Variational Russian Roulette for Deep Bayesian Nonparametrics

Kai Xu [1] **Akash Srivastava** [1 2] **Charles Sutton** [1 3 4]

Abstract

Bayesian nonparametric models provide a principled way to automatically adapt the complexity of a model to the amount of the data available, but computation in such models is difficult. Amortized variational approximations are appealing because of their computational efficiency, but current methods rely on a fixed finite truncation of the infinite model. This truncation level can be difficult to set, and also interacts poorly with amortized methods due to the over-pruning problem. Instead, we propose a new variational approximation, based on a method from statistical physics called Russian roulette sampling. This allows the variational distribution to adapt its complexity during inference, without relying on a fixed truncation level, and while still obtaining an unbiased estimate of the gradient of the original variational objective. We demonstrate this method on infinite sized variational auto-encoders using a Beta-Bernoulli (Indian buffet process) prior.

1. Introduction

A major challenge in unsupervised learning is to infer the complexity of the latent structure, such as the number of clusters or the size of a continuous representation, that is necessary to describe a data set. A principled method from statistics to choose the complexity of a model is provided by *Bayesian nonparametric* methods (Walker et al., 1999; Müller & Quintana, 2004; Orbanz & Teh, 2010; Gershman & Blei, 2012). Nonparametric methods allow for the size of the inferred model to automatically adapt to the amount of data, so that simpler models are preferred for smaller data sets, and more complex models are preferred for larger data sets. For example, a latent feature model with an Indian buffet process (IBP) prior is a representation learning method that infers a latent binary vector for each data point, where the number of binary features is chosen adaptively based on the data. Within machine learning, Bayesian nonparametric methods have been applied within models as diverse as clustering (Antoniak, 1974; Görür & Rasmussen, 2010; Teh et al., 2005), topic modeling (Teh et al., 2006), and infinite deep neural networks (Adams et al., 2010).

However, inference in Bayesian nonparametric models can be computationally challenging. Amortized variational methods (Kingma & Welling, 2013; Rezende et al., 2014; Ranganath et al., 2014a; Mnih & Gregor, 2014) are an appealing option, because they exploit the smoothing properties of deep neural networks to accelerate inference. For Bayesian nonparametric models, however, amortized inference is challenging because the dimensionality of the latent space is not fixed. Previous methods for variational inference in such models rely on a truncated approximation, which places an upper bound on the size of the latent space under the approximate posterior (Blei & Jordan, 2004; Doshi-Velez et al., 2009). Similarly, recent work on amortized inference in Bayesian non-parametrics relies on truncation (Miao et al., 2017; Nalisnick & Smyth, 2017; Chatzis, 2014; Singh et al., 2017).

However, the truncated approximation has several drawbacks. If the truncation level is chosen too small, the accuracy of the approximation degrades, whereas if the truncation level is chosen too large, then inference will be slow, removing one of the main advantages of a variational approximation. Perhaps more fundamentally, truncation level can interact poorly with amortized inference, because of the well-known issue called component collapsing (Dinh & Dumoulin, 2016; van den Oord et al., 2017), which is also called over-pruning (Burda et al., 2015; Yeung et al., 2017). This refers to the problem when the inferred latent representation includes components whose conditional distribution given a test data point tends to remains very similar to the prior. Therefore, these are useless components that do not help to explain the data.

In this work, we overcome these limitations using a new dynamic variational approximation, which we call *Roulette-based Amortized Variational Expectations (*RAVE*)*. The goal of RAVE is to allow the approximate variational posterior to adapt its size over the course of the optimization. But

[1]School of Informatics, University of Edinburgh, Edinburgh, United Kingdom [2]MIT-IBM Watson AI Lab, Cambridge, MA, United States [3]Google AI, Mountain View, CA, United States [4]Alan Turing Institute, London, United Kingdom. Correspondence to: Kai Xu <kai.xu@ed.ac.uk>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

this causes the problem that expectations for the evidence lower-bound (ELBO) then require computing an infinite summation which cannot be tackled using the reparameterization trick (Williams, 1992; Kingma & Welling, 2013; Rezende et al., 2014). To surmount this problem, we use a different Monte Carlo approximation, namely, the Russian roulette sampling method from statistical physics (Lux & Koblinger, 1991; Carter & Cashwell, 1975; Lyne et al., 2015), which allows us to approximate this sum by a sample from a Markov chain. This leads to an unbiased estimate of the gradient of the ELBO which can be maximized using stochastic gradient ascent.

We demonstrate RAVE on an *infinite variational autoencoder* (Chatzis, 2014), which assigns each data point to a continuous representation whose size is automatically inferred from the data. The prior on the number of components is given by an Indian buffet process model. We show empirically that previous amortized variational methods suffer from the component collapsing problem and infer useful components, whereas RAVE infers a model with many fewer components, while inferring an overall model of similar explanatory power.

2. Background

We review material from nonparametric Bayesian statistics and variational inference. We also introduce the basic form of the Russian roulette estimate that we will use.

2.1. Indian buffet process

An important problem in representation learning is to learn to represent each data item by a binary vector whose elements indicate latent *features* underlying the data. If the necessary number of features is unknown, we can take a Bayesian approach, and place a prior distribution over all possible latent feature matrices. One such prior distribution is the Indian buffet process (IBP), denoted $\mathbf{Z} \sim \text{IBP}(\alpha)$, which is a probability distribution over sparse binary matrices with a finite number of rows and an unbounded number of columns (Griffiths & Ghahramani, 2011). We define the IBP using the stick-breaking construction (SBC) of Teh et al. (2007), which defines a distribution over $\mathbf{Z}$ as

$$\nu_k \sim \text{Beta}(\alpha, 1),\ \pi_k = \prod_{j=1}^{k} \nu_j,\ z_{nk} \sim \text{Bern}(\pi_k), \quad (1)$$

for $n \in 1 \dots N$ and $k \in 1, 2, \dots, \infty$. Intuitively, we start with a stick of length 1 and break it at random to obtain a new stick of length ν_1. We then break this new stick at another random proportion ν_2 to obtain a new stick of length $\nu_1\nu_2$. This is repeated until we run out of stick to break. We denote it as $\mathbf{Z} \sim \text{SBC}(\alpha, N, K)$ if this process is stopped after K columns. We define IBP(α) to be the distribution $\mathbf{Z} \sim \text{SBC}(\alpha, N, K)$ as $N \to \infty$ and $K \to \infty$.

2.2. Infinite Variational Autoencoders

The IBP can be used as a prior over sparse latent representation $\mathbf{Z} = [\mathbf{z}_1 \dots \mathbf{z}_N]$ of data $\mathbf{X} = [\mathbf{x}_1 \dots \mathbf{x}_N] \in \mathbb{R}^{N \times D}$. Using this prior, we can model the data as

$$\mathbf{Z} \sim \text{IBP}(\alpha),\ \mathbf{A} \sim \mathcal{N}(0, \sigma_A^2 I),\ \mathbf{X} \sim p_\theta(\mathbf{X} \mid \mathbf{Z}, \mathbf{A}), \quad (2)$$

for $n \in 1 \dots N$. A popular model arises when $\mathbf{A}$ is a matrix with D columns and infinitely many rows and when $p_\theta(\mathbf{X} \mid \mathbf{Z}, \mathbf{A}) = \mathcal{N}(\mathbf{X} \mid \mathbf{Z}\mathbf{A}, \sigma_{\mathbf{X}}^2 I)$. This is the well-studied linear Gaussian model. Following Chatzis (2014); Singh et al. (2017), we omit the prior distribution $p(\mathbf{A})$ for this linear model when doing amortized inference, and instead optimize over $\mathbf{A}$.

