STA663 Statistical Computation Final Project

Implementation of the Indian Buffet Process (IBP)

Christine P. Chai

Executive Summary

1 Introduction

The paper I selected is "Infinite Latent Feature Models and the Indian Buffet Process" (IBP) [5]. In unsupervised machine learning, discovering the hidden variables that generate the observations is important. Many statistical models [1, 3] can provide a latent structure in probabilitistic modeling, but the problem lies in the unknown dimensionality, i.e. how many classes/features to express the latent structure. Bayesian nonparametric methods are able to determine the number of latent features; the Chinese Restaurant Process (CRP) is an example [4], but it assigns each customer to a single component (table). The Indian Buffet Process allows each customer to be assigned to multiple components (dishes), and the process can serve as a prior for an potentially infinite array of objects. In my implementation, IBP is regarded as a prior for the linear-Gaussian binary latent feature model, and I referred to some Matlab code online [9, 8].

1.1 Algorithm Description

The Indian Buffet Process is a metaphor of Indian restaurants offering buffets with a close-to-infinite number of dishes, and the number of dishes sampled by a customer is a Poisson distribution. Assume N customers enter a restaurant one after another, and the first customer takes a Poisson(α) of dishes. Starting from the second person, the ith customer takes dish k with probability $\frac{m_k}{i}$, where m_k is the number of previous customers who have sampled that dish. In this way, the ith customer samples dishes proportional to their popularity. After reaching the end of all previously sampled dishes, the ith customer tries a Poisson($\frac{\alpha}{i}$) number of new dishes. Which customer sampled which dish is recorded in a binary array Z with N rows (representing customers) and infinitely many columns (representing dishes), where $z_{ik} = 1$ if customer i sampled the dish k. Note that the customers are not exchangeable, i.e. the dishes a customer samples is dependent on whether previous customers have sampled that dish [5].

In terms of probability,

$$P(z_{ik} = 1|\mathbf{z_{-i,k}}) = \frac{m_{-i,k}}{N} \tag{1}$$

The subscript $_{\mathbf{-i,k}}$ indicates dish k and all customers except for the ith one. If the number of dishes is truncated to K, then the above equation becomes

$$P(z_{ik} = 1|\mathbf{z_{-i,k}}) = \frac{m_{-i,k} + \frac{\alpha}{K}}{N + \frac{\alpha}{K}} \tag{2}$$

The N customers can be viewed as objects, and the K dishes can be regarded as features. Formally writing, $Z \sim \mathrm{IBP}(\alpha)$, and

$$P(Z|\alpha) = \frac{\alpha^K}{\prod_{h=1}^{2^N-1} K_h!} \exp(-\alpha H_N) \prod_{k=1}^{K} \frac{(N-m_k)!(m_k-1)!}{N!} \tag{3}$$

α is a variable influencing the number of features (denoted as D in later sections); m_k is the number of objects with feature k; K_h is the number of features with history h (whether the N objects possess this feature, $2^N - 1$ possibilities in total); H_N is the N^{th} harmonic number, i.e. $H_N = \sum_{k=1}^{N} \frac{1}{k}$.

1.2 Applications and Evaluation

Many applications and variations of the Indian Buffet Process exist. For example, the linear-Gaussian binary latent feature model I implemented [9] can be used to model "noisy" matrices and reveal the latent features. In this way, image data can be processed because we can interpret binary matrices with structured representations. For another example, Yildirim and Jacob [10] proposed an IBP-based Bayesian nonparametric approach to multisensory perception in an unsupervised manner. Furthermore, variations of the Indian Buffet Process include focused topic modeling [7], hierarchical beta processes [7], and variational inference [2].

The advantages and disadvantages of IBP are clear. Using a Poisson distribution, IBP is able to model an infinite sequence of integers, and the sequence can be truncated as needed. In the implementation of IBP, the advantages of Gibbs sampling and Metropolis-Hastings (MH) can be combined. Nevertheless, IBP relies on the assumption that datapoints (dishes) in a single string are exchangeable; each dish is assumed to be equally desired by customers. Another drawback is that the number of parameters increase as the dataset gets large, but Bayesian nonparameteric methods generally have this problem [9].

