Structured Variational Autoencoders for the Beta-Bernoulli Process

Rachit Singh* **Jeffrey Ling*** **Finale Doshi-Velez**
Harvard University
{rachitsingh@college,jling@college,finale@seas}.harvard.edu

Abstract

Beta-Bernoulli processes, also known as Indian buffet processes, are nonparametric priors that allow generative models to automatically infer the number of features in datasets. However, inference for these models proves to be challenging, often relying on specific forms of the likelihood for computational tractability.

We propose to amortize inference using a variational autoencoder trained via gradient descent, allowing for arbitrary likelihood models. Our model extends previously considered mean field variational methods with a structured posterior and new developments in the training of discrete variable VAEs. We experimentally demonstrate a Beta-Bernoulli process VAE that learns decomposable latent features and allows for scalable inference of arbitrary likelihoods on large datasets.

1 Introduction

Deep generative models are becoming increasingly feasible for performing unsupervised learning tasks [Kingma and Welling, 2013, Gregor et al., 2015, Ranganath et al., 2014]. One limitation of many models is their pre-defined capacity, or that latent representations must have a fixed dimension independent of the complexity of data.

The field of Bayesian nonparametrics seeks to address this issue. By imposing a prior on an infinite dimensional latent variable, a model has the flexibility to learn its latent dimensionality. Such methods include the Dirichlet process [Neal, 2000] and the Beta-Bernoulli process, also known as the Indian buffet process (IBP) [Griffiths and Ghahramani, 2011, Thibaux and Jordan, 2007]. As a prior over infinite dimensional binary matrices, the Beta-Bernoulli process provides a principled and flexible way to estimate latent features from data. However, inference for the Beta-Bernoulli process tends to be focused on simple likelihoods and small datasets.

In this work, we overcome these limitations using developments in gradient-based optimization in variational inference [Kingma and Welling, 2013, Hoffman et al., 2013, Hoffman and Blei, 2015]. We propose a deep generative model with a nonparametric prior and train it as a variational autoencoder for the IBP. In addition, we show that a structured variational posterior improves upon the mean field assumption first explored by Chatzis [2014]. Finally, we leverage recent developments in the training of VAEs to improve the quality of inference.

2 Background

The Indian buffet process (IBP) defines a distribution on latent feature allocations $\mathbf{Z} \in \{0,1\}^{N \times K^+}$, where $z_{n,k}$ is 1 if feature k is on for the nth data point and 0 otherwise [Griffiths and Ghahramani, 2011]. It can be characterized using the *stick breaking process*, which first samples $\nu_k \sim \text{Beta}(\alpha, 1)$,

*Equal contribution.

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

and then samples the elements of each row as $z_{n,k} \sim \text{Bern}(\pi_k); \pi_k = \prod_{j=1}^{k} \nu_j$ [Teh et al., 2007]. Here, α is a hyperparameter that represents the expected number of features of each data point.

Given this prior over feature allocations $\mathbf{Z}$, we can specify a generative model for our data $\mathbf{X}$ with a likelihood $p(\mathbf{X}|\mathbf{Z})$. If our data is $\{\mathbf{x}_n\}_{n=1}^N$ for $\mathbf{x}_i \in \mathbb{R}^D$, then we have

$$\mathbf{Z}, \boldsymbol{\nu} \sim \text{IBP}(\alpha); \quad \mathbf{A}_n \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_{K^+}); \quad \mathbf{x}_n \sim p_\theta(\mathbf{x}_n|\mathbf{Z}_n \odot \mathbf{A}_n)$$

where $\odot$ is the Hadamard product. Here, $p_\theta(\mathbf{x}|\mathbf{Z}, \mathbf{A})$ is a likelihood defined by an arbitrary function with parameters θ. A well studied example is the linear Gaussian model, where $p(\mathbf{x}|\mathbf{Z}, \mathbf{A}) \sim \mathcal{N}(\mathbf{ZA}, \sigma^2)$ [Griffiths and Ghahramani, 2011] (with a global $\mathbf{A}$). For most of this work, we consider arbitrary likelihood models by feeding $\mathbf{Z}_n \odot \mathbf{A}_n$ into a neural network. Unlike in previous methods, our generative model includes non-exponential family likelihoods, making inference more difficult.

