_{ji}})$ corresponding to the kernel $K(y;\phi) = \phi_y = \prod_{w=1}^{W} \phi_w^{1\{y=w\}}$. The posterior update in step 16 (cf. 13) will be $\phi_k \mid \cdots \sim \text{Dir}(\boldsymbol{\alpha}'_k)$ where $\boldsymbol{\alpha}'_k$ has coordinates $\alpha'_{kw} = \alpha_w + \sum_{j,i} 1\{y_{ji} = w, z_{ji} = k\}$. We also note that updating $\boldsymbol{k}$—step (18)– simplifies to $k_{jt} \sim (\prod_w \phi_{kw}^{\nu'_{jtw}})_{k \in [K_{jt}]}$ where $\nu'_{jtw} = \sum_i 1\{y_{ji} = w, t_{ji} = t\}$.

3.4 Experiments

We now present some numerical experiments to illustrate the performance of the slice sampler. Since HDP mixtures are very popular and their performance well-known, we will mostly focus on studying the mixing time of the sampler. Figure 2 illustrates the mixing behavior for the multinomial HDP mixture discussed in Section 3.3. We have also experimented with the Gaussian mixtures but we omit them here due to similarity. In each case we simulated from the HDP with concentration parameters $\gamma_0 = 3$ and $\alpha_0 = 1$ and have run the slice sampler on a single sample. The multinomial parameter W is varied and we set $\alpha_w = 1/W$. In each case, we have $n_j = n$ for all j and three values $n = 30, 100, 300$ are considered. For simplicity we have set the total number of restaurants $J = W = 10, 20, 50, 200$. Figure 2 illustrates single typical runs of the algorithm without burn-in or thinning; there is also no averaging over multiple runs and the labels are all initialized to 1 as in Algorithm 1. We have calculated the normalized mutual information (NMI) between estimated (z_{ij}) and true labels (z^*_{ij}), aggregated over all (i, j). NMI measures the matching between two clusterings, its value being in $[0, 1]$ with a value of 1 corresponding to a perfect match. Figure 2 shows the quality of recovered labels relative to the true data-generating labels, over the iterations of the sampler. The plots clearly indicate a fast mixing time, somewhere between 10 to 20 iterations. Also worth noting is the decreased variance of the posterior as n increases which is what one would expect.

We have also applied the algorithm to a real world example where the documents are papers and the bag of words are made from the words in their titles. A vocabulary of a total of $W = 189$ was used after running standard text mining procedures for removing the stopwords, stemming, and so on. The information on a total of $J = 894$ papers was collected from the DBLP website. The papers were published in 2017 in three CS topics: machine learning, multimedia and security. We treated DBLP subject classification as the true cluster of each paper. The HDP mixture is run on the dataset which recovers a clustering for every word in each document. We then assign an estimated cluster to each paper by majority voting (among the estimated clusters for their words) and compare with the true labels. Figure 3 illustrates the resulting NMIs versus iteration. Both a random subset of the papers (with $J = 100$) and the whole set is considered. Again, we observe that the algorithm is mixing very fast, and by about 100 iterations we already have pretty good quality labels (NMI $\in [0.35, 0.4]$).

References

[1] Jason Chang and John W Fisher III. Parallel sampling of hdps using sub-cluster splits. In *Advances in Neural Information Processing Systems*, pages 235–243, 2014.

[2] Emily B. Fox, Erik B. Sudderth, Michael I. Jordan, and Alan S. Willsky. A sticky hdp-hmm with application to speaker diarization. *Ann. Appl. Stat.*, 5(2A):1020–1056, 06 2011.

[3] Maria Kalli, Jim E. Griffin, and Stephen G. Walker. Slice sampling mixture models. *Statistics and Computing*, 21(1):93–105, 2011.

[4] Yongdai Kim, Minwoo Chae, Kuhwan Jeong, Byungyup Kang, and Hyoju Chung. An online gibbs sampler algorithm for hierarchical dirichlet processes prior. In Paolo Frasconi, Niels Landwehr, Giuseppe Manco, and Jilles Vreeken, editors, *Machine Learning and Knowledge Discovery in Databases*, pages 509–523, Cham, 2016. Springer International Publishing.

[5] Kenichi Kurihara, Max Welling, and Yee Whye Teh. Collapsed variational dirichlet process mixture models. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, IJCAI'07, pages 2796–2801, San Francisco, CA, USA, 2007. Morgan Kaufmann Publishers Inc.

[6] David Newman, Arthur Asuncion, Padhraic Smyth, and Max Welling. Distributed algorithms for topic models. *J. Mach. Learn. Res.*, 10:1801–1828, December 2009.

[7] Padhraic Smyth, Max Welling, and Arthur U. Asuncion. Asynchronous distributed learning of topic models. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 81–88. Curran Associates, Inc., 2009.

[8] Kyung-Ah Sohn and Eric P. Xing. A hierarchical dirichlet process mixture model for haplotype reconstruction from multi-population data. *Ann. Appl. Stat.*, 3(2):791–821, 06 2009.