2 Code Structure and Simulated Data

To implement the linear-Gaussian binary latent feature model [5, 9] with IBP as the prior, a Gibbs sampler is used to generate the posterior samples, and the graphical model is shown in Figure 1. The IBP function is described in Section 1.1, and denoted as $Z \sim \mathrm{IBP}(\alpha)$, where Z is the binary matrix and $\alpha \sim Ga(1, 1)$.

2.1 Simulated Data for Likelihood

The likelihood involves simulated image data, and the variables are defined as follows:

- $N = 100$ is the number of images (customers or objects)
- $D = 6 \times 6 = 36$ is the length of vectors (dishes or features) for each image
- $K = 4$ is the number of basis images (latent or underlying variables)
- $\mathbf{X}$ represents the images generated by the K bases (each basis is present with probability 0.5), with white noises $\mathrm{Normal}(0, \sigma_X^2 = 0.5^2)$ added

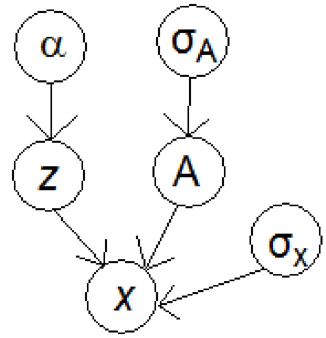

Figure 1: Graphical model for the linear-Gaussian binary latent feature model

The likelihood function is

$$\mathbf{X}|(\mathbf{Z}, \mathbf{A}, \sigma_\mathbf{X}) \sim \text{Normal}(\mathbf{ZA}, \Sigma_X = \sigma_X^2 \mathbf{I}) \tag{4}$$

$$P(\mathbf{X}|\mathbf{Z}, \sigma_X, \sigma_A) = \frac{1}{(2\pi)^{ND/2} \sigma_X^{(N-K)D} \sigma_A^{KD} |\mathbf{Z}^T\mathbf{Z} + \frac{\sigma_X^2}{\sigma_A^2}\mathbf{I}|^{D/2}} \exp\{-\frac{1}{2\sigma_X^2} \text{tr}(\mathbf{X}^T(\mathbf{I} - \mathbf{Z}(\mathbf{Z}^T\mathbf{Z} + \frac{\sigma_X^2}{\sigma_A^2}\mathbf{I})^{-1}\mathbf{Z}^T)\mathbf{X})\} \tag{5}$$

Each object i has a D-dimensional vector of properties named x_i, where:

- $x_i \sim \text{Normal}(\mathbf{z_i}\mathbf{A}, \Sigma_X = \sigma_X^2 \mathbf{I})$
- $\mathbf{z_i}$ is a K-dimensional binary vector (features)
- $\mathbf{A}$ is a $K \times D$ matrix of weights, with prior $\mathbf{A} \sim \text{Normal}(0, \sigma_A^2 \mathbf{I})$

The four basis images and an example of the simulated data are shown in Figure 2. Note that the likelihood involves close-to-zero probabilities, so the log likelihood is used in my code instead.

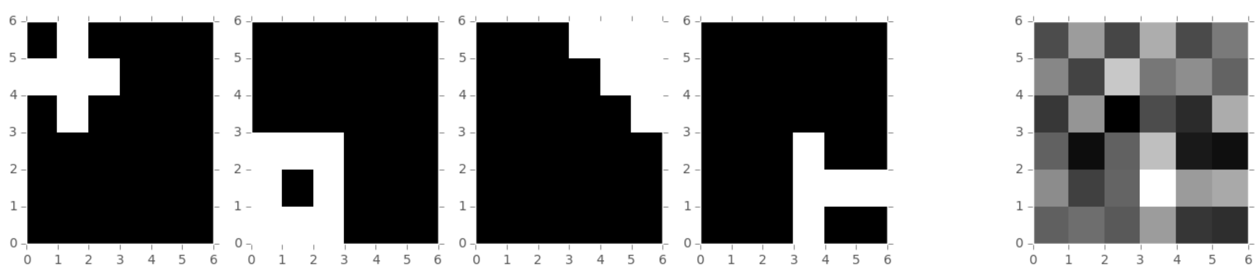

Figure 2: Simulated dataset: The four basis images (left) and an example image (right)