3 Related Work

The key difficulty in using the IBP is that, regardless of the generative model, inference and training can be complicated. Gibbs sampling [Griffiths and Ghahramani, 2011, Teh et al., 2007] can be slow for large datasets, while many variational methods [Doshi et al., 2009, Shah et al., 2015] often require strong relaxations (e.g. the mean field assumption, conjugate likelihoods) for tractability.

(Structured) stochastic variational inference ((S)SVI) [Hoffman et al., 2013, Hoffman and Blei, 2015] and variational autoencoders (VAEs) [Kingma and Welling, 2013] provide a general solution to scale up inference using minibatch ascent. VAEs with nonparametric priors have only been rarely explored due to the prevalence of nonreparametrizable variables in nonparametric distributions. Nalisnick and Smyth [2017] propose a deep generative model with the stick breaking weights of a Dirichlet process as the latent variable, avoiding discrete backpropagation. Chatzis [2014] propose a model that we use as our primary benchmark: a deep VAE with a latent Beta-Bernoulli process, trained with black box variational inference [Ranganath et al., 2014]. In this work, we improve upon the mean field assumption of Chatzis [2014] with a structured variational posterior, and we also leverage recent developments in training discrete nodes, thus pushing the state of the art for IBP inference.

4 Inference

In this section, we consider methods to learn the posterior $p(\mathbf{Z}, \mathbf{A}, \boldsymbol{\nu}|\mathbf{X})$ using variational inference: (1) mean field, as utilized by Chatzis [2014], and (2) structured, where we keep dependencies between local and global variables in the variational posterior.

Note that our latent variables are both the global variables $\boldsymbol{\nu} = \{\nu_k\}_{k=1}^\infty$ (in bijection to weights π_k) and the local hidden variables $\boldsymbol{\psi}_n = \{\mathbf{Z}_n, \mathbf{A}_n\}$ with $\mathbf{Z}_n \in \{0,1\}^{K^+}$, $\mathbf{A}_n \in \mathbb{R}^{K^+}$, in addition to likelihood weights θ. Hence, our variational posterior is of the form $q(\boldsymbol{\nu}, \{\boldsymbol{\psi}_n\}_{n=1}^N|\mathbf{X})$. While the true posterior is over an infinite latent variable, we assume a truncated posterior which has support only over finitely sized matrices, as in Doshi et al. [2009], Blei and Jordan [2004]. Throughout, we derive point estimates of θ with gradient descent on our objective.

4.1 Mean Field Inference Networks

The simplest variational posterior makes the mean-field approximation, in which we assume that q factors: $q(\boldsymbol{\nu}, \{\boldsymbol{\psi}_n\}_{n=1}^N) = \prod_{k=1}^K q(\boldsymbol{\nu}_k) \prod_{n=1}^N q(z_{n,k}) q(\mathbf{A}_n)$, where K is a truncation hyperparameter. Chatzis [2014] explored the mean-field approximation and used the following approximation: $q(\nu_k) = \text{Beta}(\nu_k|a_k(\mathbf{x}_n), b_k(\mathbf{x}_n)); q(z_{n,k}) = \text{Bern}(z_{n,k}|\pi_k(\mathbf{x}_n)); q(\mathbf{A}_n) = \mathcal{N}(\mathbf{A}_n|\mu(\mathbf{x}_n), \text{diag}(\sigma^2(\mathbf{x}_n)))$ where $a_k, b_k, \pi_k, \mu, \sigma^2$ are the outputs of neural networks which have input $\mathbf{x}_n$.

Chatzis [2014] gives the following expression for the negative ELBO or loss $\mathcal{L}$:

$$\mathcal{L} = -\mathbb{E}_q \left[\log p(\mathbf{x}|\boldsymbol{\psi})\right] + \text{KL}(q(\mathbf{Z}|\boldsymbol{\nu}) \parallel p(\mathbf{Z}|\boldsymbol{\nu})) + \text{KL}(q(\boldsymbol{\nu}) \parallel p(\boldsymbol{\nu})) + \text{KL}(q(\mathbf{A}) \parallel p(\mathbf{A}))$$

We refer to this model as MF-IBP. Note that when we train this model with the ELBO, the ν_k random variables from q do not affect the generation of $\mathbf{x}_n$, but only $\mathbb{E}_q[\log p(\mathbf{Z}|\boldsymbol{\nu})]$ and $\text{KL}(q(\boldsymbol{\nu}) \parallel p(\boldsymbol{\nu}))$. We find in practice that the first term has very little effect, and the second term encourages the posterior to approach the prior, which is a $\text{Beta}(\alpha, 1)$ distribution. So, in practice, the functions $a_k(\mathbf{x}), b_k(\mathbf{x})$ degenerate to constants near $\alpha, 1$.

