Accelerated Sampling for the Indian Buffet Process

Finale Doshi-Velez FINALE@ALUM.MIT.EDU
Zoubin Ghahramani ZOUBIN@ENG.CAM.AC.UK
Cambridge University, Trumpington Street, Cambridge CB21PZ, UK

Abstract

We often seek to identify co-occurring hidden features in a set of observations. The Indian Buffet Process (IBP) provides a non-parametric prior on the features present in each observation, but current inference techniques for the IBP often scale poorly. The collapsed Gibbs sampler for the IBP has a running time cubic in the number of observations, and the uncollapsed Gibbs sampler, while linear, is often slow to mix. We present a new linear-time collapsed Gibbs sampler for conjugate likelihood models and demonstrate its efficacy on large real-world datasets.

1. Introduction

Identifying structure is a common problem in machine learning. For example, given a piece of music, we may be interested in jointly identifying that the piece consists of a collection of notes and identifying which notes are played when. Similarly, given a set of images, we may be interested in extracting the objects that compose the images and indicating in which images the objects appear. Traditional machine learning approaches to this problem require the number of underlying features as an input. However, in truly unsupervised settings, the number of hidden features is often unknown, and an inaccurate guess may produce poor results.

The Indian Buffet Process (IBP) (Griffiths & Ghahramani, 2005) is a non-parametric prior on what features are present in which observations. Specifically, the IBP places a prior over feature-assignment matrices Z, where Z_{nk} indicates whether feature k is present in observation n. The number of features in each observation is finite with probability one, but new features as more data are observed. For example, as we see more images, we expect to see objects not present in the previous images. This property has made the IBP useful for modelling choice behaviour (Görür et al., 2006), similarity judgements (Navarro & Griffiths, 2008), protein interactions (Chu et al., 2006), and dyadic data (Meeds et al., 2007).

Appearing in *Proceedings of the 26th International Conference on Machine Learning*, Montreal, Canada, 2009. Copyright 2009 by the author(s)/owner(s).

The IBP provides a principled way to determine the number of underlying features in a dataset, but its usefulness has been limited because the inference is often computationally expensive. Current procedures for the IBP include collapsed and uncollapsed Gibbs sampling (Griffiths & Ghahramani, 2005), which may be augmented with Metropolis split-merge proposals (Meeds et al., 2007), slice sampling (Teh et al., 2007), particle filtering (Wood & Griffiths, 2007), and variational inference (Doshi-Velez et al., 2009). As the number of observations increase, collapsed samplers suffer because the likelihood computations often grow super-linearly in the number of observations N. Uncollapsed samplers suffer because their harder constraints mean that they are usually slow to mix.

We present an accelerated Gibbs sampler for the conjugate linear-Gaussian model that retains the benefits of collapsing the feature identities but has a linear per-iteration running time. Inspired from Maximization-Expectation clustering (Welling & Kurihara, 2009), where hard cluster assignments are combined with distributions over cluster indices, our accelerated sampler samples the feature assignments Z but keeps a posterior over the feature identities. We use this posterior to efficiently compute the likelihood of an observation without being constrained by a single, sampled feature identity. Our approach easily scales to datasets of 10,000 observations or 1,500 dimensions. More generally, we believe that maintaining posteriors over a few key variables within a sampler has the potential to speed computations in a variety of applications.

2. Latent Feature Model

The Indian Buffet Process model posits that each observation can be explained by a set of latent features. The feature-assignment matrix Z describes what features are present in what observations: Z_{nk} is 1 if feature k is present in observation n and 0 otherwise. To generate a sample from the IBP, first imagine that the rows of Z (the observations) are customers and the columns of Z (the features) are dishes in an infinite buffet. The first customer takes the first Poisson(α) dishes. The following customers try previously sampled dishes with probability m_k/n, where m_k is the number of people who tried dish k before customer n. Each customer also takes Poisson(α/n) new dishes. The value Z_{nk} records if customer n tried dish k.

The exchangeability properties of the IBP ensure the order in which the customers attend the buffet has no impact on the distribution of Z. Customers will use "earlier" columns more than "later" ones, but if features are sampled independently from the column index, then they will also be exchangeable. Since the ordering of the columns in Z is arbitrary, it is convenient to define a distribution over $[Z]$, a form of Z that reorders the columns depending on the history of observations who have used that feature (see (Griffiths & Ghahramani, 2005) for details):

$$p([Z]) = \frac{\alpha^K}{\prod_{h=1}^{2^N-1} K_h!} \exp\{-\alpha \sum_{n=1}^{N} \frac{1}{n}\} \prod_{k=1}^{K} \frac{(N-m_k)!(m_k-1)!}{N!}, \quad (1)$$

where K is the number of nonzero columns in Z, m_k is the number of observations using feature k, and K_h is the number of columns in Z with binary representation h. We refer to these K features as the initialised features, that is, the features that have been observed so far. The parameter α controls the expected number of features in each observation. Equation 1 ensures that the number of nonzero columns K will be finite for finite N with probability one without bounding K.