2.2 Gibbs Sampler for the Posterior Distribution

The full (posterior) conditional distribution is

$$P(z_{ik}|\mathbf{X}, \mathbf{Z_{-i,k}}, \sigma_X, \sigma_A) \propto P(\mathbf{X}|\mathbf{Z_{-i,k}}, \sigma_X, \sigma_A) P(z_{ik}|\mathbf{z_{-i,k}}) \tag{6}$$

When initializing the Gibbs sampler, set $\sigma_A = 1, \sigma_X = 1, \alpha \sim Ga(1, 1)$. Then the sampler does the following steps: (K in my code is denoted as K_+, to differentiate it from the true value.

1. Generate $P(z_{ik}|\mathbf{X}, \mathbf{Z_{-i,k}}, \sigma_X, \sigma_A)$ using the full conditional distribution

   (a) Remove singular features (at most one object has it); decrease K_+ by 1 for each feature removed

   (b) Determine each z_{ik} to be 0 or 1 by Metropolis

   (c) Add new features from Pois($\frac{\alpha}{i}$)

2. Sample $\sigma_X^* = \sigma_X + \epsilon$, where $\epsilon \sim$ Unif$(-0.05, 0.05)$, and accept σ_X^* by Metropolis

3. Sample $\sigma_A^* = \sigma_A + \epsilon$, where $\epsilon \sim$ Unif$(-0.05, 0.05)$, and accept σ_A^* by Metropolis

4. Generate $\alpha|Z \sim Ga(1 + K_+, 1 + \sum_{i=1}^{N} H_i)$, where K_+ is the number of features with $m_k > 0$

The Metropolis part for σ_A is demonstrated as follows (similar case for σ_X):

- Genenerate a candidate value $\sigma_A^* = \sigma_A + \epsilon$, with $\epsilon \sim$ Unif$(-0.05, 0.05)$
- Generate a random number $r \sim$ Unif$(0, 1)$
- Accept σ_A^* if $r < \min\{1, \dfrac{P(\sigma_A^*|\mathbf{Z}, \mathbf{X}, \sigma_X)}{P(\sigma_A|\mathbf{Z}, \mathbf{X}, \sigma_X)}\}$, where σ_A is the current value

The candidate value σ_A^* is always accepted when the likelihood ratio $\frac{P(\sigma_A^*|\mathbf{Z},\mathbf{X},\sigma_X)}{P(\sigma_A|\mathbf{Z},\mathbf{X},\sigma_X)}$ is larger than 1, i.e. $P(\sigma_A^*|\mathbf{Z}, \mathbf{X}, \sigma_X) > P(\sigma_A|\mathbf{Z}, \mathbf{X}, \sigma_X)$. Nevertheless, when the likelihood ratio is less than 1, there is still a non-zero probability to accept σ_A^*, so the sampler can "move forward". Note that in my code, the log likelihoods are used in the following way:

$$\min\{1, \frac{P(\sigma_A^*|\mathbf{Z}, \mathbf{X}, \sigma_X)}{P(\sigma_A|\mathbf{Z}, \mathbf{X}, \sigma_X)}\} = \exp(\min\{0, \log(P(\sigma_A^*|\mathbf{Z}, \mathbf{X}, \sigma_X)) - \log(P(\sigma_A|\mathbf{Z}, \mathbf{X}, \sigma_X))\}) \tag{7}$$

3 Algorithm Output and Testing

My implementation of the linear-Gaussian binary latent feature model with the IBP prior generates the results in images and traceplots. The simulated dataset contains four latent features (see Figure 2), but my code reveals five latent features in Figure 4, three of which are the linear combinations of two latent features. The traceplots in Figure 3 show my Gibbs sampler is converging: K_+ fluctuates between 5 and 8; the IBP parameter α is within [0.5,1.5]; σ_X converges to the true value 0.5; σ_A oscillates around 0.4. A total of 1000 Gibbs sampling iterations were performed, but the values started to converge at the 100th iteration.