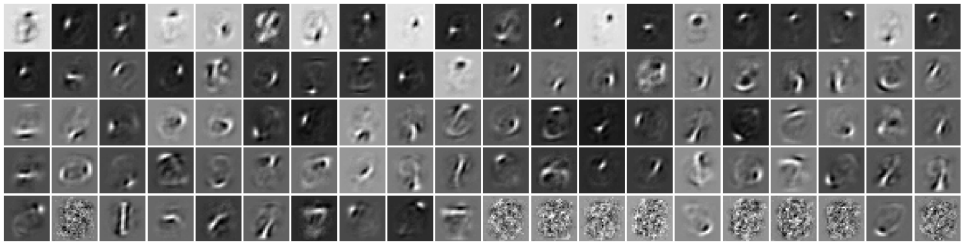

Figure 1: All features learned from MNIST in a linear Gaussian generative model. Note that many of the 'noise' features are never activated.

4.2 Extension: Structured Inference Networks

We can factor our variational posterior into a hierarchical distribution over global and local variables. We view the ν_k variables as being global, since they are shared between data points, and the $z_{n,k}$ variables as local. So, we now have $q(\boldsymbol{\nu}, \{\boldsymbol{\psi}_n\}_{n=1}^N) = \prod_{k=1}^K q(\nu_k) \prod_{n=1}^N q(z_{n,k} \mid \nu_k) q(\mathbf{A}_n)$ with $q(\nu_k) = \text{Beta}(\nu_k | a_k, b_k)$; $\quad q(z_{n,k} \mid \nu_k) = \text{Bern}(z_{n,k} | \pi_k)$; $\quad \pi_k := \prod_{j=1}^k \nu_j$ where a_k, b_k are parameters to be learned. We call this model/inference scheme S-IBP.

We maximize using SSVI [Hoffman and Blei, 2015], giving the following loss $\mathcal{L}$:

$$\text{KL}(q(\boldsymbol{\nu}) \parallel p(\boldsymbol{\nu})) + \sum_{n=1}^N -\mathbb{E}_q[\log p(\mathbf{x}_n | \mathbf{Z}_n, \mathbf{A}_n)] + \text{KL}(q(\mathbf{A}_n) \parallel p(\mathbf{A}_n)) + \text{KL}(q(\mathbf{Z}_n | \boldsymbol{\nu}) \parallel p(\mathbf{Z}_n | \boldsymbol{\nu}))$$

We approximate the expectations by taking a sample $\boldsymbol{\nu} \sim q(\boldsymbol{\nu})$, and then sample from $\mathbf{Z}_n \sim q(\mathbf{Z}_n \mid \boldsymbol{\nu})$ (which is implicitly a function of $\mathbf{X}$ from the inference net).

4.3 Training and Evaluation

For VAEs, it usually suffices to specify the variational posterior and the generative model, and automatic differentiation will take care of the rest. However, our model has Bernoulli and Beta latent variables, neither of which permit easy reparametrization. We consider two solutions: black box variational inference (BBVI) [Ranganath et al., 2014] and biased approximations. For the biased option, we approximate Bernoulli r.v.s with Gumbel softmax variables [Maddison et al., 2017, Jang et al., 2017] during training, and Beta r.v.s with Kumaraswamy r.v.s [Nalisnick and Smyth, 2017]. See the Appendix for details.

The importance weighted autoencoder bound (IWAE) [Burda et al., 2015] is known to be a stronger bound on the log marginal, so we use it as our metric. Since we have global latent variables in our S-IBP models, we must be careful to sample globally a set of $\boldsymbol{\nu}_k$ before calculating the IWAE estimate using subsampling. See the Appendix for details.

5 Experiments

5.1 Shallow Generative Model

First, we run a linear generative model with S-IBP, (i.e., the likelihood is $p(\mathbf{x}_n | \mathbf{Z}_n) = \mathbf{Z}_n \mathbf{A}$ with $\mathbf{A}$ learned parameters rather than sampled as in the standard linear Gaussian model) on the MNIST dataset to demonstrate the kinds of learned features that we can recover.