To complete the model, we now describe how observations are generated. Let A be the set of features. In this work, we focus on a linear-Gaussian model in which the observed data X is generated by

$$X = ZA + \epsilon, \quad (2)$$

where A is a KxD matrix of K 1xD features, with a Gaussian prior, and the noise ϵ is independent of Z and A and uncorrelated across observations (see figure 1 for an illustration). However, the core ideas of our algorithm apply to any likelihood model in which $P(X|Z) = \int P(X|Z,A)P(A)dA$ may be computed or approximated efficiently. For example, A and ϵ could be drawn from other stable distributions having support over the entire real line.

While applications may not have an linear-Gaussian form—for example, Fourier coefficients are non-negative—the Gaussian prior is often sufficient. Examples include: A is the frequency signature of a note and Z is when notes are played, A is the characteristic shape of a neuron's spike and Z is when neurons are firing, or A is an object and Z is what objects are present in an image. The noise may come from imperfections in the microphone, voltmeter, or camera.

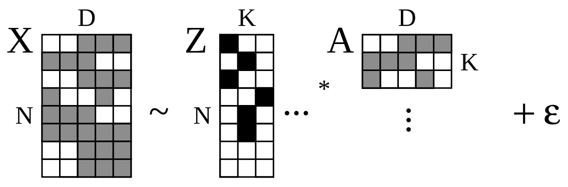


Figure 1. The latent feature model proposes the data X is the product of Z and A with some noise.

3. Gibbs Sampler

Both collapsed and uncollapsed Gibbs samplers have been derived for the IBP (Griffiths & Ghahramani, 2005). Both samplers use the exchangeability property of the IBP to imagine that the current observation n is the last customer to have entered the buffet. If we can compute $p(X|Z)$ efficiently, then for all initialised features k, we can sample Z_{nk} via

$$p(Z_{nk} = 1 | Z_{-nk}, X) \propto \frac{m_k - Z_{nk}}{N} p(X|Z) \quad (3)$$

where m_k is the number of previous observations containing feature k. Note that if observation n is the only observation using feature k, then the Gibbs sampler will always remove it. However, new features may then be sampled based on

$$p(k_{new}) \propto \mathsf{Poisson}(k_{new}, \frac{\alpha}{N}) p(X|Z_{new}) \quad (4)$$

where Z_{new} is the feature-assignment matrix with k_{new} additional columns set to one for feature n. We compute these probabilities to some truncation level k_{max}.

For the linear-Gaussian model, the collapsed likelihood $P(X|Z)$ is given by

$$p(X|Z, \sigma_x, \sigma_a) \quad (5)$$
$$= \frac{\exp(-\frac{1}{2\sigma_x^2}(X^T(I - Z(Z^TZ + \frac{\sigma_x^2}{\sigma_a^2}I)^{-1}Z^T)X))}{(2\pi)^{\frac{ND}{2}} \sigma_X^{(N-K)D} \sigma_a^{KD} |Z^TZ + \frac{\sigma_x^2}{\sigma_a^2}I|^{\frac{D}{2}}}$$

where σ_x^2 is the noise variance and σ_a^2 is the feature variance. As seen in equation 5, integrating out the features A when computing $p(X|Z)$ correlates the observations. Thus the collapsed Gibbs sampler has a relatively high computational complexity per iteration. Griffiths and Ghahramani (2005) describe a set of rank-one updates to reduce this computational cost, but the updates remain $O(N^3)$ per iteration.

The uncollapsed version of the Gibbs sampler trades a faster per-iteration runtime for (generally) slower mixing. It represents the features A explicitly and and alternates between sampling A and sampling Z. The sampling equations are identical to equations 3 and 4 except that all probabilities are now also conditioned on A. For example, $p(Z_{nk} = 1|Z_{-nk}, X)$ for existing features is now proportional to $\frac{m_k - Z_{nk}}{N} p(X|Z, A)$. When considering to add new features, the uncollapsed sampler draws k_{new} features from the prior $p(A)$, and then conditions the likelihood on those features.[1] When conditioning on A, the likelihood $p(X|Z, A)$ factors into $\prod_n p(X_n|Z_n, A)$. Thus, if only Z_{nk} is changed, only $p(X_n|Z_n, A)$ changes in $p(X|Z, A)$.

In the linear-Gaussian model, $p(A|X, Z)$ is Gaussian[2] with mean μ^A and variance Σ^A:

$$\begin{aligned} \mu^A &= (Z^T Z + \frac{\sigma_x^2}{\sigma_a^2} I)^{-1} Z^T X \qquad (6) \\ \Sigma^A &= \sigma_x^2 (Z^T Z + \frac{\sigma_x^2}{\sigma_a^2} I)^{-1} \end{aligned}$$

4. Accelerated Sampling

We describe a sampler that mixes like the collapsed Gibbs sampler but has a running time like the uncollapsed Gibbs sampler. Computing the collapsed likelihood (equation 5) is expensive because the likelihood of each Z_{nk} depends on the entire dataset. The graphical model in figure 2) illustrates this dependence, where we have split the observations into two parts. The "bottom" X_W matrix represents a window containing the last W observations and X_{-W} represents the other observations. If A is not observed, inference on Z_W depends on both X_W and X_{-W}. In contrast, if A is observed—as in the uncollapsed Gibbs sampler—inference on Z_W depends only on the corresponding data X_W. (The Z_{-W} dependence is easy to compute.)

