Discrete Neural Processes

Ari Pakman [1] **Liam Paninski** [1]

Abstract

Many data generating processes involve latent random variables over discrete combinatorial spaces whose size grows factorially with the dataset. In these settings, existing posterior inference methods can be inaccurate and/or very slow. In this work we develop methods for efficient amortized approximate Bayesian inference over discrete combinatorial spaces, with applications to probabilistic clustering (such as Dirichlet process mixture models), random communities (such as stochastic block models) and random permutations. The approach exploits the exchangeability of the generative models and is based on mapping distributed, permutation-invariant representations of discrete arrangements into conditional probabilities. The resulting algorithms parallelize easily, yield iid samples from the approximate posteriors along with a probability estimate of each sample (a quantity generally unavailable using Markov Chain Monte Carlo) and can easily be applied to both conjugate and non-conjugate models, as training only requires samples from the generative model.

1. Introduction

Discrete latent random variables appear in a wide variety of statistical models. When these variables have a combinatorial nature (e.g. permutations, graphs, partitions) the state space grows factorially with the data size, making inference challenging.

Popular inference methods in these models fall into a few broad classes. First, we can attempt to compute a maximum a posteriori (MAP) point estimate. However, exploring the full posterior is crucial whenever there is irreducible uncertainty about the latent structure (or when many separate local optima exist), as is often the case in these models. Second, Markov Chain Monte Carlo (MCMC) methods for exploring the posterior (Neal, 2000; Jain & Neal, 2004; Diaconis, 2009; McDaid et al., 2013) are asymptotically accurate but time-consuming, with convergence that is difficult to assess. Models whose likelihood and prior are non-conjugate are particularly challenging, since in general in these cases the model parameters cannot be marginalized and must be kept as part of the state of the Markov chain. Finally, variational methods (Blei & Jordan, 2004; Kurihara et al., 2007; Airoldi et al., 2008; Hughes et al., 2015; Linderman et al., 2018) are typically much faster but do not come with accuracy guarantees.

In this work we propose a novel technique to perform approximate posterior inference in combinatorial spaces. While the details differ in each generative model, the common motif is to use neural networks to express posterior distributions of latent discrete variables in terms of fixed-dimensional, distributed data representations that respect the symmetries imposed by the discrete variables.

The method can be applied to both conjugate and non-conjugate models, and is amortized in the sense that, after investing computational time in training a neural network with samples from a particular generative model, we can obtain independent, parallelizable, approximate posterior samples of the discrete variables for any new set of observations of arbitrary size, with no need for expensive MCMC steps.

We present our approach in three different settings: in Section 2 we study random clustering; in Section 3 random community graph models; and in Section 4 random permutations. In each case, we present experimental results to illustrate the method. Section 5 discusses related works, and we close in Section 6 by discussing potential directions for future work.

2. Clusters

Probabilistic models for clustering (McLachlan & Basford, 1988) introduce random variables c_i denoting the cluster number to which the data point x_i is assigned, and assume

[1]Department of Statistics, Center for Theoretical Neuroscience, and Grossman Center for the Statistics of Mind, Columbia University. Correspondence to: Ari Pakman <aripakman@gmail.com>.

a generating process of the form

$$\begin{aligned} \alpha_1, \alpha_2 &\sim p(\alpha) \\ N &\sim p(N) \\ c_1 \dots c_N &\sim p(c_1, \dots, c_N | \alpha_1) \\ \mu_1 \dots \mu_K | c_{1:N} &\sim p(\mu_1, \dots \mu_K | \alpha_2) \\ x_i &\sim p(x_i | \mu_{c_i}) \quad i = 1 \dots N \end{aligned}$$

Here α_1, α_2 are hyperameters. The number of clusters K is a random variable, indicating the number of distinct values among the sampled c_i's, and μ_k denotes a parameter vector controlling the distribution of the k-th cluster (e.g., μ_k could include both the mean and covariance of a Gaussian mixture component). We assume that the priors $p(c_{1:N}|\alpha_1)$ and $p(\mu_{1:K}|\alpha_2)$ are exchangeable, namely,

$$p(c_1, \dots, c_N | \alpha_1) = p(c_{\sigma_1}, \dots, c_{\sigma_N} | \alpha_1)\,,$$

where $\{\sigma_i\}$ is an arbitrary permutation of the indices, and similarly for $p(\mu_{1:K}|\alpha_2)$. Examples of this setting include Mixtures of Finite Mixtures (Miller & Harrison, 2018) and many Bayesian nonparametric models, such as Dirichlet process mixture models (DPMM); see (Rodriguez & Mueller, 2013) for a recent overview.