The *infinite variational autoencoder* (infinite VAE) arises when we choose a deep model to parameterize $p_\theta(\mathbf{X} \mid \mathbf{Z}, \mathbf{A})$. Specifically, choose $p_\theta(\mathbf{X} \mid \mathbf{Z}, \mathbf{A}) = \mathcal{N}(\mathbf{X} \mid \mu_\theta(\mathbf{Z} \odot \mathbf{A}), \sigma_\theta(\mathbf{Z} \odot \mathbf{A}))$ or $p_\theta(\mathbf{X} \mid \mathbf{Z}, \mathbf{A}) = \text{Bern}(\mathbf{X} \mid p_\theta(\mathbf{Z} \odot \mathbf{A}))$ where $\odot$ is the Hadamard product (Chatzis, 2014; Singh et al., 2017), μ_θ, σ_θ, and p_θ are multi-layer neural networks with parameter θ. We refer to these three neural networks as a deep decoder. The infinite VAE arises when we use variational inference with this model, as described next.

2.3. Variational Inference

The posterior distribution over the latent variables $P(\mathbf{Z}, \mathbf{A}, \boldsymbol{\nu} \mid \mathbf{X})$ is intractable. One approach to inference in such models is variational inference. Singh et al. (2017) present a structured variational inference method for this model, based on the method of Hoffman & Blei (2015), which performs better than the more common mean-field approximation, as it introduces dependencies in the approximate posterior distribution, between $\mathbf{Z}$ and $\boldsymbol{\nu}$. The variational posterior from Singh et al. (2017) has the form

$$q(\mathbf{Z}, \mathbf{A}, \boldsymbol{\nu}_{(1:K)}) = q(\mathbf{A})q(\boldsymbol{\nu}_{(1:K)}) \prod_{n=1}^{N} \prod_{k=1}^{K} q(z_{nk} \mid \boldsymbol{\nu}_{(1:K)}),$$

where K is the truncation level. Each component of q has parameters, called *variational parameters*, which are optimized to make $q(\mathbf{Z}, \mathbf{A}, \boldsymbol{\nu}_{(1:K)})$ as close as possible to the true posterior $p(\mathbf{Z}, \mathbf{A}, \boldsymbol{\nu}_{(1:K)} | \mathbf{X})$ as measured by KL-divergence. This is accomplished by optimizing a lower bound called the *evidence lower-bound (ELBO)*. Optimizing the ELBO requires sampling from a Monte Carlo estimate, which is designed to be differentiable with respect to the model parameters and the variational parameters. This can be made possible with the reparameterization trick.[1]

[1] Alternatives include REINFORCE (Williams, 1992), generalized reparameterization gradient (Ruiz et al., 2016), and automatic differentiation variational inference (Kucukelbir et al., 2017).

Singh et al. (2017) employs reparameterizations of the Beta distribution (Nalisnick & Smyth, 2017) and the Bernoulli distribution (Jang et al., 2016; Maddison et al., 2016).

2.4. Russian roulette sampling

Russian roulette sampling (Lux & Koblinger, 1991; Carter & Cashwell, 1975; Lyne et al., 2015; Georgoulas et al., 2017) is a Monte Carlo technique for estimating very large sums. Suppose we want to compute an infinite sum

$$S = \sum_{k=1}^{\infty} T_k, \tag{3}$$

where we assume $S < \infty$. In Russian roulette sampling, we estimate S by truncating the sum after τ terms, where τ is random. As a simple example, let $S = T_1 + T_2$. We can approximate S by the estimate $\hat{S} = T_1$ with probability 0.5, and $\hat{S} = T_1 + 2T_2$ with probability 0.5. It is easy to see that $\mathbb{E}[\hat{S}] = S$. Applying this trick recursively yields the general Russian roulette estimate.

More generally, let $P(\tau) = (1 - \rho_{\tau+1}) \prod_{s=1}^{\tau} \rho_s$, be a distribution over the number of terms in the estimate. Define the truncated sum

$$\hat{S}_K = \sum_{k=1}^{K} \frac{T_k}{p_k}, \tag{4}$$

where $p_k = \prod_{j=1}^{k} \rho_j$ is the probability that $\tau > k$; dividing by this quantity has the effect of correcting for the fact that later terms are less likely to be included in the estimate. Then we define the Russian roulette estimate as $\hat{S} = S_\tau$ with probability $P(\tau)$. It can be shown that this estimate is unbiased, e.g. see the Appendix of Lyne et al. (2015).

3. Method

Now we introduce RAVE, an amortized variational inference method based on dynamic truncation. For concreteness, we describe RAVE in the context of a deep latent factor model with an IBP prior, but the method can be used more generally. First, we introduce a variational family in which the number of latent dimensions is random, governed by its own variational parameters (Section 3.1); this is essentially an infinite mixture of truncated variational distributions. Then, we present the ELBO over all the variational parameters, showing that it can be written as an infinite sum (Section 3.2). Then we show how Russian roulette sampling can be used to obtain an unbiased Monte Carlo estimate the gradient of this sum (Section 3.3). Finally, we put all of these ideas together into a stochastic gradient optimization algorithm that works on a finite representation of the infinite number of parameters (Section 3.4).

3.1. Infinite-sized Variational Family

We start by describing the variational family of approximate posterior distributions that we consider in RAVE. Unlike previous amortized variational methods, the dimensionality is not bounded a priori, but is controlled by continuous variational parameters. We define the variational family using the stick-breaking construction as

$$\begin{aligned} &\nu_k \sim \text{Beta}(\alpha_k, \beta_k), \quad \pi_k = \prod_{j=1}^{k} \nu_j, \\ &K^* = k \quad \text{with probability } m_k = (1 - \rho_{k+1}) \prod_{i=1}^{k} \rho_i, \\ &z_{nk} \sim \text{Bern}(f_\phi(\pi_k, \mathbf{x}_n) \cdot \delta\{k \le K^*\}) \end{aligned} \tag{5}$$

for $n \in 1 \ldots N$ and $k \in 1, 2, \ldots, \infty$. We denote a single variational distribution in this family as $q(\boldsymbol{\nu}, K^*, \mathbf{Z} | \alpha, \beta, \phi, \rho)$. In this equation, α_k, β_k, ρ_k, and ϕ are the variational parameters, and the neural network f_ϕ is an inference network that amortizes the approximation of the posterior distribution.

The only parameters we amortize are those for the variational distribution of $\mathbf{Z}$. Let ϕ be an infinite sequence of vectors $\phi = (\phi_0, \phi_1 \ldots)$ with $\phi_k \in \mathbb{R}^{D+1}$. Then our inference network is

$$f_\phi(\pi_k, \mathbf{x}_n) = \sigma(\text{logit}(\pi_k) + \phi_k^\top [\mathbf{x}_n, 1]), \tag{6}$$

where σ is the sigmoid function.[2]

In order to use the Monte Carlo “reparameterization trick”, we use the Kumaraswamy reparameterzation for the Beta distribution (Nalisnick & Smyth, 2017), the Concrete reparameterization for the Bernoulli distribution (Jang et al., 2016; Maddison et al., 2016).

Note that we have defined this variational family to have an infinite number of parameters: all of the variational parameters α, β, ρ, and ϕ are infinite sequences. In order to optimize these parameters practically, observe that once we fix K^* to some integer k, then the conditional distribution $q(\boldsymbol{\nu}, \mathbf{Z} | K^* = k, \alpha, \beta, \phi, \rho)$ depends only on the first k variational parameters. It is this property that we use to approximate the ELBO in the next section.