Our accelerated sampler maintains the posterior $p(A|X_{-W}, Z_{-W})$. By keeping the posterior, instead of sampling a fixed value for A, the accelerated sampler retains the flexibility—that is, the mixing properties—of the collapsed Gibbs sampler (regardless of W). However, similar to the uncollapsed Gibbs sampler, the posterior blocks the dependence of Z_W on X_{-W}. As a result, the accelerated Gibbs sampler has similar runtime to the uncollapsed sampler.

[1]We found that sampling new features from the prior seemed to be one of the key factors that slowed down the mixing of the uncollapsed sampler. Our experiments used a semi-collapsed Gibbs sampler in which initialised features were instantiated, but new features were integrated out.

[2]The mean μ^A has the dimensionality of A (KxD). All dimensions are independent with identical variances, so the joint covariance is block diagonal with D identical KxK blocks. We let Σ^A be the KxK covariance for $A_{\cdot,d}$.

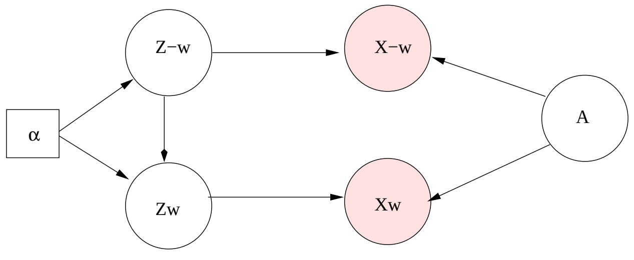


Figure 2. Graphical model for the IBP, showing the observations and the feature-assignment matrix split into two (arbitrary) parts. The observations corresponding to Z_W occur "after" Z_{-W} and depend on the counts of Z_{-W}.

Formally, let X_W denote some window of observations containing observation n. The exchangeability of the IBP allows us to imagine that X_W are the final W observations and X_n is the last observation. Using Bayes rule, we write the probability $p(Z_{nk} = 1|Z_{-nk}, X)$ as:

$$\begin{aligned} p(Z_{nk} = 1|Z_{-nk}, X) &\propto p(Z_{nk}|Z_{-nk}) p(X|Z) \\ &= \frac{m_k}{n} \int_A p(X|Z, A) p(A) dA. \end{aligned}$$

We split the data into sets X_W and X_{-W} and apply the conditional independencies from figure 2 to get

$$\begin{aligned} &p(Z_{nk} = 1|Z_{-nk}, X) \\ &= \frac{m_k}{n} \int_A p(X_W, X_{-W}|Z_W, Z_{-W}, A) p(A) dA \\ &= \frac{m_k}{n} \int_A p(X_W|Z_W, A) p(X_{-W}|Z_{-W}, A) p(A) dA. \end{aligned}$$

Finally, we apply Bayes rule again to $p(X_{-W}|Z_{-W}, A)$:

$$\begin{aligned} &p(Z_{nk} = 1|Z_{-nk}, X) \qquad (7) \\ &\propto \frac{m_k}{n} \int_A p(X_W|Z_W, A) p(A|X_{-W}, Z_{-W}) dA \end{aligned}$$

Thus, given $p(A|X_{-W}, Z_{-W})$, we can compute $p(Z_{nk} = 1|Z_{-nk}, X)$ exactly without touching X_{-W}.

In the linear-Gaussian model, the feature posterior $p(A|X_{-W}, Z_{-W})$ and likelihood $p(X_W|Z_W, A)$ are both Gaussian, and the integral in equation 7 yields

$$\begin{aligned} &P(Z_{nk}|Z_{-nk}, X) \qquad (8) \\ &\propto \frac{m_k}{n} \mathcal{N}(X_W; Z_W \mu^A_{-W}, Z_W \Sigma^A_{-W} Z_W^T + \Sigma^X), \end{aligned}$$

where (μ^A_{-W},Σ^A_{-W}) is the mean and covariance of the feature posterior and $\Sigma^X = \sigma_x^2 I$ is the noise variance.

The accelerated sampling procedure, summarized in Algorithm 1, only uses a small window of observations X_W at one time. We first compute the feature posterior given all of the data using equation 6.