Given N data points $\mathbf{x} = \{x_i\}$, we are interested in sampling the c_i's, using a decomposition

$$p(c_{1:N}|\mathbf{x}) = p(c_1|\mathbf{x}) p(c_2|c_1, \mathbf{x}) \dots p(c_N | c_{1:N-1}, \mathbf{x}). \quad (1)$$

Note that $p(c_1 = 1|\mathbf{x}) = 1$, since the first data point is always assigned to the first cluster. While we might be interested in all the hidden variables, the reason to focus on the discrete variables c_i's is that given samples from them, it is generally easy to obtain posterior samples from $p(\alpha_1|c_{1:N})$ and $p(\mu_i, \alpha_2|\mathbf{x}, c_{1:N})$.

We would like to model all the factors in (1) in a unified way, with a generic factor given by

$$p(c_n | c_{1:n-1}, \mathbf{x}) = \frac{p(c_1 \dots c_n | \mathbf{x})}{\sum_{c'_n=1}^{K+1} p(c_1 \dots c'_n | \mathbf{x})}. \quad (2)$$

Here we assumed that there are K unique values in $c_{1:n-1}$, and therefore c_n can take $K+1$ values, corresponding to x_n joining any of the K existing clusters, or forming its own new cluster.[1]

We are interested in approximating (2) with a neural network as

$$p(c_n | c_{1:n-1}, \mathbf{x}) \simeq p_\theta(c_n | c_{1:n-1}, \mathbf{x})\,. \quad (3)$$

[1] If the prior $p(c_{1:N}|\alpha_1)$ only allows up to K_m clusters, and $K = K_m$, then the sum in the denominator of (2) should be up to K only.

Algorithm 1 $O(NK)$ Neural Clustering Process Sampling

$h_i \leftarrow h(x_i) \quad i = 1 \dots N$ {Notation}
$Q \leftarrow \sum_{i=2}^{N} h_i$ {Initialize unassigned set}
$H_1 \leftarrow h_1$ {Create first cluster with x_1}
$G \leftarrow g(H_1)$
$K \leftarrow 1, c_1 \leftarrow 1$
for $n \leftarrow 2 \dots N$ **do**
  $Q \leftarrow Q - h_n$ {Remove x_n from unassigned set}
  $H_{K+1} \leftarrow 0$ {We define $g(0) = 0$}
  for $k \leftarrow 1 \dots K+1$ **do**
    $G \leftarrow G + g(H_k + h_n) - g(H_k)$ {Add x_n}
    $p_k \leftarrow e^{f(G, Q, h_n)}$
    $G \leftarrow G - g(H_k + h_n) + g(H_k)$ {Remove x_n}
  end for
  $p_k \leftarrow p_k / \sum_{k'=1}^{K+1} p_{k'}$ {Normalize probabilities}
  $c_n \sim p_k$ {Sample assignment for x_n}
  if $c_n = K+1$ **then**
    $K \leftarrow K + 1$
  end if
  $G \leftarrow G - g(H_{c_n}) + g(H_{c_n} + h_n)$ {Add point x_n}
  $H_{c_n} \leftarrow H_{c_n} + h_n$
end for
Return $c_1 \dots c_N$

Such a network should extract features from the data and combine them nonlinearly. A major contribution of this work is to let the design of such a network be guided by the highly symmetric structure of the lhs of (3).

To make this symmetric structure more transparent, and in light of the expression (2), let us consider the joint distribution of the assignments of the first n data points,

$$p(c_1, \dots, c_n | \mathbf{x})\,. \quad (4)$$

A neural representation of this quantity should respect the permutation symmetries imposed on the x_i's by the values of $c_{1:n}$. Therefore, our first task is to build symmetry invariant representations of the observations $\mathbf{x}$.

We proceed by considering three distinct symmetries:

- **Permutations within a cluster:** (4) is invariant under permutations of x_i's belonging to the same cluster. For each of the K clusters that have been sampled so far, we define the encoding

$$H_k = \sum_{i: c_i = k} h(x_i) \qquad k = 1 \dots K\,, \quad (5)$$

  where $h : \mathbb{R}^{d_x} \to \mathbb{R}^{d_h}$ is a function we will learn from data. This encoding H_k is clearly invariant under permutations of x_i's belonging to the same cluster.