We visualize the rows of $\mathbf{A}$ learned by the model in 100 epochs. We use a truncation of 100 for our variational posterior, and Figure 1 shows all 100 features learned, sorted by IBP prior weight in row major order. Note that the model has automatically learned stroke-like features, and made use of the IBP truncation by leaving some features as noise.

5.2 Deep Generative Model

We experiment with training deep generative models on MNIST and Omniglot [Lake et al., 2015], two large scale datasets. We use the fixed binarization of Larochelle and Murray [2011] for MNIST.

	MNIST IWAE		Omniglot IWAE	
Model	Train	Test	Train	Test
MF-IBP BBVI	102.6	104.5	129.4	134.5
MF-IBP Gumbel	94.2	96.4	125.0	129.5
S-IBP BBVI	93.8	96.2	115.2	124.5
S-IBP Gumbel	81.7	**86.5**	101.4	**113.0**

Table 1: MNIST and Omniglot IWAE test results. K (the truncation level) was set to 100, and $\alpha = 10$.

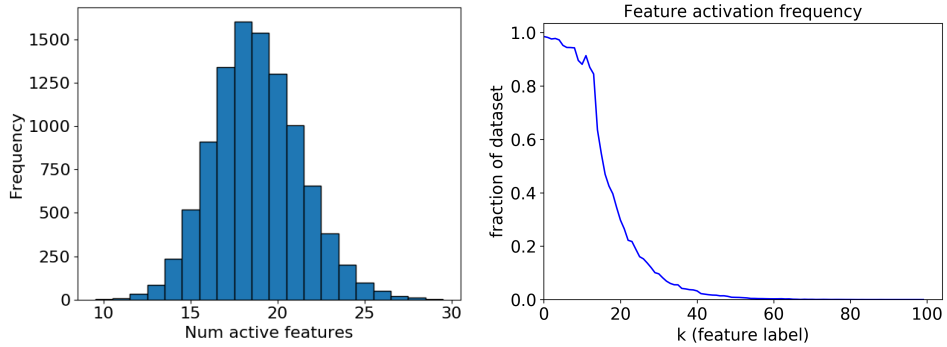

Figure 2: Trained S-IBP Gumbel model on MNIST with $\alpha = 10$, KL for ν weighted by 10^4 during training. Left: Distribution of number of features used per data point. The average data point uses about 18 features, a bit higher than the prior. Right: Number of test data that use each feature. Note the long tail of feature activations.

We construct a generative model whose likelihood is a multi-layer perceptron with a single hidden layer of 500 units, such that the neural network parametrizes a function $\log p(\mathbf{x}|\mathbf{Z}, \mathbf{A}) = f_\theta(\mathbf{Z}, \mathbf{A})$. Similarly, we construct a fully connected inference network with a single hidden layer of 500 units.

We consider 4 models as the combination of two choices: (1) either the mean field approximation of Chatzis [2014] (MF-IBP) or the structured approach in section 4.2, and (2) the gradient estimation technique for training the Bernoulli hidden variables, either BBVI or the Gumbel softmax method. For both structured posterior approaches, we use the Kumaraswamy variational approximation for the Beta random variables. See Appendix for training and hyperparameter details.

Our models are implemented in PyTorch, and are open source[2]. Table 1 shows our results, evaluated with IWAE. We see that the structured posterior outperforms the corresponding mean-field method.

6 Discussion

We take a deep generative model (S-IBP Gumbel) trained on MNIST and analyze its inference properties with respect to the IBP prior. Figure 2 illustrates the usage of features, where we trained with the KL for ν weighted by 10^4 to encourage adhering to the prior (this did not affect the score much). We observe that although we set $\alpha = 10$ to indicate our prior for about 10 features, the model prefers to use more to reduce its reconstruction loss. We see a "cliff" of features around 20, which is more than we would like for an $\text{IBP}(\alpha)$ with $\alpha = 10$. This shows that our model is not being regularized sufficiently by the IBP prior, and that it prefers to use more features and the power of the deep likelihood function to reconstruct the image. Future work is needed to improve feature sparsity.

7 Conclusion

In this paper, we presented several forms of Indian buffet process VAEs. We showed empirically that choosing a structured posterior allows for a better optimum, and that the learned features are interesting. Future work will involve applying the latent feature discovery to more interesting generative problems.