Algorithm 1 Accelerated Gibbs Sampler

```
Initialize $Z$.
Compute $p(A|Z,X)$.
for each iteration do
  $S \leftarrow \{1 \dots N\}$
  while $S$ is not empty do
    Sample and remove $W$ elements from $S$.
    Compute $p(A|Z_{-W}, X_{-W})$ from $p(A|Z,X)$.
    for each observation $n$ in $W$ do
      Sample $Z_{nk}$ according to equation 9.
      Sample new features for each observation $n$
        according to equation 4.
    end for
    Update $p(A|Z,X)$ from $p(A|Z_{-W}, X_{-W})$.
  end while
end for
```

Next, we randomly choose W observations as our window. Given the full feature posterior $p(A|X,Z) = \mathcal{N}(\mu^A, \Sigma^A)$, the posterior $p(A|X_{-W}, Z_{-W})$ with the window X_W removed is given by

$$\begin{aligned}\Sigma^A_{-W} &= (I - \Sigma^A Z_W^T (Z_W \Sigma^A Z_W^T - \Sigma^X)^{-1} Z_W)\Sigma^A \\ \mu^A_{-W} &= b(\mu^A - \Sigma^A Z_W^T (Z_W \Sigma^A Z_W^T + \Sigma^X)^{-1} X_W)\end{aligned}$$

where $b = (I - \Sigma^A Z_W^T (Z_W \Sigma^A Z_W^T + \Sigma^X)^{-1} Z_W \Sigma^A)^{-1}$. Now each Z_{nk} in Z_W may be sampled using equation 9. New features are sampled using equation 4.[3]

Each sampling step requires the computation of $P(X_W|Z_W) = \mathcal{N}(X_W; Z_W \mu^A_{-W}, Z_W \Sigma^A_{-W} Z_W^T + \Sigma^X)$. If only one element Z_{nk} of Z_W changes, the new $(\Sigma^X_W)^{-1}$ and its determinant can be computed by a pair of rank-one updates; new features may also be incorporated into the covariance via a single rank-one update. If the Woodbury formulas are used, the updates require $O(W^2)$ elementary operations. Griffiths and Ghahramani (2005) show an $O(K^2)$ update is also possible. However, the $O(W^2)$ inversions will be faster because we expect K to grow with the number of observations while W can be fixed to remain small.

Once we have completed sampling the window, we recompute the full feature posterior $p(A|X,Z)$:

$$\begin{aligned}\mu^A &= \mu^A_{-W} + \Sigma^A_{-W} Z_W^T \\ &\quad \cdot (Z_W \Sigma^A_{-W} Z_W^T + \Sigma^X)^{-1} (X_i - Z_W \mu^A_{-W}) \\ \Sigma^A &= (I - \Sigma^A_{-W} Z_W^T (Z_W \Sigma^A_{-W} Z_W^T + \Sigma^X)^{-1} Z_W) \Sigma^A_{-W}\end{aligned}$$

For each iteration, the sampler then chooses W new observations without replacement and repeats until all N observations have been updated.

[3]In practice, using the information form $P^A = (\Sigma^A)^{-1}$, $h^A = P\mu^A$ results in slightly more stable updates. We convert P^A_{-W} and h^A_{-W} to Σ^A_{-W} and μ^A_{-W} before computing likelihoods, so the computational complexity is unchanged.

5. Per-Iteration Running Times

We consider the number of addition and multiplication operations for one sweep through an NxK feature-assignment matrix Z under a linear-Gaussian likelihood model (ignoring the addition of new features). The running time of the collapsed Gibbs sampler is dominated by the computation of the exponent $X^T(I - Z(Z^TZ + I\sigma_x^2/\sigma_a^2)^{-1}Z^T)X$. If only one element of Z is changed, then the inverse $(Z^TZ + I\sigma_x^2/\sigma_a^2)^{-1}$ may be updated in $O(K^2)$ time. However, the remaining matrix products require $O(N^2K)$ and $O(N^2D)$ operations respectively. Thus, for NK elements, the running time for large N is $O(NK(N^2K + N^2D))$.

The uncollapsed Gibbs sampler requires many fewer operations per element: $p(X_n|Z_n, A)$ is independent of the remaining observations and only requires the computation of Z_nA, which is $O(KD)$. The features A are resampled once per iteration. Computing the mean and variance for A given Z requires steps that $O(K^3 + NK^2 + NKD)$, but the $O(NK^2D)$ time required to sample NK elements dominates these terms.

Finally, as with the collapsed Gibbs sampler, the accelerated Gibbs sampler's per-iteration running time also has a dominant term from computing the likelihood. If Woodbury inversions are used, the likelihood computations are $O(W^2 + DW^2)$, for a complexity $O(NKDW^2)$ for sampling all the Z_{nk}. However, the feature posterior must also be updated N/W times, and each update requires steps of complexity $O(W^3 + W^2K + WK^2)$. Thus, the overall complexity is $O(N(KDW^2 + K^2))$, and the optimal value for W is 1.[4] Like the uncollapsed Gibbs sampler, the accelerated sampler's complexity is linear in N.

Table 1 summarises the running times. Both the uncollapsed and accelerated Gibbs sampler are linear in N, but the accelerated Gibbs sampler has a slightly better complexity in K and D. The lower order dependence is beneficial for scaling because we expect K to grow with N. While constants and lower-order terms are important, our experiments confirm the lower complexity is significant in the larger datasets.