- **Permutations between clusters:** (4) is invariant under permutations of the cluster labels. In terms of

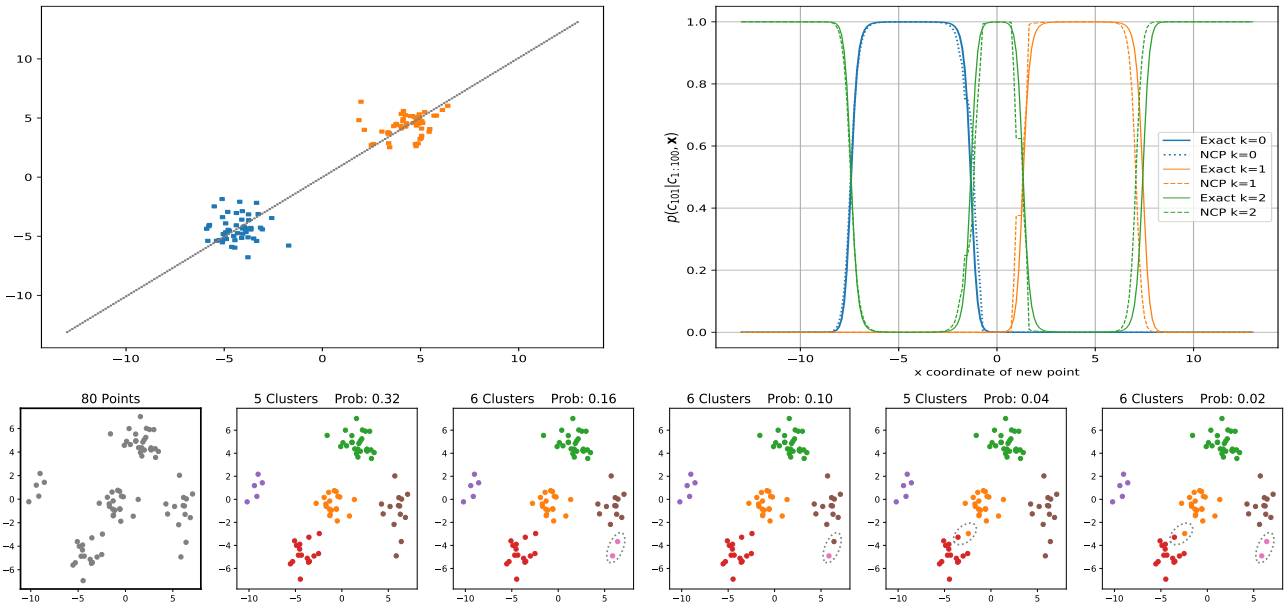

Figure 1. **Neural Clustering Process vs. Exact Posteriors.** *Upper left:* Two 2D clusters of 50 points each ($k = 0, 1$) and a line over possible locations of a 101st last point. *Upper right:* Assuming a DPMM (here with $\alpha = 0.7$, and 2D Gaussian observations with unit variance and mean with a prior $N(0, \sigma_\mu = 10 \times \mathbf{1}_2)$), the posterior $p(c_{101}|c_{1:100}, \mathbf{x})$ can be computed exactly, and we compare it to the NCP estimate as a function of the horizontal coordinate of x_{101}, as this point moves over the gray line on the upper left panel. *Lower:* Five samples from the posterior of the same 2D model, given the observations in the leftmost panel. For each sample we indicate its number of clusters and its probability under the posterior. Note that the posterior samples are reasonable, and less-reasonable samples are assigned lower probability by the NCP. The dotted ellipses indicate areas where some cluster assignments differ from the first, highest-probability sample. In our GPU-parallelized implementation, we obtain thousands of such NCP samples in a fraction of a second. (Best seen in color.)

the within-cluster invariants H_k, this symmetry can be captured by

$$G = \sum_{k=1}^{K} g(H_k), \tag{6}$$

where $g : \mathbb{R}^{d_h} \to \mathbb{R}^{d_g}$.

- **Permutations of the unassigned data points:** (4) is also invariant under permutations of the $N - n$ unassigned data points. This can be captured by

$$Q = \sum_{i=n+1}^{N} h(x_i)\,, \tag{7}$$

where we used the same encoding function h as in (5). Note that G and Q provide fixed-dimensional, symmetry-invariant representations of all the assigned and non-assigned data points, respectively, for any number of N data points and K clusters.

Now, each of the $K + 1$ possible values for c_n yields a configuration with its associated vector G_k. In terms of the G_k's and Q, we propose to model (2) as

$$p_\theta(c_n = k|c_{1:n-1}, \mathbf{x}) = \frac{e^{f(G_k, Q, h_n)}}{\sum_{k'=1}^{K+1} e^{f(G_{k'}, Q, h_n)}} \tag{8}$$

for $k = 1 \ldots K + 1$, where $h_n = h(x_n)$ and we defined a new function f that takes real values.