This completes the description of the approximate posterior distribution $q(\boldsymbol{\nu}, \mathbf{Z}, K^*)$ for the parameters of the IBP prior. We will also need a variational distribution $q(\mathbf{A})$ over the parameters $\mathbf{A}$ of the observation model also have a Gaussian prior. We omit the details for space because this is standard.

[2]The notation $[\mathbf{x}_n, 1]$ represents vector concatenation, so that the dot product implicitly incorporates the bias term.

3.2. Approximating the ELBO Gradient

To find the best approximate posterior distribution, we maximize the ELBO

$$\mathcal{L} = -\,\mathrm{KL}\left[q(\boldsymbol{\nu}) \,\|\, p(\boldsymbol{\nu})\right] - \mathrm{KL}\left[q(\mathbf{A}) \,\|\, p(\mathbf{A})\right] + \mathbb{E}_{q_{\nu}}\left[\mathbb{E}_{q_{\mathbf{Z}}}\left[\log \frac{p(\mathbf{X} \mid \mathbf{Z}, \mathbf{A}) p(\mathbf{Z} \mid \boldsymbol{\nu})}{q(\mathbf{Z} \mid \boldsymbol{\nu})}\right]\right], \tag{7}$$

which can be derived from the KL-divergence between the marginal distribution $q(\boldsymbol{\nu}, \mathbf{Z}, \mathbf{A})$ and the true posterior $p(\boldsymbol{\nu}, \mathbf{Z}, \mathbf{A}|\mathbf{X})$. (We have omitted the dependence of q on the variational parameters for brevity.) Optimizing this function is challenging for several reasons. First, as mentioned in the previous section, there are an infinite number of variational parameters, so we need to obtain a finite representation. Second, the distribution $q(\mathbf{Z} \mid \boldsymbol{\nu})$ in the third term is not easy to compute, because it is a marginal distribution $q(\mathbf{Z}|\boldsymbol{\nu}) = \sum_{k=0}^{\infty} q(\mathbf{Z}, K^* = k|\boldsymbol{\nu})$. Finally, optimizing with respect to ρ is particularly challenging, intuitively because ρ determines stochastic control flow of q; see Algorithm 1.

Computing the first two terms of Equation 7 is straightforward (see Appendix). For the third term, we re-write this expectation using the tower property

$$\mathbb{E}_{q_{\mathbf{Z}}}\left[\log \frac{p(\mathbf{X} \mid \mathbf{Z}, \mathbf{A}) p(\mathbf{Z} \mid \boldsymbol{\nu})}{q(\mathbf{Z} \mid \boldsymbol{\nu})}\right] = \sum_{k=0}^{\infty} m_k \mathbb{E}_{q_{\mathbf{Z}}}\left[\log \frac{p(\mathbf{X} \mid \mathbf{Z}, \mathbf{A}) p(\mathbf{Z} \mid \boldsymbol{\nu})}{q(\mathbf{Z}|\boldsymbol{\nu})} \mid K^* = k\right]. \tag{8}$$

Because the expection now conditions on $K^* = k$, we know that Z will have at most k nonzero columns, and so the numerator within the expectation is now computable. The denominator $q(\mathbf{Z}|\boldsymbol{\nu})$ is still challenging, because it marginalizes out K^*, and still contains an infinite sum. We can obtain a slightly looser variational lower bound using the inequality

$$q(\mathbf{Z}|\boldsymbol{\nu}) = \sum_{j=1}^{\infty} m_j q(\mathbf{Z}|K^* = j, \boldsymbol{\nu}) \leq q(\mathbf{Z}|K^* = K^\dagger, \boldsymbol{\nu}), \tag{9}$$

where $K^\dagger := \max\{k \mid \exists n, z_{nk} \neq 0\}$, the maximum column index for which that column of $\mathbf{Z}$ is not all 0s. The inequality comes from two facts. First, $q(\mathbf{Z}|K^* = j, \boldsymbol{\nu}) = 0$ for $j < K^\dagger$ because it is impossible to generate more than j features if it is truncated at j. Second, $q(\mathbf{Z}|K^* = j, \boldsymbol{\nu})$ is a monotonically decreasing function for $j \geq K^\dagger$ because observing more columns of zeros will only decrease the probability under the Bernoulli distribution. A more detailed proof is provided in the Appendix.

Combining Equations 7–9, we obtain the training objective

$$\tilde{\mathcal{L}} = \sum_{k=0}^{\infty} m_k \tilde{\mathcal{L}}^k, \tag{10}$$

where

$$\tilde{\mathcal{L}}^k = -\mathrm{KL}\left[q(\boldsymbol{\nu}) \,\|\, p(\boldsymbol{\nu})\right] - \mathrm{KL}\left[q(\mathbf{A}) \,\|\, p(\mathbf{A})\right] + \mathbb{E}_{q_{\nu}}\left[\mathbb{E}_{q_{\mathbf{Z}}}\left[\log \frac{p(\mathbf{X} \mid \mathbf{Z}, \mathbf{A}) p(\mathbf{Z} \mid \boldsymbol{\nu})}{q(\mathbf{Z}|K^* = K^\dagger, \boldsymbol{\nu})} \mid K^* = k\right]\right]. \tag{11}$$

Note that we also move the KL terms for $\boldsymbol{\nu}$ and $\mathbf{A}$ inside the infinite summation. This is valid as the infinite summation is an expectation. This expectation cannot be computed exactly, but we will present a method for approximating it in the next section.

When RAVE is used with the IBP, we will refer to the overall method as RRS-IBP, where RR stands for Russian roulette and S stands for either Structured or Sampling.

Interpretation as random truncation. We give another interpretation of the ELBO in Equation 10. First note that during training, $q(\mathbf{Z}|K^* = k, \boldsymbol{\nu})$ is relaxed to a Concrete distribution, we will have $K^\dagger = k$ because $\mathbf{Z}$ never contains an exact 0 under the continuous relaxation within the truncation level. Thus during training $\tilde{\mathcal{L}}^k$ is exactly the ELBO for the truncated variational distribution that has been used in previous work (Singh et al., 2017), with truncation level k. Therefore, the lower bound $\tilde{\mathcal{L}}$ that we use can be interpreted as the expectation of the truncated ELBO, where the expectation is taken over our variational distribution $q(K^* = k) = m_k$ over the truncation level.

3.3. Russian roulette estimation of the ELBO gradient

To complete our description of the inference algorithm, we describe how we optimize the ELBO $\tilde{\mathcal{L}}$. To simplify the presentation, we introduce the notation $\psi_k = (\alpha_k, \beta_k, \phi_k, \theta_k)$, the vector of all of the variational parameters for component k except for ρ_k as well as the parameter of the generative network for component k, and we define the matrix $\psi_{1:k} = (\psi_1 \ldots \psi_k)$. Then, observe that $\tilde{\mathcal{L}}$ has the form

$$\tilde{\mathcal{L}} = \sum_{k=1}^{\infty} m_k T_k(\psi_{1:k}), \tag{12}$$

where $m_k = (1 - \rho_{k+1}) \prod_{i=1}^{k} \rho_i$ depends only on $\rho_{1:k+1}$, and T_k depends only on the other parameters $\psi_{1:k}$. This can be optimized by stochastic gradient ascent if we can obtain an unbiased estimate of its gradient.

First, the gradient with respect to ψ_k is

$$\partial_{\psi_k} := \frac{\partial \tilde{\mathcal{L}}}{\partial \psi_k} = \sum_{i=k}^{\infty} m_i \frac{\partial T_i}{\partial \psi_k}, \tag{13}$$

where we assume each $\frac{\partial T_i}{\partial \psi_k}$ can be computed by standard automatic differentiation techniques. To estimate this, we use a Russian roulette estimate $\hat{\partial}_{\psi}^{\mathrm{RR}}$ with probabilities m_t.