6. Experiments

We compared the computational efficiency and inference quality of the collapsed Gibbs sampler, the semi-collapsed Gibbs sampler (an uncollapsed Gibbs sampler that integrates over the new features when sampling k_{new}), and the accelerated Gibbs sampler. All samplers were optimised to take advantage of rank-one updates and vectorisation.

[4]We included W in our original formulation because in general the optimal choice of W will depend on the relative costs of computing the feature posterior and the likelihood.

Table 1. Per-iteration running times for large N and a linear-Gaussian likelihood model, given an NxK matrix Z and D-dimensional data.

Algorithm	Running Time
Collapsed Gibbs	$O(N^3(K^2+KD))$
Uncollapsed Gibbs	$O(NDK^2)$
Accelerated Gibbs	$O(N(KDW^2+K^2))$
Accelerated Gibbs, W=1	$O(N(K^2+KD))$

Per-iteration runtimes were used to evaluate computational efficiency. Since Gibbs sampling produces a sequence of correlated samples, another important metric was the degree of independence between successive samples: more independent samples indicate faster mixing. With the synthetic data, we ran 5 chains in parallel to evaluate the effective number of independent samples per actual sample (Gelman et al., 2003). This quantity will be 1 if successive samples are independent and less than 1 otherwise. Experiments with real-world data took longer to complete, so we estimated the burn-in time instead. In both cases, the number of features K and the training log-likelihood were the chain statistics used to evaluate mixing.

We evaluated the inference quality by first holding out approximately $100D$ X_{nd} values during the initial inference. No observation had all of its dimensions missing, and no dimension had all of its observations missing. The quality of the inference was measured by evaluating the test log-likelihood and the L2 reconstruction error of the missing elements. The test metrics were averaged from the final five samples.

Synthetic Data The synthetic data were generated from the IBP prior ($\alpha = 2$) and the linear-Gaussian model ($\mu^A = 0, \sigma_a = 2, \sigma_x = .2$, $D = 10$). We ran 5 chains from different starting positions for each sampler with N equal to 50, 100, 250, and 500 for 1000 iterations. Figure 3 shows the evolution of the per-iteration runtime as the number of observations grows (note the log-log scale). The slopes of the lines indicate the order of the runtime polynomial. The semi-collapsed and accelerated Gibbs samplers have nearly identical slopes, while the per-iteration running time of the collapsed Gibbs sampler scales much more poorly.

Figure 4 plots the effective sample count (per sample) of for each sampler. Again, as expected, the semi-collapsed Gibbs sampler had the lowest effective sample count, and the accelerated Gibbs sampler and the collapsed Gibbs sampler had similar counts. From these two plots, we see that the accelerated Gibbs sampler mixes like the collapsed Gibbs sampler but has a run-time like the uncollapsed Gibbs sampler.

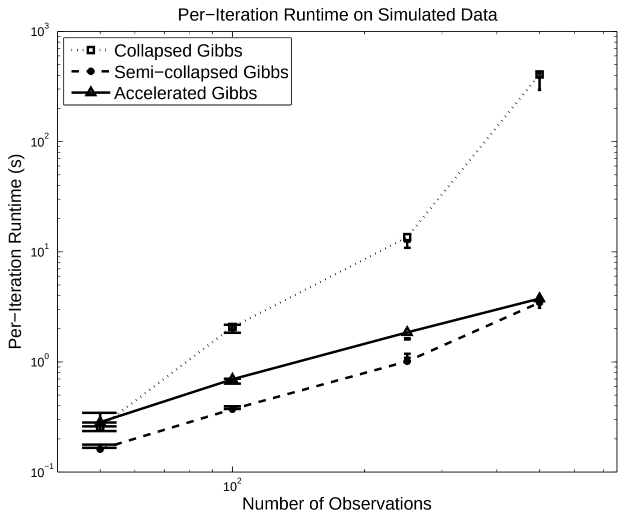


Figure 3. Per iteration runtime vs. number of observations (note the log scale on the x-axis).

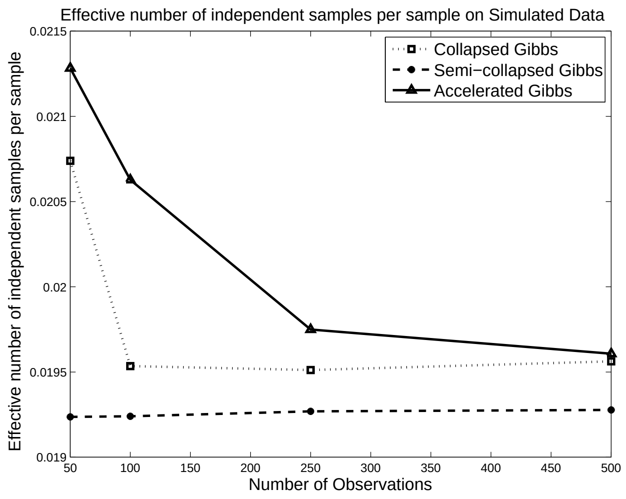


Figure 4. Effective number of samples vs. dataset size.

Table 2. Performance statistics on data from the prior. Each value is an average over 5 chains.