In eq. (8), θ denotes the parameters in the functions h, g and f; we represent these functions with neural networks. By storing and updating G and Q for successive values of n, the computational cost of a full i.i.d. sample of $c_{1:N}$ is $O(NK)$, the same as a single Gibbs sweep. See Algorithm 1 for details; we term this approach the Neural Clustering Process (NCP).

In order to learn the parameters θ of the neural networks, we use stochastic gradient descent to minimize the expected negative log-likelihood,

$$-\mathbb{E}_{p(N)}\mathbb{E}_{p(c_{1:N}, \mathbf{x})}\left[\sum_{n=2}^{N} \log p_\theta(c_n|c_{1:n-1}, \mathbf{x})\right].$$

Samples from $p(c_{1:N}, \mathbf{x})$ are obtained from the generative model, irrespective of the model being conjugate.

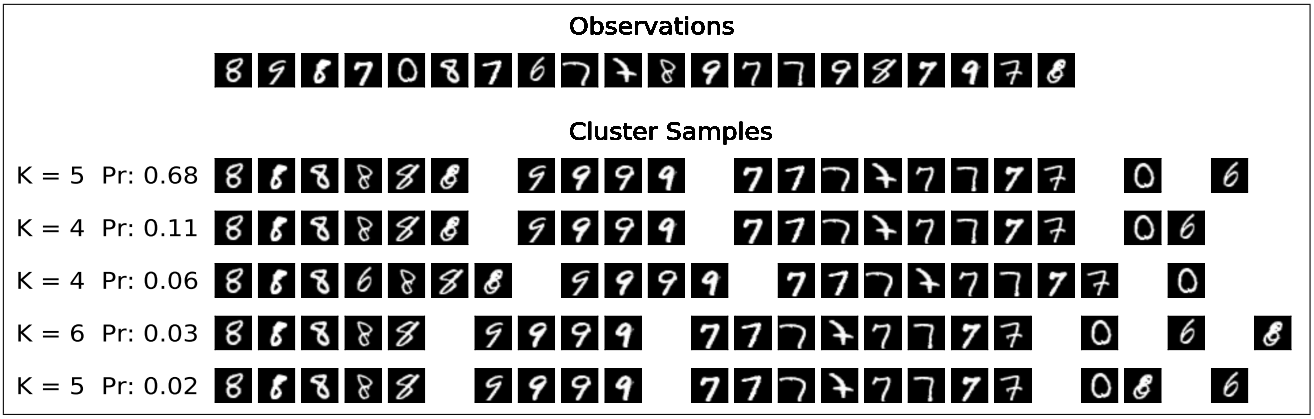

Figure 2. **Clustering of MNIST data.** The generative model is a DPMM with concentration parameter $\alpha = 0.7$ and a uniform discrete base measure over the 10 labels. Conditioned on a label, observations are sampled uniformly from the MNIST training set. The figure shows above $N = 20$ observations, generated similarly from the MNIST test set. The five rows below the observations show five samples of $c_{1:20}$ from the NCP posterior of these 20 images, with their corresponding probabilities, each capturing some ambiguity suggested by the form of particular digits.

Note that since we can take an unlimited number of samples from the generative model, we can potentially train a neural network to approximate $p(c_n|c_{1:n-1}; \mathbf{x})$ arbitrarily accurately - modulo the usual concerns about local optima and training computation time.

2.1. Global symmetry from exchangeability

The symmetries we have exploited rely on $p(c_{1:N}|\alpha_1)$ being exchangeable, which in turn implies an additional global symmetry. This can be seen in the lhs of the autoregressive factorization (1). The joint posterior of the c_i's inherits the exchangeability of the prior, but this symmetry is not explicit in the rhs of (1), where a particular order is chosen for the expansion.

If our model learns the correct form for the conditional probabilities, this symmetry should be (approximately) satisfied, and this should be monitored during training, as we show in the Appendix.

2.2. Examples

Here we illustrate the method by presenting two examples.[2] In the Appendix we provide details of the network architectures used in all the examples of the paper. Note that each sample comes with a probability estimate, generally unavailable using Markov Chain Monte Carlo.