[9] Yee Whye Teh, Michael I Jordan, Matthew J Beal, and David M Blei. Hierarchical dirichlet processes. *Journal of the American Statistical Association*, 101(476):1566–1581, 2006.

[10] Chong Wang and David M. Blei. A split-merge mcmc algorithm for the hierarchical dirichlet process. *CoRR*, abs/1201.1657, 2012.

A Useful lemma

Let $x \mapsto b(x;\alpha,\beta)$ be the density of $\mathrm{Beta}(\alpha,\beta)$. The following is a key lemma in deriving the slice sampler updates. The derivations in this paper can be extended to any stick-breaking prior for which a conjugacy relation similar to the one described in the lemma holds:

Lemma 1. *Assume that the joint density of $\boldsymbol{x} = (x_1, x_2, \dots) \in [0,1]^{\mathbb{N}}$ is proportional to*

$$\prod_{i=1}^{n} [F(\boldsymbol{x})]_{z_i} \prod_{j=1}^{\infty} b(x_j;\alpha,\beta),$$

where $\boldsymbol{z} = (z_1,\dots,z_n) \in \mathbb{N}^n$ and $F : [0,1]^{\mathbb{N}} \mapsto [0,1]^{\mathbb{N}}$ is defined as in (2)*. Then*

$$x_j \mid \boldsymbol{x}_{-j} \sim \mathrm{Beta}\left(n_j(\boldsymbol{z}) + \alpha, n_{>j}(\boldsymbol{z}) + \beta\right)$$

where $n_j(\boldsymbol{z}) = |\{i :\ z_i = j\}|$ and $n_{>j}(\boldsymbol{z}) = |\{i :\ z_i > j\}|$.

Proof. Since $[F(\boldsymbol{x})]_j$ only depends on $x_1,\dots,x_{j-1}$, we have

$$\begin{aligned}
p(x_j \mid \boldsymbol{x}_{-j}) &\propto b(x_j;\alpha,\beta) \prod_{i:\ z_i \geq j} [F(\boldsymbol{x})]_{z_i} \\
&= b(x_j;\alpha,\beta) \prod_{i:\ z_i = j} [F(\boldsymbol{x})]_j \prod_{i:\ z_i > j} [F(\boldsymbol{x})]_{z_i} \\
&\propto b(x_j;\alpha,\beta) \prod_{i:\ z_i = j} x_j \prod_{i:\ z_i > j} (1 - x_j) \\
&= b(x_j;\alpha,\beta)\ x_j^{n_j(\boldsymbol{z})} (1-x_j)^{n_{>j}(\boldsymbol{z})}
\end{aligned}$$

which gives the desired result. □

Due to the sick-breaking interpretation of $F(\boldsymbol{x})$, it is not hard to see that

$$\sum_{\ell=1}^{j} [F(\boldsymbol{x})]_\ell + [P(\boldsymbol{x})]_j = 1, \quad \forall j \in \mathbb{N}.$$

where $P : [0,1]^{\mathbb{N}} \to [0,1]^{\mathbb{N}}$ is defined by $[P(\boldsymbol{x})]_j := \prod_{\ell \leq j}(1 - x_\ell)$. That is, $[F(\boldsymbol{x})]_j = x_j [P(\boldsymbol{x})]_{j-1}$ hence $[P(\boldsymbol{x})]_{j-1} < \tau$ implies $[F(\boldsymbol{x})]_j < \tau$. In other words,

$$\{j :\ [F(\boldsymbol{x})]_j \geq \tau\} \ \subseteq\ \{j :\ [P(\boldsymbol{x})]_{j-1} \geq \tau\}$$

and we can use the latter set to guarantee that we have enough atoms when truncating $F(\boldsymbol{x})$ at level τ. This is due to the fact the $j \mapsto [P(\boldsymbol{x})]_j$ is nonincreasing in j as opposed to $j \mapsto [F(\boldsymbol{x})]_j$ which is not necessarily monotone. Note that $[P(\boldsymbol{x})]_{j-1}$ is easy to keep track of since it is the complement to the cumulative distribution associated with $F(\boldsymbol{x})$ up to index $j-1$. We take $[P(\boldsymbol{x})]_0 = 1$.

Gauss updates. For the Gaussian HDP mixture model of 3.3, it is not hard to see that the posterior update in step 16 is equivalent to drawing the atoms from a Gaussian distribution $\phi_k \mid \cdots \sim N(\mu_k, \tau_k^2 I_d)$ where

$$\mu_k = \frac{\tau_y^2}{\tau_\phi^2 + n_k(\boldsymbol{z})\tau_y^2} \sum_{(j,i):\ z_{ji}=k} y_{ji}, \quad \text{and} \quad \tau_k^2 = \tau_\phi^2 + n_k(\boldsymbol{z})\tau_y^2. \tag{14}$$