Evaluation Statistic	Collapsed Gibbs	Semi-collapsed Gibbs	Accelerated Gibbs
Iteration Time	90.273	**1.302**	1.566
Effective N	0.020	0.019	**0.020**
L2 Test Error	**0.710**	4.534	1.780
Test log Likelihood	-24.557	**-13.977**	-26.936

The running times and quality criteria are summarised in table 2. The accelerated Gibbs sampler's running time is on par with the semi-collapsed Gibbs sampler and nearly two orders of magnitude faster than the collapsed Gibbs sampler. It does not achieve the best test likelihoods or reconstruction errors, but they are always on par with collapsed Gibbs sampler.

Real Data Table 3 summarises the real-world data sets. All datasets were first centred to have 0-mean. Values for the hyperparameters σ_a and σ_x were fixed to $.75\sigma$ and $.25\sigma$, respectively, where σ was the standard deviation of the observations across all dimensions. We set $\alpha = 2$ in all experiments. Each sampler ran for 500 iterations or up to 150 hours.

Table 3. Descriptions of datasets.

Dataset	N	D	Description
Block-Images[a] (Wood & Griffiths, 2007)	1000	36	noisy overlays of four binary shapes on a grid
Emoticons[b]	702	1032	different smiley faces
Yale (Georghiades et al., 2001)	722	1032	faces with various lighting
AR (Mart'inez & Kak, 2001)	2600	1598	faces with lighting, accessories
EEG (Hoffmann et al., 2008)	4400	32	EEG recording on various tasks
Piano (Poliner & Ellis, 2007)	10000	161	DFT of a piano recording

[a]The block images are not a real-world dataset, but they do not come from the linear-Gaussian model.

[b]Obtained from www.smileyset.com; post-processed to normalise image size and centre.

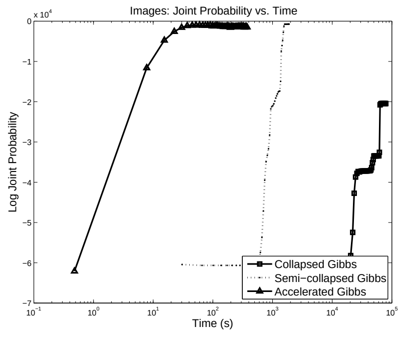
Figure 5 plots how the log-joint probability $\log p(X, Z)$ of the training data evolves for all six datasets (note the log-scale on the time axis). In almost all the datasets, the accelerated Gibbs sampler equilibrates to the same log-joint probability as the semi-collapsed Gibbs sampler but orders of magnitude faster. The plots suggest the time allowed was not sufficient for the collapsed Gibbs sampler to complete burning-in, though it seems to be approaching a similar joint probability. For the higher dimensional and larger datasets in the bottom row, the collapsed Gibbs sampler often failed to complete even one iteration in 150 hours. The semi-collapsed sampler also suffered in the higher dimensional datasets. The accelerated Gibbs sampler always completed its run in less than two hours.

Table 4 shows the per-iteration running times and the measures on the missing data for the six datasets. A dash indicates insufficient iterations were completed to compute the statistic. The speed benefit of the accelerated Gibbs sampler is particularly apparent on these larger datasets: often it is two orders of magnitude faster per-iteration than the collapsed sampler. However, its test L2 reconstruction error and likelihood remains on-par or better than either sampler. For slightly troublesome EEG dataset, most of the likelihood loss came from 4 dimensions of 4 observations. These four troublesome dimensions had a variance about two orders of magnitude larger than the remaining dimensions (the other datasets were more homogenous). All samplers had to manage this issue, but the accelerated Gibbs sampler's heavy reliance on rank-one updates made it particularly sensitive to ill-conditioning from extreme situations.

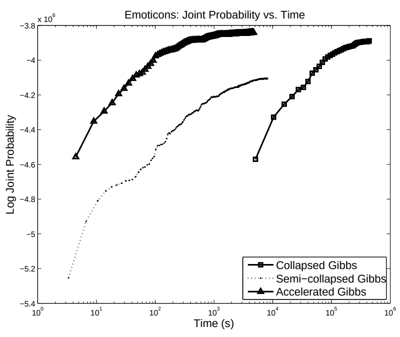
Table 4. Results on realworld data sets. Dashes indicate values that could not be computed in the time allotted.