I also performed in-line code testing by various methods. In the IBP prior, the `assert` command is used to verify $\frac{m_k}{i}$ to be a probability, i.e. between 0 and 1 – because the ith ($i > 2$) customer takes dish k with probability $\frac{m_k}{i}$ in the IBP algorithm. In many parts of my code, I used `np.dot` from `numpy` to do matrix multiplications even when the size of matrices is small, instead of multiplying each column/row one by one. In this way, the dimensions in matrix multiplications are assured to match each other.

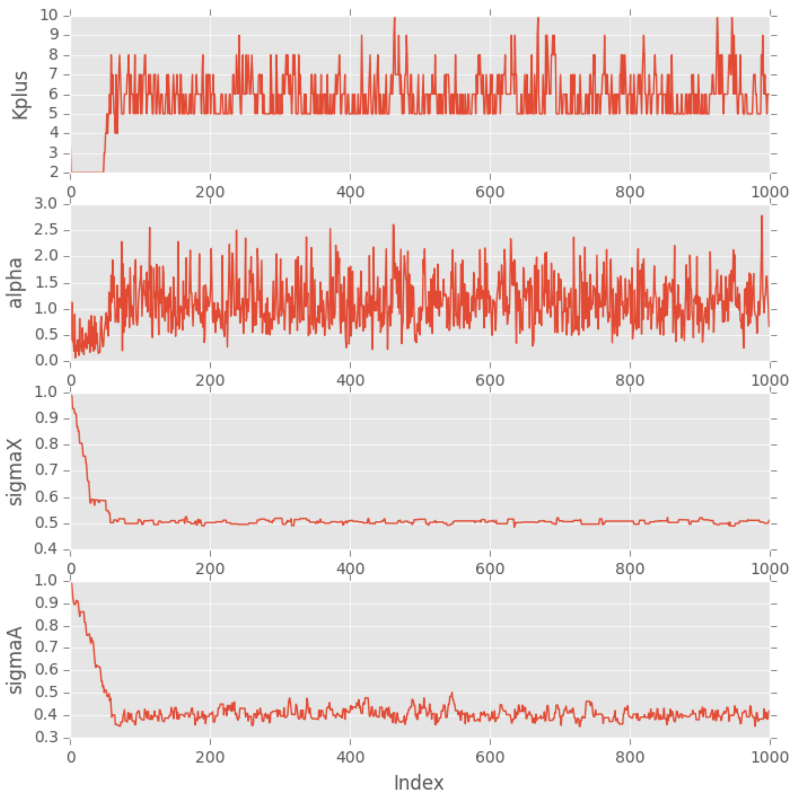

Figure 3: The traceplots for $K_+, \alpha, \sigma_X, \sigma_A$

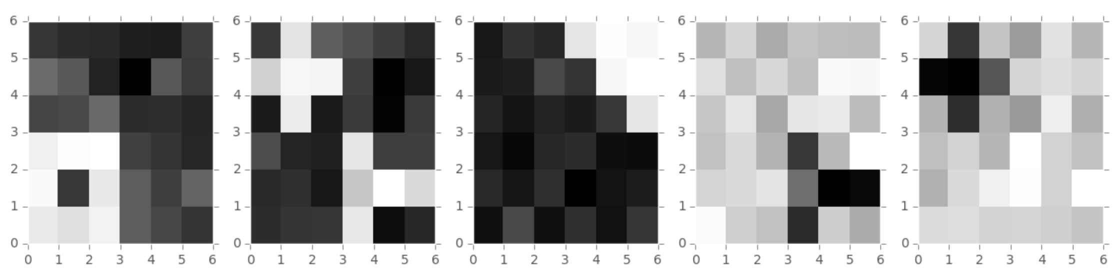

Figure 4: Simulated dataset: My results

4 Optimization

In this section, I performed optimization on the IBP linear-Gaussian model. Before optimizing the code, I performed profiling to identify the bottlenecks; the code structure can be visualized as a tree in Figure 5. In one Gibbs sampling iteration, generating $Z|\alpha$ and sampling σ_X, σ_A are performed once each. In generating $Z|\alpha$, sampling dishes from K_+ and sampling new dishes are done for each customer (image or object), so they are each performed $N = 100$ times. In sampling dishes from K_+, calculation refers to the process of sampling the posterior distribution of $Z|K_+$, and initialization is the part of removing features which are all zero. Both calculation and initialization are performed $N \times K_+ \sim 500$ times for each iteration. The calculation in sampling from K_+ for generating $Z|\alpha$ accounts for 70% of the time, approximately 1.4 seconds per iteration because it involves matrix inversion and likelihood calculation. Table 1 shows the profiling results for my initial code.