[2] `https://github.com/rachtsingh/ibp_vae/`

References

David M. Blei and Michael I. Jordan. Variational methods for the dirichlet process. In *ICML*, 2004.

Yuri Burda, Roger B. Grosse, and Ruslan Salakhutdinov. Importance weighted autoencoders. *ICLR*, 2015.

Sotirios P Chatzis. Indian buffet process deep generative models. *arXiv preprint arXiv:1402.3427*, 2014.

Finale Doshi, Kurt Miller, Jurgen Van Gael, and Yee Whye Teh. Variational inference for the indian buffet process. In *AISTATS 2009*, pages 137–144, 2009.

Emily B Fox, Michael C Hughes, Erik B Sudderth, Michael I Jordan, et al. Joint modeling of multiple time series via the beta process with application to motion capture segmentation. *The Annals of Applied Statistics*, 8(3):1281–1313, 2014.

Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Jimenez Rezende, and Daan Wierstra. Draw: A recurrent neural network for image generation. *arXiv preprint arXiv:1502.04623*, 2015.

Thomas L. Griffiths and Zoubin Ghahramani. The indian buffet process: An introduction and review. *Journal of Machine Learning Research*, 12:1185–1224, 2011.

Matthew Hoffman and David Blei. Stochastic structured variational inference. In *Artificial Intelligence and Statistics*, pages 361–369, 2015.

Matthew D Hoffman, David M Blei, Chong Wang, and John Paisley. Stochastic variational inference. *The Journal of Machine Learning Research*, 14(1):1303–1347, 2013.

Michael Hughes, Dae Il Kim, and Erik Sudderth. Reliable and Scalable Variational Inference for the Hierarchical Dirichlet Process. In Guy Lebanon and S. V. N. Vishwanathan, editors, *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, volume 38 of *Proceedings of Machine Learning Research*, pages 370–378, San Diego, California, USA, 09–12 May 2015. PMLR.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical Reparametrization with Gumbel-Softmax. *ICLR*, 2017.

Diederik P. Kingma and Jimmy Lei Ba. Adam: a Method for Stochastic Optimization. *International Conference on Learning Representations 2015*, pages 1–15, 2015.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Brendan Lake, Ruslan Salakhutdinov, and Joshua B. Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332—-1338, 2015.

Hugo Larochelle and Iain Murray. The neural autoregressive distribution estimator. In *AISTATS 2011*, pages 29–37, 2011.

Chris J Maddison, Andriy Mnih, and Yee Whye Teh. the Concrete Distribution: a Continuous Relaxation of Discrete Random Variables. *ICLR*, pages 1–17, 2017.

Erick Nalisnick and Padhraic Smyth. Stick-Breaking Variational Autoencoders. *ICLR*, 2017.

Radford M Neal. Markov chain sampling methods for dirichlet process mixture models. *Journal of computational and graphical statistics*, 9(2):249–265, 2000.

Rajesh Ranganath, Sean Gerrish, and David M. Blei. Black box variational inference. In *AISTATS 2014*, pages 814–822, 2014.

Amar Shah, David Knowles, and Zoubin Ghahramani. An Empirical Study of Stochastic Variational Inference Algorithms for the Beta Bernoulli Process. *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, 37:1594–1603, 2015.

Yee Whye Teh, Dilan Görür, and Zoubin Ghahramani. Stick-breaking construction for the indian buffet process. In *AISTATS 2007*, pages 556–563, 2007.

Romain Thibaux and Michael I. Jordan. Hierarchical beta processes and the indian buffet process. In *AISTATS 2007*, pages 564–571, 2007.

A Appendix

A.1 Inference

Gumbel softmax The Gumbel softmax, or Concrete_λ, variable is a continuous approximation of a Bernoulli variable used during training to get (biased) estimates of the gradient [Jang et al., 2017, Maddison et al., 2017]. We follow the notation of Maddison et al. [2017].