Data Set	Collapsed Gibbs	Semi-collapsed Gibbs	Accelerated Gibbs
	Iteration Time		
Images	1526.4	38.1	**7.6**
Emoticons	10205.4	32.8	**19.0**
Yale	8759.0	**18.1**	25.5
AR Faces	-	13291.8	**120.9**
EEG	112692.3	**13.0**	21.9
Piano	-	-	**221.5**
	Burn-in Count		
Images	12.0	**2.0**	4.0
Emoticons	**35.0**	82.0	42.0
Yale	**45.0**	184.0	133.0
AR Faces	-	-	**31.0**
EEG	-	-	**29.0**
Piano	-	-	**25.0**
	L2 Test Error		
Images	0.1066	0.1239	**0.0555**
Emoticons	5905.8	11781.1	**5612.7**
Yale	459.4	**413.0**	435.9
AR Faces	-	3836.4	**919.9**
EEG	267292.3	282108.1	**215174.5**
Piano	-	-	**0.0005**
	Test Likelihood		
Images	-2.0	**-1.7**	-2.0
Emoticons	-14.4	**-12.0**	-14.2
Yale	-17.8	**-15.8**	-16.0
AR Faces	-	**-13.3**	-13.7
EEG	-26274.0	**-3621.5**	-14133.3
Piano	-	-	**-7.0**

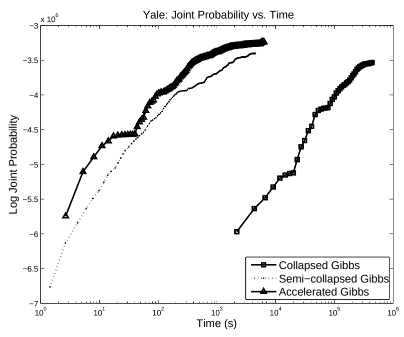
Finally, figures 6 and 7 show qualitative results on the emoticon and AR faces datasets. Our primary objective here was to show that the accelerated sampler finds reasonable features (rather than discover previously unknown structure in the data). Each figure decomposes three observations into features found by the sampler (all features contribute equally in the additive model). The sampler finds features associated with the emoticon's expression (first column of 'Parts') and features that make up emoticon's accessories. In



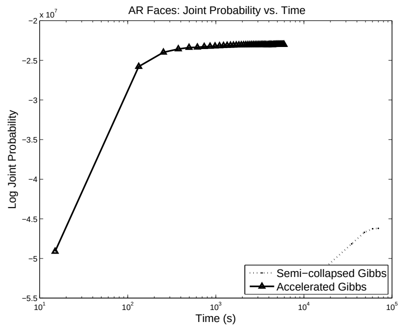
(a) Block Images: N = 1000, D = 36



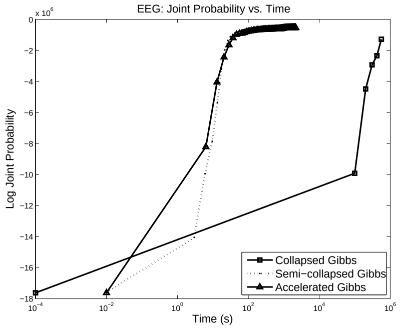
(b) Emoticons: N = 702 , D = 1032



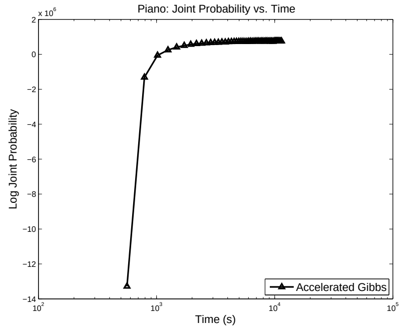
(c) Yale Faces: N = 722 , D = 1032



(d) AR Faces: N = 2600 , D = 1598



(e) EEG: N = 4400 , D = 32



(f) Piano: N = 10000 , D = 161

Figure 5. Evolution of the log joint-probability vs. time for several datasets.

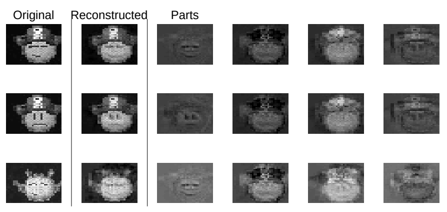


Figure 6. Some reconstructed emoticons: the expression is a different feature than the type of the face.

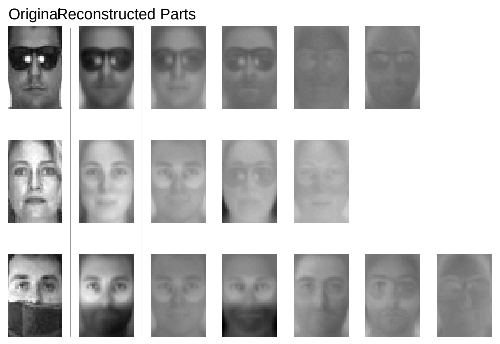


Figure 7. Some reconstructed AR faces: faces contain features for the face, lighting, scarves, and sunglasses.

the larger AR Faces dataset, the sampler finds features corresponding to lighting conditions, facial features, and accessories such as scarves or sunglasses. These decompositions are similar to those found by the other samplers.

7. Discussion

The accelerated Gibbs sampler attained similar test likelihood and reconstruction errors as the existing Gibbs samplers but did so faster than either approach. Its per-iteration runtime was two orders of magnitude faster than the collapsed Gibbs sampler. While it is difficult to judge the convergence of the samplers from a single run, the accelerated Gibbs sampler's training likelihoods were generally on par with the other samplers, suggesting that the gains in speed were achieved without sacrifices in performance.