Algorithm 1 Sampling the truncation level τ during variational optimization, with lazy parameter initialization.

```
$t \leftarrow 0$
loop
  $t \leftarrow t+1$
  if $t > L$ then
    $\rho_{t+1} \leftarrow 0.5$, $\alpha_t \leftarrow \alpha$, $\beta_t \leftarrow 1.0$, $\phi_t \leftarrow \epsilon_0$, $\theta_t \leftarrow \epsilon_0$
    $L \leftarrow L+1$
  return $\tau = t$ with probability $1-\rho_{t+1}$
$\{\alpha$ is the IBP parameter and $\epsilon_0 \sim \mathcal{N}(0, I)\}$
```

More specifically, we sample τ with probability $P(\tau = t) = m_t$, and then return the estimate $\hat{\partial}^{\text{RR}}_{\psi_k} = \hat{\partial}^{\tau}_{\psi_k}$, where

$$\hat{\partial}^{\tau}_{\psi_k} = \sum_{i=k}^{\tau} \frac{m_i}{\left(\prod_{j=1}^{i} \rho_j\right)} \frac{\partial T_i}{\partial \psi_k} = \sum_{i=k}^{\tau} (1-\rho_{i+1}) \frac{\partial T_i}{\partial \psi_k}. \quad (14)$$

To derive the derivatives for ρ_k, first the chain rule yields

$$\partial_{\rho_k} := \frac{\partial \tilde{\mathcal{L}}}{\partial \rho_k} = \sum_{i=1}^{\infty} \frac{\partial \tilde{\mathcal{L}}}{\partial m_i} \frac{\partial m_i}{\partial \rho_k} = \sum_{i=1}^{\infty} T_i \frac{\partial m_i}{\partial \rho_k}, \quad (15)$$

where

$$\frac{\partial m_i}{\partial \rho_k} = \begin{cases} 0 & i < k-1 \\ -\frac{m_i}{1-\rho_k} & i = k-1 \\ \frac{m_i}{\rho_k} & i > k-1 \end{cases}. \quad (16)$$

This yields

$$\partial_{\rho_k} = \sum_{i=k-1}^{\infty} m_i w_i T_i, \quad w_i = \begin{cases} \frac{1}{\rho_k - 1} & i = k-1 \\ \frac{1}{\rho_k} & i > k-1 \end{cases}. \quad (17)$$

Finally, the Russian roulette estimate $\hat{\partial}^{\text{RR}}_{\rho_k}$ of this summation is

$$\hat{\partial}^{\tau}_{\rho_k} = \sum_{i=k-1}^{\tau} \frac{m_i w_i T_i}{\left(\prod_{j=1}^{i} \rho_j\right)} = \sum_{i=k-1}^{\tau} (1-\rho_{i+1}) w_i T_i \quad (18)$$

with probability $P(\tau = t) = m_t$.

Now, each T_i is still difficult to compute, because it contains the expectation

$$\mathbb{E}_{q_\nu}\left[\mathbb{E}_{q_{\mathbf{Z}}}\left[\log \frac{p(\mathbf{X} \mid \mathbf{Z}, \mathbf{A}) p(\mathbf{Z} \mid \boldsymbol{\nu})}{q(\mathbf{Z} | K^* = K^\dagger, \boldsymbol{\nu})} \mid K^* = i\right]\right].$$

We obtain a Monte Carlo approximation of both of these expectations using the standard reparameterization trick.

3.4. Stochastic gradient algorithm

Now that we have Monte Carlo estimates of the necessary derivatives, we can define the stochastic gradient algorithm. The key point is how we manage to operate with only a finite representation of the variational parameters; essentially, we lazily instantiate only the finite subset of variational parameters that receive stochastic gradient updates, More specifically, at every point in the optimization algorithm, we maintain a finite representation of the variational parameters $\psi = (\psi_0 \dots \psi_L)$ and $\rho = (\rho_0 \dots \rho_L)$. These matrices will lazily grow in size as needed, that is, L will grow over the course of the optimization. At the beginning of the algorithm $L = 0$.

Each iteration of stochastic gradient ascent computes new parameters (ψ', ρ') from the current values (ψ, ρ). To do this, first we sample τ from the distribution $P(\tau)$ defined in the previous section. Importantly, it can happen that $\tau > L$. To make this clear, we explicitly give the algorithm for sampling τ in Algorithm 1.

Given a value of τ, we make the gradient updates

$$\psi'_k \leftarrow \psi_k + \epsilon_0 \hat{\partial}^{\tau}_{\psi_k}, \quad k \in 1, 2, \dots \tau$$
$$\rho'_k \leftarrow \rho_k + \epsilon_1 \hat{\partial}^{\tau}_{\rho_k}, \quad k \in 2, 3, \dots \tau + 1$$

where ϵ_0 and ϵ_1 are step sizes. We only need perform the updates for $k \in 1, 2, \dots \tau$ for ψ_k because $\hat{\partial}^{\tau}_{\psi_k} = 0$ if $k > \tau$, and similarly for $\hat{\partial}^{\tau}_{\rho_k}$. Finally, we enforce that $\rho_1 = 1$ to ensure $\tau > 0$.

4. Experiments

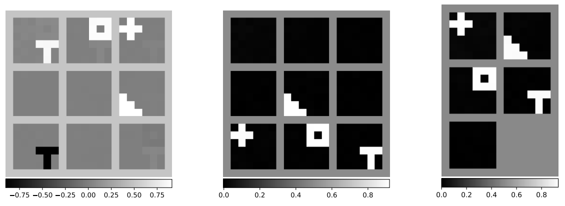
In order to evaluate the efficacy of the structured posterior and variational truncation in inference results of RRS-IBP, we compare it with two previous amortized inference approaches for IBP models MF-IBP from Chatzis (2014) and S-IBP from Singh et al. (2017) respectively.

For optimization, we use Adam (Kingma & Ba, 2014) with a learning rate of 0.001 and momentum parameters set to 0.99 and 0.999, for all parameters except for ρ. For ρ we use a stochastic gradient descent optimizer (Robbins & Monro, 1985) with a learning rate of 0.002. During training, the temperature of Concrete reparameterization is set to 0.1.

For the RRS-IBP sampler, it is possible to vectorize the summations in Section 3.3, implementing them as matrix multiplications. Our Julia implementation has a mean running time of $26.447\mu s$ when $K = 100$. This is very cheap given a moderate K in IBP latent feature models .

4.1. Synthetic data

First, in order to check the correctness of the inference, we compare the different inference methods on synthetic dataset, for which the true data distribution and number of components are known. This data set, proposed by Griffiths & Ghahramani (2011), contains 6×6 grayscale images, each of which are generated as a linear combination of four



(a) MF-IBP (b) S-IBP (c) RRS-IBP

Figure 1. Features learned by VAEs ($\alpha = 4.0$).

black and white images with random weights, plus Gaussian noise $\mathcal{N}(0, 0.1^2)$. We sample 3,000 images for training and 500 held-out images for testing.[3]

All inference methods are applied to the linear-Gaussian IBP model described in Section 2.1. We set $\alpha = 4$ so that it is not exactly same as the true distribution. For MF-IBP and S-IBP, the truncation level is set as 9, so as to be greater than the number of true features in the data.