We relax the discrete KL between two Bernoullis into a KL divergence between two Concrete_λ variables. If the original ELBO is

$$\mathbb{E}_{z\sim q_\alpha(z|x)}\left[\log \frac{p_\theta(x|z)p_a(z)}{q_\alpha(z|x)}\right] \tag{1}$$

where z is sampled discretely from $\text{Bern}((\alpha))$, and α is the logit probability for the variational posterior. We relax this by replacing discrete samples with Concrete samples with different temperatures for the priors:

$$\mathbb{E}_{z\sim q_{\alpha,\lambda}(z|x)}\left[\log \frac{p_\theta(x|z)p_{a,\lambda_{\text{prior}}}(z)}{q_{\alpha,\lambda}(z|x)}\right] \tag{2}$$

Here, $\lambda, \lambda_{\text{prior}}$ are temperature hyperparameters, and a is the logit probability for the prior (for us, π_k). This expectation is computed using a single sample.

$q_{\alpha,\lambda}(z|x)$ and $p_{a,\lambda_{\text{prior}}}(z)$ are Concrete distributions, where the density for binary random variables is

$$p_{\alpha,\lambda}(x) = \frac{\lambda\alpha x^{-\lambda-1}(1-x)^{-\lambda-1}}{(\alpha x^{-\lambda}+(1-x)^{-\lambda})^2} \tag{3}$$

Alternatively, we have the representation $X = \sigma((L+\log\alpha)/\lambda)$ where L is Logistic.

At train time, we sample from the Concrete density to obtain the ELBO. At test time, we sample according to the Bernoulli.

Kumaraswamy The Kumaraswamy distribution with parameters a, b has density function

$$p(x; a, b) = abx^{a-1}(1-x^a)^{b-1}$$

and approximates a Beta density. In particular, in the case of $a = 1$ or $b = 1$ the Kumaraswamy and Beta distributions are identical. We have a reparametrization as

$$x \sim (1-u^{1/b})^{1/a}$$

where $u \sim \text{Unif}(0, 1)$. Nalisnick and Smyth [2017] use these as a variational posterior instead of Beta variables in order to have full differentiability of their model, and we do the same in our S-IBP model.

The KL divergence between Kumaraswamy and Beta distribution is given by for $q(v_k) \sim \text{Kumaraswamy}(a, b)$ and $p(v_k) \sim \text{Beta}(\alpha, \beta)$:

$$\begin{aligned}\text{KL}(q(v_k) \parallel p(v_k)) = \frac{a-\alpha}{a}\left(-\gamma - \Psi(b) - \frac{1}{b}\right) + \\ + \log ab + \log B(\alpha,\beta) - \frac{b-1}{b} + \\ + (\beta-1)b\sum_{m=1}^{\infty}\frac{1}{m+ab}B\left(\frac{m}{a}, b\right)\end{aligned} \tag{4}$$

γ is the Euler constant, Ψ is the digamma function, B is the Beta function [Nalisnick and Smyth, 2017]. In practice, we take a finite approximation of the infinite sum.

A.2 Modification of the IWAE bound

The difference is that our joint and variational approximations factor only once we've conditioned on the value of $\boldsymbol{\nu}$:

$$\log p(\mathbf{x}) = \log \int_{\boldsymbol{\psi},\boldsymbol{\nu}} p(\mathbf{x},\boldsymbol{\nu},\boldsymbol{\psi})\frac{q(\boldsymbol{\nu},\boldsymbol{\psi})}{q(\boldsymbol{\nu},\boldsymbol{\psi})}\,d\boldsymbol{\nu}\,d\boldsymbol{\psi} \geq \mathbb{E}_q\left[\log \frac{1}{k}\sum_{l=1}^{k}\frac{p(\boldsymbol{\nu}_k)}{q(\boldsymbol{\nu}_k)}\prod_{i=1}^{N}\frac{p(\mathbf{x}_i,\boldsymbol{\psi}_{i,k}|\boldsymbol{\nu}_k)}{q(\boldsymbol{\psi}_{i,k}|\boldsymbol{\nu}_k)}\right]$$

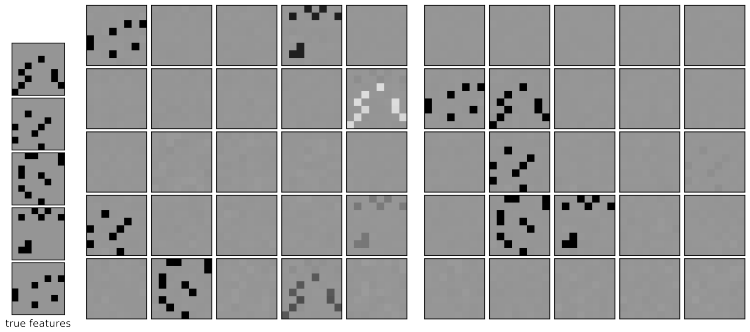

Figure 3: The global features $\mathbf{A}$ learned by the IBP models. On the left is the true matrix of features $\mathbf{A}$. Black corresponds to a value of 1 (or higher), grey corresponds to 0, and white corresponds to a value of -1 (or lower).