To optimize the code, redundant calculations are removed first, and this version is named as "usable". When generating $Z|\alpha$, the inverted matrix $\mathbf{M} = (\mathbf{Z}^T\mathbf{Z} + \frac{\sigma_X^2}{\sigma_A^2}\mathbf{I})^{-1}$ is only calculated directly before the likelihood computation, so more than $N = 100$ matrix inversions can be removed. The profiling results are shown in Table 2. The "usable" version code can also be cythonized (converted from Python to C), and Table 3 is a summary of profiling results, but the Cythonized version only improved the speed about 0.5%.

I also attempted to alleviate the bottleneck of calculating $\mathbf{M}$ by using Equations (51)-(54) in Griffiths' and Ghahramani's paper [6], but this did not work because K_+ got stuck at 2. Theoretically, this method below allows us to effeciently compute M when only one $\mathbf{z_i}$ is changed:

$$\text{Define } \mathbf{M}_{-i} = (\sum_{j\neq i} \mathbf{z}_j^T \mathbf{z}_j + \frac{\sigma_X^2}{\sigma_A^2}\mathbf{I})^{-1} \tag{8}$$

$$\mathbf{M}_{-i} = (\mathbf{M}^{-1} - \mathbf{z}_i^T\mathbf{z}_i)^{-1} = \mathbf{M} - \frac{\mathbf{M}\mathbf{z}_i^T\mathbf{z}_i\mathbf{M}}{\mathbf{z}_i\mathbf{M}\mathbf{z}_i^T - 1} \tag{9}$$

$$\mathbf{M} = (\mathbf{M}_{-i}^{-1} - \mathbf{z}_i^T\mathbf{z}_i)^{-1} = \mathbf{M}_{-i} - \frac{\mathbf{M}_{-i}\mathbf{z}_i^T\mathbf{z}_i\mathbf{M}_{-i}}{\mathbf{z}_i\mathbf{M}_{-i}\mathbf{z}_i^T + 1} \tag{10}$$