The difference between the NK^2D term in the running time of the uncollapsed Gibbs sampler and the $N(K^2 + KD)$ of the accelerated Gibbs sampler becomes more pronounced as the dimensionality of the observations and the number of features increases. If K grows logarithmically with N, as posited by the IBP prior, then large datasets will impact the complexity because K will be larger. For example, in the

AR Faces dataset, where D is 1598 and feature counts ranged between 15-80, the difference between the two expected running times is one to two orders of magnitude. (Of course, implementation details affect the differences in per-iteration running times, but the experiments confirm this speed-up factor.[5])

Loss of precision during the rank-one updates is the primary implementation concern for the accelerated sampler. This effect was particularly apparent in the EEG dataset because the dimensions had widely varying variances, leading to ill-conditioned matrices. An important question is how often expensive full-updates of the feature posterior and data posterior must be completed; we did so once per run and once per window, respectively. While a window size W of 1 produces an optimal runtime, larger windows might slow the degradation due to repeated rank-one updates.

Our comparison focused on performance relative to other Gibbs samplers. Regarding other IBP inference techniques, we expect it to mix faster than the slice sampler, as the slice sampler mixes more slowly than the collapsed Gibbs sampler (Teh et al., 2007). Well-designed MH-proposals may make a Metropolis sampler efficient in some domains, and other domains may be well-suited for particle filtering, but one advantage of the accelerated Gibbs sampler is its simplicity: neither MH-proposals nor a particle count is needed to perform principled, effective inference. However, an interesting question is if population Monte Carlo techniques can be used to merge the benefits of our accelerated Gibbs sampler and particle filtering.

8. Conclusion

We presented a new accelerated Gibbs sampler for the Indian Buffet Process with the mixing properties of the collapsed Gibbs sampler but the running time of the uncollapsed Gibbs sampler. The key insight was to allow data to share information only through the statistics on the feature posterior. Thus, likelihoods of feature assignments could efficiently be computed using local computations without sacrificing the flexibility of a collapsed sampler. The accelerated Gibbs sampler scaled to inference for several large, real-world datasets. An interesting future direction may be to explore how maintaining distributions over parameters may help accelerate other MCMC inference methods.

References

Chu, W., Ghahramani, Z., Krause, R., & Wild, D. (2006). Identifying protein complexes in high-throughput protein interaction screens using an infinite latent feature model. *Pacific Symposium on Biocomputing* (pp. 231–242).

Doshi-Velez, F., Miller, K. T., Gael, J. V., & Teh, Y. W. (2009). Variational inference for the Indian buffet process. *Proceedings of the Intl. Conf. on Artificial Intelligence and Statistics* (pp. 137–144).

Gelman, A., Carlin, J. B., Stern, H. S., & Rubin, D. B. (2003). *Bayesian Data Analysis.* Chapman & Hall/CRC.

Georghiades, A., Belhumeur, P., & Kriegman, D. (2001). From few to many: Illumination cone models for face recognition under variable lighting and pose. *IEEE Trans. Pattern Anal. Mach. Intelligence*, *23*, 643–660.

Görür, D., Jäkel, F., & Rasmussen, C. E. (2006). A choice model with infinitely many latent features. *Intl. Conf. on Machine Learning* (pp. 361–368).

Griffiths, T., & Ghahramani, Z. (2005). Infinite latent feature models and the Indian buffet process. *TR 2005-001, Gatsby Comp. Neuroscience Unit.*

Hoffmann, U., Vesin, J.-M., Ebrahimi, T., & Diserens, K. (2008). An efficient P300-based brain-computer interface for disabled subjects. *Journal of Neuroscience Methods*, *167*, 115–125.

Mart'inez, A. M., & Kak, A. C. (2001). PCA versus lDA. *IEEE Trans. Pattern Anal. Mach. Intelligence*, *23*, 228–233.

Meeds, E., Ghahramani, Z., Neal, R., & Roweis, S. (2007). Modeling dyadic data with binary latent factors. *Advances in Neural Information Processing Systems* (pp. 977–984).

Navarro, D. J., & Griffiths, T. L. (2008). Latent features in similarity judgments: A nonparametric Bayesian approach. *Neural Comp.*, *20*, 2597–2628.

Poliner, G. E., & Ellis, D. P. W. (2007). A discriminative model for polyphonic piano transcription. *EURASIP J. Appl. Signal Process.*, *2007*, 154–154.

Teh, Y. W., Görür, D., & Ghahramani, Z. (2007). Stick-breaking construction for the Indian buffet process. *Proceedings of the Intl. Conf. on Artificial Intelligence and Statistics* (pp. 556–563).

Welling, M., & Kurihara, K. (2009). Bayesian K-means as a "maximization-expectation" algorithm. *Neural Computation* (pp. 1145–1172).

Wood, F., & Griffiths, T. L. (2007). Particle filtering for nonparametric Bayesian matrix factorization. *Advances in Neural Information Processing Systems* (pp. 1513–1520).

[5]Using the semi-collapsed Gibbs sampler instead of the uncollapsed Gibbs sampler will also affect the runtime, but this effect appeared to be small in preliminary experiments.