First, we are interested in whether the inference methods identify the correct number of features for this data. One way to measure this is by the expected number of features per images under the posterior distribution, which we define as $M = N^{-1} \sum_{n=1}^{N} \mathbb{E}_{q\mathbf{z}} \left[\sum_k z_{nk} | \mathbf{x}_n \right]$, where $\mathbf{x}_n$ are the test images. For the true generating process, $M = 2.576$. Because the training data set is large, and the model family contains the generating distribution, the variational approximations should identify a similar number of features. We find that $M = 5.806$ for MF-IBP and $M = 7.612$ for S-IBP, while $M = 2.876$ for RRS-IBP. In other words, both MF-IBP and S-IBP infer many more features than are present in the true distribution, while RRS-IBP infers a value that is closer to the true one.

It may be surprising that the methods are inferring such different values for the number of features, because all three approximate the same posterior distribution. To understand this better, we visualize the learned features in Figure 1, which is simply the matrix $\mathbf{A}$ in the linear decoder. As we can see, S-IBP and RRS-IBP recover four black-and-white image features, which indeed are the images that were used to generated the data. MF-IBP recovers these four features up to a scaling of the image intensities, but also introduces an unnecessary negative feature (lower left in Figure 1(a)). Additionally, both MF-IBP and SS-IBP infer several useless features, which we will call "dummy features".

Such dummy features do no harm if they are never activated, that is, if their corresponding column in $\mathbf{Z}$ is always zero. However, in Figure 2, we plot the activation frequencies

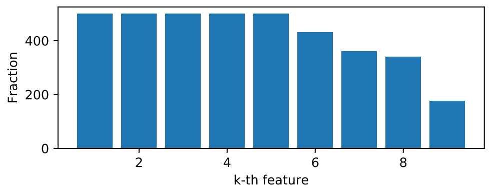
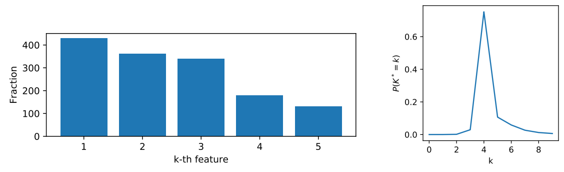


Figure 2. Number of activations per feature for S-IBP



(a) Trace of $\mathbb{E}_m[K^m_{\text{RR}}]$ (b) $P(K^* = k)$

Figure 3. Activation frequencies and m_k for RRS-IBP

i.e. the number of features activated from a single sample of $q(\mathbf{Z}|\boldsymbol{\nu})$, inferred by S-IBP, for the testing set. This figure shows that the top five features from S-IBP are always activated, and in fact these are the 5 black features in Figure 1(b). In other words, *meaningful* features in S-IBP do not necessary come in order, as the method can infer "dummy" features.[4] On the other hand, RRS-IBP mostly avoids this issue (Figure 3(a)), as it learns only one dummy feature and this feature has the lowest probability of those that it infers. As we can see the 4 meaningful features for S-IBP and RRS-IBP actually has similar activation probabilities. We hypothesis that this activation of dummy feature phenomena comes from the fact that when training with a high truncation level, there are many local maximums for the optimization.

The plot of the probability mass function of the stopping level in Figure 3(b) shows that after training, the variational posterior for the truncation level puts most of its mass on the true number of features. This is plot is not available for other methods are the truncation level is fixed.

4.2. Image data

Now we compare the inference methods on benchmark image data sets, namely the MNIST hand-writing digits dataset (LeCun et al., 1998) and Fashion-MNIST, a dataset of images of products (Xiao et al., 2017). We use a deep decoder, where the prior $p(\mathbf{A}) = \mathcal{N}(\mathbf{A}; 0, 1)$, and a Bernoulli observation distribution. Both the encoder and the decoder are two layer neural networks with 500 hidden units and ReLU activation function. We also report the performance of the same model and architecture on the synthetic data from the last section; for that simpler data, we use hidden layer of

[3]Different from Griffiths & Ghahramani (2011) in which each feature is activated with 0.5 probability, we sample an activation matrix $\mathbf{Z} \sim \text{SBC}(4.0, 3500, 4)$ (Equation 1) and generate the dataset as $\mathbf{X} = \mathbf{Z}\mathbf{A}$, where $\mathbf{A}$ is the pre-defined feature matrix.

[4]This was also observed by Singh et al. (2017) on a different synthetic data set (see their appendix).

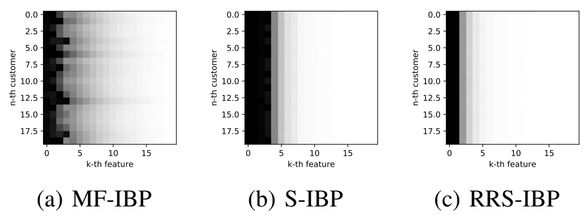


Figure 4. Posterior feature activation probability in grey-scale for VAEs with deep decoders on a subset of SYNTH; darker the higher. The x-axis is the feature indices and the y-axis are data indices. The plot for RRS-IBP is padded to the same number of features for easy comparison.

size 50 and a Gaussian observation distribution.

We measure both the quality of the inferred models and the number of inferred features. To measure model quality, we estimate the marginal probability of the test using the IWAE (Burda et al., 2015), which is a tighter lower bound than the ELBO. For S-IBP, we use the modification of the IWAE from Singh et al. (2017).[5] For RRS-IBP, in order to compare against S-IBP using the same IWAE, we compute the mean of variational distribution of the truncation level, $\bar{m} = \mathbb{E}_{k \sim m_k}[k]$, and use $\lceil \bar{m} \rceil$ as the truncation level when computing the IWAE, where $\lceil \cdot \rceil$ is the ceiling function. To measure the number of inferred features, we report the number of features that have a non-negligible posterior activation probability, defined as $\tilde{K} = \sum_{k=1}^{K_{\max}} \max_{n \in \{1 \ldots N\}} \delta\{q(z_{nk} = 1) > \epsilon\}$ with $\epsilon = 0.01$. We will call $\tilde{K}$ the *number of activated features*.

The results are shown in Table 1. MF-IBP has the worst IWAE over all datasets, consistent with the results of Singh et al. (2017). The model quality of S-IBP and RRS-IBP are mostly similar. However, looking at $\tilde{K}$, we see that RRS-IBP infers many fewer features than the other methods, even though the overall model is of similar or better quality.

Visualizing activation posterior probability We first verify our finding by visualizing the activation posterior probability, $q_{\mathbf{Z}}$, for models on the synthetic dataset. We choose synthetic dataset for visualization because the truncation level for other datasets are too high to populate readable plots. We visualize $q_{\mathbf{Z}}$ on a subset of the synthetic dataset by plotting the probabilities in grey-scale, which is shown in Figure 4. We confirm our previous finding on the visualization: MF-IBP spreads the activation probability after its dark region on the left widely. S-IBP has a more compact dark region and the spread is much less. RRS-IBP has the smallest dark region and spreading effect.

[5] In Singh et al. (2017), a multiplier of 1,000 on the KL term for $\boldsymbol{\nu}$ is used during training, to increase the long tail property of the IBP prior. This increases the number of hidden units used, leading to better IWAE. Because this multiplier is orthogonal to the comparison between models, it is not used in our experiments.