Clustering in 2D Gaussian models

We consider a DPMM clustering model for 2D points. The generative model is

$$\begin{aligned} N &\sim \text{Uniform}[5, 100] \\ c_{1:N} &\sim \text{CRP}(\alpha) \\ \mu_k &\sim N(0, \sigma_\mu^2 \mathbf{1}_2) \quad k = 1 \dots K \\ x_i &\sim N(\mu_{c_i}, \sigma^2 \mathbf{1}_2) \quad i = 1 \dots N \end{aligned}$$

where CRP stands for the Chinese Restaurant Process, with concentration parameter $\alpha = 0.7$, $\sigma_\mu = 10$, $\sigma = 1$, and $d_x = 2$. Figure 1 shows some results. In particular, we compare the estimated assignment probabilities for the final observation of a set, c_N, against their exact values, which are computable for conjugate models, showing excellent agreement.

Clustering of MNIST digits

We consider next a DPMM over the empirical distribution of digits from the MNIST dataset. The generative model is

$$\begin{aligned} N &\sim \text{Uniform}[5, 100] \\ c_{1:N} &\sim \text{CRP}_{10}(\alpha) \\ l_k &\sim \text{Unif}[0, 9] - \text{without replacement.} \quad k = 1 \dots K \\ x_i &\sim \text{Unif}[\text{MNIST digits with label } l_{c_i}] \quad i = 1 \dots N \end{aligned}$$

where CRP_{10} is a Chinese Restaurant Process truncated to up to 10 clusters, with $\alpha = 0.7$, $d_x = 28 \times 28$. Figure 2 shows results for this model. In particular, note how the posterior samples correctly capture the shape ambiguity of some of the digits. Note that in this case the generative model has no analytical expression (and therefore is non-conjugate), but this presents no problem; a generative model that we can sample from is all we need for training.

[2] Code available at http://www.github.com/aripakman/neural_clustering_process

3. Communities

Our next setting has a similar prior as above over cluster labels, but the observation model is more challenging:

$$\begin{aligned} \alpha, N &\sim p(\alpha), p(N) \\ c_1 \dots c_N &\sim p(c_1, \dots, c_N | \alpha) \\ \phi_{k_1,k_2} &\sim \text{Beta}(\alpha, \beta) \quad k_1 \le k_2 \\ x_{i,j} &\sim \text{Bernoulli}(\phi_{c_i,c_j}) \,, \quad i \le j \,, \quad i,j = 1 \dots N \end{aligned} \tag{9}$$

where $k_1, k_2 = 1 \dots K$. The prior $p(c_{1:n}|\alpha)$ can be any of the priors used for clustering above, and the observations $x_{i,j}$ represent now the presence or absence of an edge in a graph of N vertices. We set $\phi_{k_1,k_2} = \phi_{k_2,k_1}$ and $x_{i,j} \equiv x_{j,i}$, and assume $x_{ij} \in \{+1, -1\}$.

Examples of this family of models include stochastic block models (Holland et al., 1983; Nowicki & Snijders, 2001) and the single-type Infinite Relational Model (Kemp et al., 2006; Xu et al., 2006). One can also consider different Beta priors in (9) for $k_1 = k_2$ and $k_1 \neq k_2$.

3.1. Encoding each row of the adjacency matrix

In principle posterior inference in this case can proceed similarly to the clustering case, by considering N particles, each given by a row of the adjacency matrix

$$\mathbf{x}_i = (x_{i,1}, \dots, x_{i,N}) \quad i = 1 \dots N \,. \tag{10}$$

But we should be careful when encoding these particles. Consider the situation when the values of $c_{1:n}$ have been assigned. Encoding with a generic function $h(\mathbf{x}_i)$ would ignore the permutation symmetries present among the components of $\mathbf{x}_i$, i.e., the columns of the matrix $x_{i,j}$, as a result of the $c_{1:n}$ assignments. These symmetries are the same three symmetries discussed above for clustering models. Moreover, a fixed function $h(\mathbf{x}_i)$ would not be able to accommodate the fact that the length of $\mathbf{x}_i$ changes with the size N of the dataset.

Suppose that there are K clusters among the $c_{1:n}$, each with s_k elements. In order to simplify the notation, let us assume that the $N - n$ unassigned points all belong to an additional $(K+1)$-th cluster with $s_{K+1} = N - n$, so we assume $c_{n+1:N} = K + 1$, and we have $\sum_{k=1}^{K+1} s_k = N$ and $s_k = \sum_{j=1}^{N} \delta(c_j = k)$.

Now, in each row $\mathbf{x}_i$, the number s_k of elements in the k-th cluster can be split as

$$\begin{aligned} s_k &= s^-_{i,k} + s^+_{i,k} \\ s^+_{i,k} &= \sum_{j=1}^{N} \delta(c_j = k)\delta(x_{i,j} = +1) \\ s^-_{i,k} &= \sum_{j=1}^{N} \delta(c_j = k)\delta(x_{i,j} = -1) \end{aligned}$$

and note that both $s^-_{i,k}$ and $s^+_{i,k}$ are invariant under the symmetry of permuting the indices within cluster k.