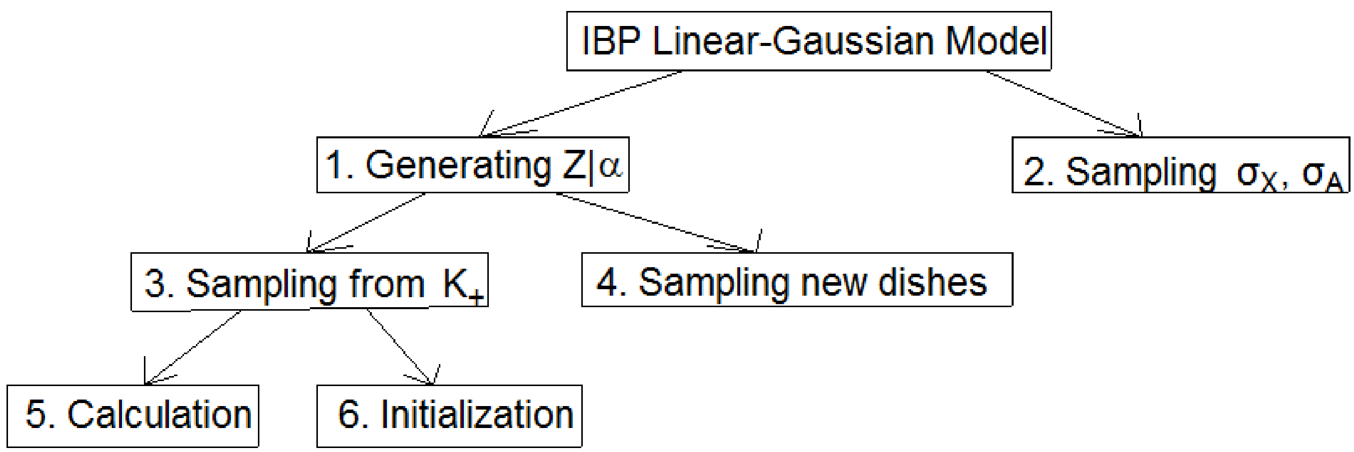

Figure 5: IBP code structure for profiling

	Time (seconds)/action	Times performed	Total time (seconds)
Generating Z given alpha	2.010423	1	2.010423
Sampling sigmaX, sigmaA	0.004174	1	0.004174
Sampling from K+	0.014984	100	1.498403
Sampling new dishes	0.005013	100	0.501266
Calculation	0.002762	500	1.380943
Initialization	0.000003	500	0.001453

Table 1: Naive code: Profiling results per iteration

	Time (seconds)/action	Times performed	Total time (seconds)
Generating Z given alpha	1.962989	1	1.962989
Sampling sigmaX, sigmaA	0.004164	1	0.004164
Sampling from K+	0.014690	100	1.469040
Sampling new dishes	0.004936	100	0.493605
Calculation	0.002529	500	1.264704
Initialization	0.000003	500	0.001566

Table 2: Usable code: Profiling results per iteration

	Time (seconds)/action	Times performed	Total time (seconds)
Generating Z given alpha	1.951373	1	1.951373
Sampling sigmaX, sigmaA	0.003792	1	0.003792
Sampling from K+	0.014627	100	1.462701
Sampling new dishes	0.004883	100	0.488332
Calculation	0.002317	500	1.158574
Initialization	0.000003	500	0.001499

Table 3: Cythonized code: Profiling results per iteration

5 Comparative Analysis

Start the comparative analysis.

5.1 Chinese Restaurant Process

Figure refers to Gershman's and Blei's paper [3].

5.2 Another Matlab Version Online

5.3 Another Python Version Online

6 Results

7 Conclusion

Write your conclusion here

References

[1] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[2] Finale Doshi, Kurt Tadayuki Miller, Jurgen Van Gael, and Yee Whye Teh. Variational inference for the indian buffet process. In *Proceedings of the 12th International Conference on Artificial Intelligence and Statistics*, volume 5, 2008.

[3] Samuel J Gershman and David M Blei. A tutorial on bayesian nonparametric models. *Journal of Mathematical Psychology*, 56(1):1–12, 2012.

[4] DMBTL Griffiths and MIJJB Tenenbaum. Hierarchical topic models and the nested chinese restaurant process. *Advances in neural information processing systems*, 16:17, 2004.

[5] Thomas Griffiths and Zoubin Ghahramani. Infinite latent feature models and the indian buffet process. In *Advances in Neural Information Processing Systems*, volume 18. NIPS Proceedings, 2005.

[6] Thomas Griffiths and Zoubin Ghahramani. Infinite latent feature models and the indian buffet process. Technical report, Gatsby Computational Neuroscience Unit, 2005.

[7] Sinead Williamson, Chong Wang, Katherine Heller, and David Blei. Focused topic models. In *NIPS workshop on Applications for Topic Models: Text and Beyond, Whistler, Canada*, 2009.

[8] Ilker Yildirim. Indian buffet process – sample code. `http://www.mit.edu/~ilkery/`. Online; accessed 2015.

[9] Ilker Yildirim. Bayesian statistics: Indian buffet process. `http://www.bcs.rochester.edu/people/robbie/jacobslab/cheat_sheet/IndianBuffetProcess.pdf`, 2012. Online; accessed 2015.

[10] Ilker Yildirim and Robert A Jacobs. A bayesian nonparametric approach to multisensory perception. In *The annual meeting of the Cognitive Science society*, pages 2633–2638, 2010.