Truncating collapsed components Another way to gain insight into whether the additional features inferred by S-IBP are indeed meaningless dummy features is to visualize the approximate posterior distributions for the S-IBP model. First, to understand $\mathbf{A}$, we plot, on the MNIST dataset[6], the L2 norm of the posterior mean of each feature k, $\|\mathbb{E}_{q_{\mathbf{A}}}[\mathbf{A}_k]\|_2$ ($\mathbf{A}_k$ is the k-th column of $\mathbf{A}$) in the order inferred by the model.[7] This is shown in Figure 5(a). We see that after the first 15 features has a posterior distribution $q_{\mathbf{A}_k}$ with a mean that is not close to 0; for other features, the posterior mean is very close to 0. This is component collapsing (Dinh & Dumoulin, 2016; van den Oord et al., 2017), a known phenomenon in VAEs, which means that some hidden units collapse to the prior and hence convey no information to the decoder. The vertical line in all the plots in Figure 5 indicates the number of features inferred by RRS-IBP. It is striking that RRS-IBP learns to truncate the model precisely at the point where S-IBP stops producing informative features.

Now, to understand $\mathbf{Z}$, we show the activation frequencies, i.e. the number of times each feature is used, for the testing set (Figure 5(b)). This confirms that many of the uninformative features are activated for a substantial fraction of data points. the meaningful dimensions have been activated but also the dimensions in which $q_{\mathbf{A}_k}$ conveys no information. This is similar to what we have visually seen in Section 4.1 for the synthetic data.

As a final test of whether these uninformative features contribute to the quality of the inferred model, we report the test set IWAE of the S-IBP model if after training is complete, the model is truncated after k features (Figure 5(c)). Figure 5(c) confirms our statement that the dummy features conveys no information to the decoder because the IWAE reaches its maximum right after all non-collapsed features are used. It is clear that the additional features do not help improve the inference results, and is a waste of computation, shown in Figure 5(d). However, RRS-IBP avoids this problem due to its nature of automatic truncation and the inferred truncation level (vertical line in red) is just the optimal level of truncation under our post analysis.

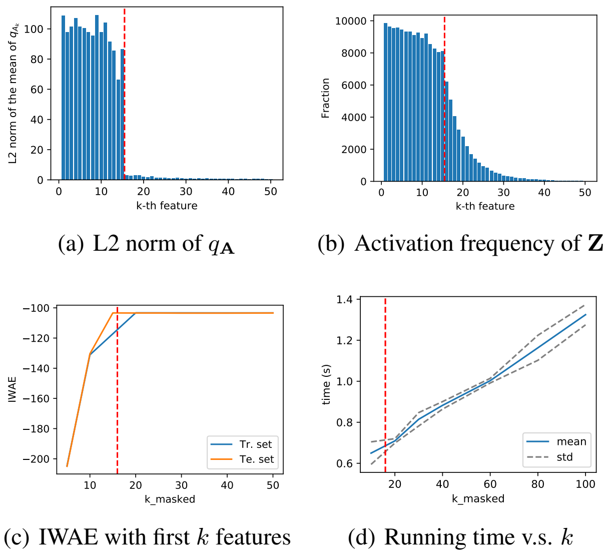
We can understand why S-IBP activates dummy features by considering the effect of dummy features on the ELBO. The additional KL penalty for the dummy features is very small, because $q_{\mathbf{A}_k}$ for the dummy features collapses to the prior. Furthermore, the reconstruction term of the ELBO also does not discourage dummy features too much, because

[6] We conducted the same analysis on Fashion-MNIST and observed qualitatively similar results (see corresponding plots in Appendix).

[7] Observe that because all the inference procedures are based on the stick breaking construction, this order is meaningful; the inferred features are automatically sorted by π_k.

Table 1. IWAE for different VAEs under the IBP prior on various datasets.

METHOD	SYNTH ($\alpha = 4$, $K_{\text{MAX}} = 20$)			MNIST ($\alpha = 10$, $K_{\text{MAX}} = 100$)			FASHION-MNIST ($\alpha = 20$, $K_{\text{MAX}} = 100$)		
	TRAINING	TESTING	$\tilde{K}$	TRAINING	TESTING	$\tilde{K}$	TRAINING	TESTING	$\tilde{K}$
MF-IBP	43.57	43.68	20	-108.74	-110.83	63	-239.7	-242.8	100
S-IBP	44.35	44.38	17	-103.359	-103.448	43	-236.3	-239.8	33
RRS-IBP	45.97	45.94	4	-101.00	-103.29	17	-236.62	-239.7	9



(a) L2 norm of $q_{\mathbf{A}}$ (b) Activation frequency of $\mathbf{Z}$

(c) IWAE with first k features (d) Running time v.s. k

Figure 5. Effects of truncation level for S-IBP with a deep decoder. All plots are computed from the whole testing set of MNIST dataset. The vertical line in red is the corresponding truncation level learned by RRS-IBP. For the first three plots, we omit the last 50 feature level to make the visualization more readable.

the dummy features are just ignored by the decoder. As a result, the ELBO for S-IBP does not provide a large penalty for including these features.

As an aside, we found that optimization becomes difficult MF-IBP and S-IBP with higher truncation levels. For example, on the synthetic data, we were unable to train these methods with $K_{\text{MAX}} = 50$. We hypothesize this is due to an increased number of local maxima introduced by the larger inference and generation networks.

5. Related Work

Variational inference for IBP latent feature models is proposed by Doshi-Velez et al. (2009) and relies on the stick-breaking construction from Teh et al. (2007). We are aware of only a few papers that apply amortized inference to IBP models. Chatzis (2014) uses black box variational inference (BBVI, Ranganath et al. (2014b)) to train a VAE with an IBP prior using the mean-field variational approximation. Later Singh et al. (2017) propose a more accurate approximation by using the Kumaraswamy and Gumbel reparameterizations instead of BBVI, and by introducing a structured variational approximation instead of mean-field. We follow all of those innovations in this work. However, both of these papers rely on finite truncation in the variational distribution, which we avoid in this work. Another type of nonparametric variational autocoder is proposed by Abbasnejad et al. (2017), who introduce an infinite mixture of finite-sized variational autoencoders.

In other Bayesian models, some authors have reduced the impact of truncation by combining truncated variational inference with other optimization methods; however, these methods require running the variational optimisation to convergence separately for different values of K. For example, Hughes et al. (2015) use split and merge steps to change the size of the model but this results in a complex heuristic search algorithm. Similarly, Nalisnick & Smyth (2017) mention some initial experiments on making the level of stick-breaking adaptive by putting a threshold on the percentage of the remaining stick but reports that this approach is slow. Eslami et al. (2016) introduce a generative model on images in which the number of objects in the image is unknown. However, this model still relies on truncation, as indeed the prior distribution over the number of objects is explicit and has finite support. Thus the model lacks some of the appealing theoretical properties of a Bayesian nonparametric prior. Even so, it is likely that the framework of RAVE could be applied for inference in this model; this could be an interesting direction for future work.

Within Bayesian statistics, Russian roulette sampling has been previously used within MCMC algorithms for doubly-intractable distributions (Lyne et al., 2015; Georgoulas et al., 2017). We are unaware of previous work applying Russian roulette sampling within variational methods. Indeed, to our knowledge, ours is the first variational inference method for Bayesian nonparametrics (whether amortized or the more traditional mean-field approach) that avoids truncation.

6. Conclusions and Discussions

In this work we propose RAVE, a novel inference method for BNP models that provides a probabilistic way to perform amortized (neural) variational inference without the need of explicitly truncating the maximum number of inferred posterior features as is common in previous work on variational inference for BNP. We clearly demonstrate that our inference method outperforms both structured and mean-filed amortised variational inference methods that resort to explicit truncation of the posterior features. In future work we aim to extend this method to other non-parametric models.

Acknowledgements

This work was funded by Edinburgh Huawei Research Lab, which is generously funded by Huawei Technologies Co. Ltd.

References

Abbasnejad, M. E., Dick, A., and van den Hengel, A. Infinite variational autoencoder for semi-supervised learning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 781–790. IEEE, 2017.