Example: $N = 5$ and $\mathbf{x}_1 = (+1, +1, -1, +1, +1)$. If four assignments were made $c_1 = c_2 = 1$, $c_3 = c_4 = 2$, then $K = 2$ and $c_5 = 3$, and from $\mathbf{x}_1$ we get $s^+_{1,1} = 2, s^-_{1,1} = 0$, $s^+_{1,2} = 1, s^-_{1,2} = 1, s^+_{1,3} = 1, s^-_{1,3} = 0$. If we permute the columns 3 and 4, both from cluster $k = 2$, we get $\mathbf{x}_1 = (+1, +1, +1, -1, +1)$, but all the $s^\pm_{1,j}$'s stay invariant.

Additional invariants can be obtained combining $s^+_{j,k}$ and $s^-_{j,k}$ across all rows $\mathbf{x}_j$'s with $c_j = c_i$, such as

$$m^+_{c_i,k} = \frac{1}{s_{c_i}} \sum_{j:c_j=c_i} s^+_{j,k} \tag{11}$$

$$v^+_{c_i,k} = \frac{1}{s_{c_i}} \sum_{j:c_j=c_i} (s^+_{j,k} - m^+_{c_i,k})^2 \tag{12}$$

and similarly $m^-_{c_i,k}$ and $v^-_{c_i,k}$. Note that these invariants are the same for all rows $\mathbf{x}_j$ with $c_j = k$. The motivation to consider them is that, if the partition corresponding to $c_{1:n}$ is correct, then for $i \le n$ and $k \le K$ we have $n^+_{i,k} \simeq m^+_{c_i,k}$ since they are both estimators of the latent Bernoulli parameter $\phi_{c_i,k}$. For the same reason, if the partition is correct and those two estimators of $\phi_{c_i,k}$ are exact, then $v^+_{c_i,k} \simeq 0$. Similarly for $m^-_{c_i,k}$ and $v^-_{c_i,k}$. Then these values provide learning signals to the network that estimates the probability of the assignments $c_{1:n}$ being correct.

Therefore we propose to encode the components of $\mathbf{x}_i$ belonging to cluster k as

$$r_{i,k} = (s^+_{i,k}, m^+_{c_i,k}, v^+_{c_i,k}, s^-_{i,k}, m^-_{c_i,k}, v^-_{c_i,k}) \quad \in \mathbb{R}^6. \tag{13}$$

In order to preserve the symmetry of the first K labels under permutations, we combine them as

$$t_i \equiv \sum_{k=1}^{K} t(r_{i,k}) \quad \in \mathbb{R}^{d_t} \tag{14}$$

where the encoding function is $t : \mathbb{R}^6 \to \mathbb{R}^{d_t}$. The encoding (13) of the unassigned components $x_{i,n+1:N}$ is kept separate and denoted as $q_i = r_{i,K+1}$.

In summary, each row $\mathbf{x}_i$ of the adjacency matrix is represented by the fixed-dimensional pair $(t_i, q_i) \in \mathbb{R}^{d_t+6}$ in a way that respects the symmetries of the assignments $c_{1:n}$: permutations between members of a cluster, permutations of cluster labels and permutations among unassigned columns.

3.2. Clustering the rows of the adjacency matrix

We can proceed now as in regular clustering, encoding each cluster of $\mathbf{x}_i$'s within $c_{1:n}$ as

$$H_k = \sum_{i:c_i=k} h(t_i, q_i) \quad \in \mathbb{R}^{d_h} \qquad k = 1 \dots K, \tag{15}$$