To calculate this via sub-sampling, we sample the $\boldsymbol{\nu}$ k times, and then iterate over the dataset, sampling $\mathbf{z}_n$ from the conditional posterior $q(\mathbf{z}_n|\boldsymbol{\nu})$, and computing the overall log-marginal bound. We average to get the per data numbers in Table 1.

A.3 Training Details

For all our models, we perform gradient descent using Adam [Kingma and Ba, 2015] with learning rate 0.001, set the hidden layer size of all neural networks to 500, set the truncation of our variational posterior to $K = 100$, and set $\alpha = 10$, which regularizes the model to use an average of 10 features for each data point. We tune hyperparameters λ and λ_{prior} using grid search. For the models that use BBVI, we use 10 samples of the gradient to estimate the control variate.

For MNIST, MF-IBP Gumbel and S-IBP Gumbel were run with $\lambda_{prior} = 0.25$, $\lambda = 0.3$. For Omniglot, MF-IBP Gumbel and S-IBP Gumbel were run with $\lambda_{prior} = 0.8$, $\lambda = 1$.

A.4 Synthetic Data

We generate a small synthetic dataset using randomly sampled features, and show that our main implementations are able to recover approximately the true number of features, and compare the run time of each approximate method. Specifically, we sample a global feature matrix $\mathbf{A} \in \mathbb{R}^{K \times D}$ with $K = 5, D = 64$, a feature allocation matrix $\mathbf{Z} \in \{0, 1\}^{N \times K}$, and our data $\mathbf{X} \sim \mathbf{ZA} + \epsilon$ where ϵ is Gaussian noise. Since we know the generative model for the data, we can verify our models and examine the effects of different inference methods.

Specifically, we use our implementation of the model of Chatzis [2014] (MF-IBP), and our model with a structured posterior and trained using the Concrete and Kumaraswamy distributions (S-IBP). We modify the models to remove the hidden variables $\mathbf{A}_n$, since the generative model only uses the latent matrix $\mathbf{Z}$ and a global matrix $\mathbf{A}$. We leave $\mathbf{A}$ as a parameter with a point estimate variational posterior, and a Gaussian prior.

In Figure 3 we have visualizations of the true features, the features learned by the MF-IBP (BBVI), and by the S-IBP (Concrete). Note that there's duplication of features and negation of features in the MF-IBP, which is a recurring problem in Indian buffet process inference. Also, because of the Gaussian prior on $\mathbf{A}$, many of the features are regularized down to 0. However, they are not eliminated from the model as they might be in other IBP formulations that can take advantage of split/merge steps [Fox et al., 2014, Hughes et al., 2015].

A.5 Reconstructions from Shallow Generative Model

In Figure 4, we examine a few digits and their reconstructions via inference by the IBP prior. We see that the features near the front of the IBP prior are used more often, and that features in the tail encode more information about individual digits. Unfortunately, the model does not learn sparsity well, likely due to the balance between minimizing the reconstruction loss and the KL between the IBP prior and the inferred feature distribution.

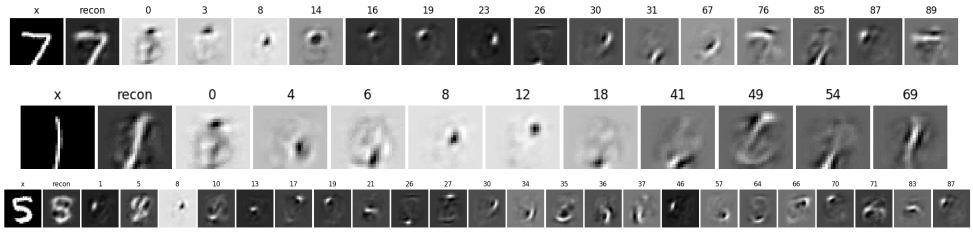

Figure 4: Reconstructions of MNIST digits, along with the numbered features used, via the shallow generative model.