Adams, R., Wallach, H., and Ghahramani, Z. Learning the structure of deep sparse graphical models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pp. 1–8, 2010.

Antoniak, C. E. Mixtures of dirichlet processes with applications to bayesian nonparametric problems. *The annals of statistics*, pp. 1152–1174, 1974.

Blei, D. M. and Jordan, M. I. Variational methods for the dirichlet process. In *International Conference in Machine Learning (ICML)*, 2004. URL http://doi.acm.org/10.1145/1015330.1015439.

Burda, Y., Grosse, R., and Salakhutdinov, R. Importance weighted autoencoders. *arXiv preprint arXiv:1509.00519*, 2015.

Carter, L. L. and Cashwell, E. D. Particle-transport simulation with the monte carlo method. Technical report, Los Alamos Scientific Lab., 1975.

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

Dinh, L. and Dumoulin, V. Training neural bayesian nets, 2016.

Doshi-Velez, F., Miller, K., Van Gael, J., and Teh, Y. W. Variational inference for the indian buffet process. In *Artificial Intelligence and Statistics*, pp. 137–144, 2009.

Eslami, S. A., Heess, N., Weber, T., Tassa, Y., Szepesvari, D., Hinton, G. E., et al. Attend, infer, repeat: Fast scene understanding with generative models. In *Advances in Neural Information Processing Systems*, pp. 3225–3233, 2016.

Georgoulas, A., Hillston, J., and Sanguinetti, G. Unbiased bayesian inference for population markov jump processes via random truncations. *Statistics and computing*, 27(4): 991–1002, 2017.

Gershman, S. and Blei, D. A tutorial on Bayesian nonparametric models. *Journal of Mathematical Psychology*, 56: 1–12, 2012.

Görür, D. and Rasmussen, C. E. Dirichlet process gaussian mixture models: Choice of the base distribution. *Journal of Computer Science and Technology*, 25(4):653–664, 2010.

Griffiths, T. L. and Ghahramani, Z. The indian buffet process: An introduction and review. *Journal of Machine Learning Research*, 12(Apr):1185–1224, 2011.

Hoffman, M. and Blei, D. Stochastic structured variational inference. In *Artificial Intelligence and Statistics*, pp. 361–369, 2015.

Hughes, M., Kim, D. I., and Sudderth, E. Reliable and Scalable Variational Inference for the Hierarchical Dirichlet Process. In *Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 38, pp. 370–378, 2015.

Jang, E., Gu, S., and Poole, B. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kingma, D. P. and Welling, M. Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Kucukelbir, A., Tran, D., Ranganath, R., Gelman, A., and Blei, D. M. Automatic differentiation variational inference. *The Journal of Machine Learning Research*, 18(1): 430–474, 2017.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Lux, I. and Koblinger, L. *Monte Carlo Particle Transport Methods: Neutron and Photon Calculations*. CRC press, 1991.

Lyne, A.-M., Girolami, M., Atchadé, Y., Strathmann, H., Simpson, D., et al. On Russian roulette estimates for Bayesian inference with doubly-intractable likelihoods. *Statistical science*, 30(4):443–467, 2015.

Maddison, C. J., Mnih, A., and Teh, Y. W. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*, 2016.

Miao, Y., Grefenstette, E., and Blunsom, P. Discovering discrete latent topics with neural variational inference. In *International Conference in Machine Learning (ICML)*, 2017.

Mnih, A. and Gregor, K. Neural variational inference and learning in belief networks. In *International Conference on Machine Learning*, Proceedings of Machine Learning Research, pp. 1791–1799. PMLR, 2014.

Müller, P. and Quintana, F. A. Nonparametric bayesian data analysis. *Statistical science*, pp. 95–110, 2004.

Nalisnick, E. and Smyth, P. Stick-breaking variational autoencoders. In *International Conference on Learning Representations (ICLR)*, 2017.

Orbanz, P. and Teh, Y. W. Bayesian nonparametric models. In *Encyclopedia of Machine Learning*. Springer, 2010.

Ranganath, R., Gerrish, S., and Blei, D. Black box variational inference. In *Artificial Intelligence and Statistics*, 2014a.

Ranganath, R., Gerrish, S., and Blei, D. Black box variational inference. In *Artificial Intelligence and Statistics*, pp. 814–822, 2014b.

Rezende, D. J., Mohamed, S., and Wierstra, D. Stochastic backpropagation and approximate inference in deep generative models. In *International Conference on Machine Learning*, 2014.

Robbins, H. and Monro, S. A stochastic approximation method. In *Herbert Robbins Selected Papers*, pp. 102–109. Springer, 1985.

Ruiz, F. R., AUEB, M. T. R., and Blei, D. The generalized reparameterization gradient. In *Advances in neural information processing systems*, pp. 460–468, 2016.

Singh, R., Ling, J., and Doshi-Velez, F. Structured variational autoencoders for the beta-bernoulli process. 2017.

Teh, Y. W., Jordan, M. I., Beal, M. J., and Blei, D. M. Sharing clusters among related groups: Hierarchical dirichlet processes. In *Advances in neural information processing systems*, pp. 1385–1392, 2005.

Teh, Y. W., Jordan, M. I., Beal, M. J., and Blei, D. M. Hierarchical Dirichlet processes. *Journal of the American Statistical Association*, 101(476):1566–1581, 2006.

Teh, Y. W., Grür, D., and Ghahramani, Z. Stick-breaking construction for the indian buffet process. In *Artificial Intelligence and Statistics*, pp. 556–563, 2007.

van den Oord, A., Vinyals, O., et al. Neural discrete representation learning. In *Advances in Neural Information Processing Systems*, pp. 6297–6306, 2017.

Walker, S. G., Damien, P., Laud, P. W., and Smith, A. F. Bayesian nonparametric inference for random distributions and related functions. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 61(3): 485–527, 1999.

Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. In *Reinforcement Learning*, pp. 5–32. Springer, 1992.

Xiao, H., Rasul, K., and Vollgraf, R. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

Yeung, S., Kannan, A., Dauphin, Y., and Fei-Fei, L. Tackling over-pruning in variational autoencoders. *arXiv preprint arXiv:1706.03643*, 2017.

A. KL terms for $\mathbf{A}$ and $\boldsymbol{\nu}$

The computation for the KL terms for $\mathbf{A}$ and $\boldsymbol{\nu}$ in Equation 7 is omitted in the main paper and we present here to show how to compute them.

The KL term for $\mathbf{A}$ is the KL between two Normal distributions $q_{\mathbf{A}} = \mathcal{N}(\mu_\phi(\mathbf{x}), \sigma_\phi(\mathbf{x}))$ and $p_{\mathbf{A}} = \mathcal{N}(0, I)$. We use its closed-form expression adapted from the Appendix of Kingma & Welling (2013)

$$\mathrm{KL}\left[q(\mathbf{A}) \,\|\, p(\mathbf{A})\right] = -\frac{1}{2}\sum_{k=1}^{\infty}\left(1 + \log((\sigma_k)^2) - (\mu_k)^2 - \sigma_k)^2\right), \tag{19}$$

where $\mu_k := \mu_{\phi_k}(\mathbf{x})$ and $\sigma_k := \sigma_{\phi_k}(\mathbf{x})$.