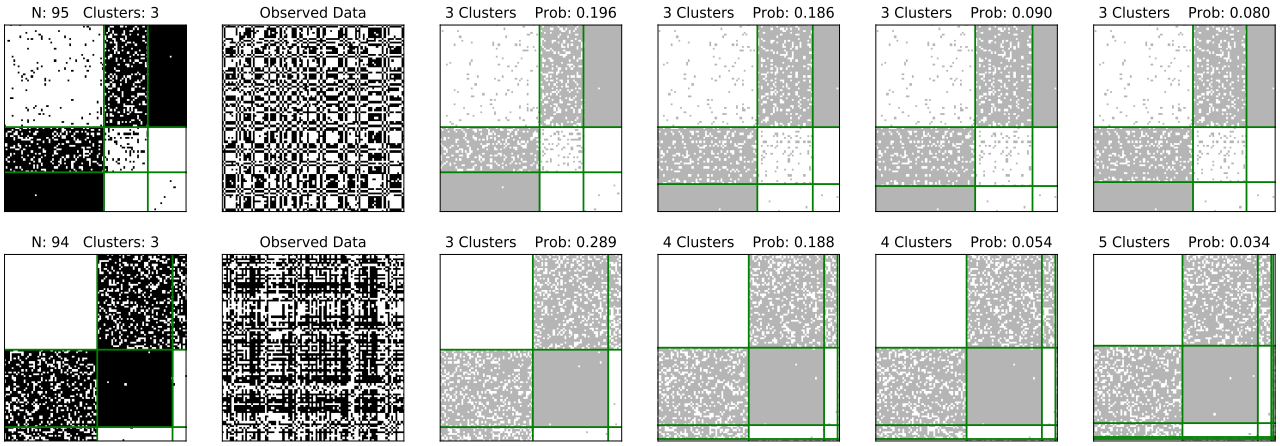

Figure 3. **Community Detection with Neural Block Processes.** The model is a single-type Infinite Relational Model (Kemp et al., 2006; Xu et al., 2006), with a Chinese Restaurant Process prior with $\alpha = 0.7$. The entries in each block are Bernoulli samples, with a block parameter sampled from a Beta$(0.2, 0.2)$ prior. Each row of panels shows, for two different cases, from left to right: (i) the original block structure, sampled from the generative model, (ii) the observed network, obtained by randomly permuting rows and columns, (iii) four different samples from the NBP posterior, along with their estimated probabilities. The blocks in the original data and in the samples appear in decreasing cluster size in order to facilitate the visual comparison.

and defining the permutation invariant, fixed-dimensional vectors

$$G = \sum_{k=1}^{K} g(H_k), \tag{16}$$

$$Q = \sum_{i=n+1}^{N} h(t_i, q_i). \tag{17}$$

In terms of these quantities, the conditional probabilities are defined as usual as

$$p_\theta(c_n = k | c_{1:n-1}, \mathbf{x}) = \frac{e^{f(G_k, Q, h_n)}}{\sum_{k'=1}^{K+1} e^{f(G_{k'}, Q, h_n)}} \tag{18}$$

for $k = 1 \dots K+1$, with $h_n = h(t_n, q_n)$ and with G_k being the value of G for the different configurations. Compared to the regular clustering case, here we need to learn four functions, t, h, g and f, instead of three. We call our approach Neural Block Process (NBP).

3.3. Example: Infinite Relational Model

We consider a single-type Infinite Relational Model, with generating process

$$\begin{aligned} N &\sim \text{Uniform}[5, 100] \\ c_{1:N} &\sim \text{CRP}(\alpha) \\ \phi_{k_1,k_2} &\sim \text{Beta}(0.2, 0.2) \quad k_1 \le k_2 \\ x_{i,j} &\sim \text{Bernoulli}(\phi_{c_i,c_j}), \quad i \le j, \quad i,j = 1 \dots N \end{aligned}$$

with $\alpha = 0.7$. Figure 3 shows results for this model, exhibiting an excellent capacity of the NBP to cluster the network into distinct blocks.

4. Permutations

Our last set of models have a generating process of the form

$$\begin{aligned} \alpha_1, \alpha_2, \alpha_3 &\sim p(\alpha) \\ N &\sim p(N) \\ c_1 \dots c_N &\sim p(c_1, \dots, c_N | \alpha_1) \\ x_1 \dots x_N &\sim p(x_1, \dots, x_N | \alpha_2) \\ \mu_1 \dots \mu_N &\sim p(\mu_1, \dots, \mu_N | \alpha_3) \\ y_i &\sim p(y_i | x_{c_i}, \mu_i) \quad i = 1 \dots N. \end{aligned}$$

Here $p(c_{1:N}|\alpha_1)$ is a distribution (usually uniform) over permutations, with the random variable

$$c_i \in \{1, \dots, N\}, \quad c_i \neq c_j \text{ for } i \neq j$$

denoting that x_{c_i} is paired with y_i. As a concrete example, think of y_i as a noise-corrupted version of a permuted sample x_{c_i}; here μ_i is a parameter that controls the conditional distribution of y_i given x_{c_i}. The distributions $p(c_{1:N}|\alpha_1), p(x_{1:N}|\alpha_2)$ and $p(\mu_{1:N}|\alpha_3)$ are assumed to be exchangeable.