The KL term for $\boldsymbol{\nu}$ is the KL between the Kumaraswamy and Beta distributions $q(\nu_k) = \mathrm{Kumaraswamy}(a_k, b_k)$ and $p(\nu_k) = \mathrm{Beta}(\alpha, 1)$. We use its closed-form expression adapted from Appendix of Nalisnick & Smyth (2017)

$$\mathrm{KL}\left[q(\boldsymbol{\nu}) \,\|\, p(\boldsymbol{\nu})\right] = \sum_{k=1}^{\infty} \mathrm{KL}\left[\mathrm{Kumaraswamy}(a_k, b_k) \,\|\, \mathrm{Beta}(\alpha, 1)\right], \tag{20}$$

where

$$\begin{aligned}&\mathrm{KL}\left[\mathrm{Kumaraswamy}(a, b) \,\|\, \mathrm{Beta}(\alpha, \beta)\right]\\ =&\frac{a-\alpha}{a}\left(-\gamma - \Psi(b) - \frac{1}{b}\right) + \log ab + \log B(\alpha, \beta) - \frac{b-1}{b} + (\beta - 1)b\sum_{m=1}^{\infty}\frac{1}{m+ab}B\left(\frac{m}{a}, b\right).\end{aligned} \tag{21}$$

We approximate the infinite sum in Equation 21 using its first 11 terms as it is suggested by Nalisnick & Smyth (2017).

B. Proof of the Inequality in Equation 9

Equation 9 states the inequality that

$$q(\mathbf{Z}|\boldsymbol{\nu}) = \sum_{j=1}^{\infty} m_j q(\mathbf{Z}|K^* = j, \boldsymbol{\nu}) \leq q(\mathbf{Z}|K^* = K^\dagger, \boldsymbol{\nu}), \tag{9}$$

where $K^\dagger := \max_k\{\exists n, z_{nk} \neq 0\}$, the maximum column index for which that column of $\mathbf{Z}$ is not all 0s.

To prove this, we begin with the definition of $q_{\mathbf{Z}}$ given a truncation level j

$$q(\mathbf{Z}|K^* = j, \boldsymbol{\nu}) := \delta\{j \geq K^\dagger\}\prod_{n=1}^{N}\prod_{k=1}^{j}\pi_k^{z_{nk}}(1-\pi_k)^{(1-z_{nk})}. \tag{22}$$

First, $q(\mathbf{Z}|K^* = j, \boldsymbol{\nu}) = 0$ for $j < K^\dagger$ because of the delta function that comes from the truncation. Second, $q(\mathbf{Z}^k|K^* = j, \boldsymbol{\nu})$ is a monotonically decreasing function for $j \geq K^\dagger$. To see this, consider $q(\mathbf{Z}|K^* = l, \boldsymbol{\nu})$ for which $K^\dagger \leq j < l$

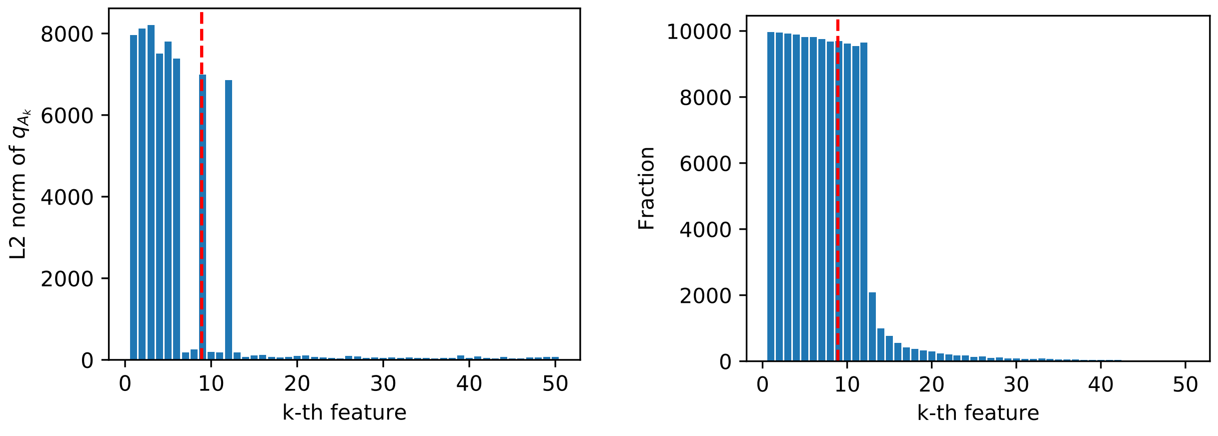
$$\begin{aligned}q(\mathbf{Z}|K^* = l, \boldsymbol{\nu}) &= \delta\{l \geq K^\dagger\}\prod_{n=1}^{N}\prod_{k=1}^{l}\pi_k^{z_{nk}}(1-\pi_k)^{(1-z_{nk})}\\ &= \prod_{n=1}^{N}\left(\prod_{k=1}^{j}\pi_k^{z_{nk}}(1-\pi_k)^{(1-z_{nk})}\prod_{k=j+1}^{l}\pi_k^{z_{nk}}(1-\pi_k)^{(1-z_{nk})}\right)\\ &= \left(\prod_{n=1}^{N}\prod_{k=1}^{j}\pi_k^{z_{nk}}(1-\pi_k)^{(1-z_{nk})}\right)\left(\prod_{n=1}^{N}\prod_{k=j+1}^{l}\pi_k^{z_{nk}}(1-\pi_k)^{(1-z_{nk})}\right)\\ &= q(\mathbf{Z}|K^* = j, \boldsymbol{\nu})\left(\prod_{n=1}^{N}\prod_{k=j+1}^{l}\pi_k^{z_{nk}}(1-\pi_k)^{(1-z_{nk})}\right)\\ &=: q(\mathbf{Z}|K^* = j, \boldsymbol{\nu})Q\\ &\leq q(\mathbf{Z}|K^* = j, \boldsymbol{\nu})\end{aligned} \tag{23}$$

The last step comes from the fact that $Q <= 1$ since $\pi_k \in [0, 1], \forall k = 1, \ldots, \infty$. Now we can show

$$\begin{aligned} q(\mathbf{Z}|\boldsymbol{\nu}) &= \sum_{j=1}^{\infty} m_j q(\mathbf{Z}|K^* = j, \boldsymbol{\nu}) \\ &= \sum_{j=1}^{K^\dagger - 1} m_j q(\mathbf{Z}|K^* = j, \boldsymbol{\nu}) + \sum_{j=K^\dagger}^{\infty} m_j q(\mathbf{Z}|K^* = j, \boldsymbol{\nu}) \\ &\leq \sum_{j=K^\dagger}^{\infty} m_j q(\mathbf{Z}|K^* = K^\dagger, \boldsymbol{\nu}) \\ &\leq \sum_{j=1}^{\infty} m_j q(\mathbf{Z}|K^* = K^\dagger, \boldsymbol{\nu}) = q(\mathbf{Z}|K^* = K^\dagger, \boldsymbol{\nu}) \end{aligned}$$

C. Visualization of component collapsing of S-IBP on Fashion-MNIST

For Fashion-MNIST, the corresponding component collapsing visualization of Figure 5 is given in Figure 6. Note that worse



(a) L2 norm of $q_{\mathbf{A}}$

(b) Activation frequency of $\mathbf{Z}$

Figure 6. Effects of truncation level for S-IBP with a deep decoder. All plots are computed from the whole testing set of Fashion-MNIST dataset. The vertical line in red is the corresponding truncation level learned by RRS-IBP. For the first three plots, we omit the last 50 feature level to make the visualization more readable.

than MNIST, there are even dummy features that are activated as frequently as active features, e.g. the 7-th and 8-th features in Figure 6(a) are shown to be frequently activated in Figure 6. The number of active features shown in Figure 6(a) is 8 and the mean of the truncation distribution inferred by RRS-IBP is 8.9, indicated in vertical red lines in both plots, which means RRS-IBP successfully inferred the same number of active features.