Given two sets of N data points $\mathbf{x} = \{x_i\}$, $\mathbf{y} = \{y_i\}$, we are interested in sampling the posterior of the c_i's, using a decomposition

$$\begin{aligned} p(c_{1:N}|\mathbf{x}, \mathbf{y}) = \; & p(c_1|\mathbf{x}, \mathbf{y}) p(c_2|c_1, \mathbf{x}, \mathbf{y}) \\ & \dots p(c_N|c_{1:N-1}, \mathbf{x}, \mathbf{y}); \end{aligned} \tag{19}$$

note now that $p(c_N|c_{1:N-1}, \mathbf{x}, \mathbf{y}) = 1$, since the last point y_N is always matched with the last unmatched point among the x_i's.

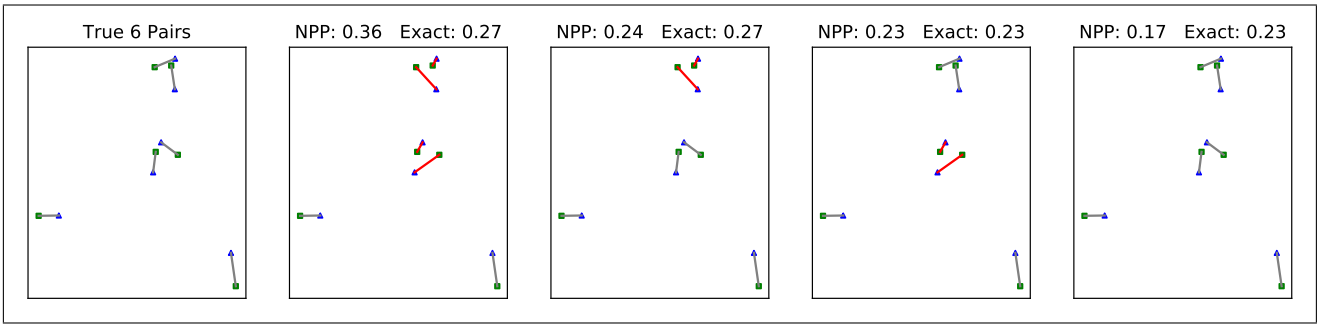

Figure 4. **Neural Permutation Process for 2D Points with Gaussian Noise.** The data are generated by matching each point x_i (green square) with a point y_i (blue triangle), obtained by sampling from $N(x_i, 0.6\mathbf{1}_2)$. The labels of x_i are then scrambled and the two sets $\{y_i\}$ and $\{x_i\}$ are observed. *First panel:* Observed y_i's and x_i's for $N = 6$. The matching links indicated by gray lines are not observed, and are shown here only for reference. *Second to fifth panel:* Four samples of the discrete variables c_i from the NPP posterior. These four values of c_i (corresponding to switches of the nearby pairs of y_i) dominate the posterior probabilities. In each case, we indicate the probability of the sample estimated by the NPP conditional, and its corresponding exact value, which can be computed in this model. The red matching links indicate cases in which the sampled pairing differs from the reference matching in the first panel.

Our interest is again a generic factor in (19)

$$p(c_n|c_{1:n-1}, \mathbf{x}, \mathbf{y}) = \frac{p(c_1 \dots c_n|\mathbf{x}, \mathbf{y})}{\sum_{i=1}^{N-n} p(c_1, \dots, c_n = k_i|\mathbf{x}, \mathbf{y})} \quad (20)$$

where c_n takes values in $\{k_i\}_{i=1}^{N-n+1}$. We proceed again by building representations of the data that respect the symmetries in the joint distribution of the permutation of the first n data points $y_{1:n}$,

$$p(c_1, \dots, c_n|\mathbf{x}, \mathbf{y})\,. \quad (21)$$

The symmetries of this expression are:

- **Permutations among the paired variables:** (21) is invariant under permutations of the assigned pairs $(y_i, x_{c_i})_{i=1}^n$. This invariance can be encoded as

$$H = \sum_{i=1}^{n} h(y_i, x_{c_i}) \quad (22)$$

  where $h : \mathbb{R}^{d_x+d_y} \to \mathbb{R}^{d_h}$.

- **Separate permutations among unpaired variables:** (21) is also invariant under separate permutations of the unpaired data points. Denoting by $\{k_i\}_{i=1}^{N-n}$ the set of $N - n$ indices not taken by the $c_{1:n}$, these two symmetries can be captured by

$$G_y = \sum_{i=n+1}^{N} g_y(y_i), \quad (23)$$

$$G_x = \sum_{i=1}^{N-n} g_x(x_{k_i}), \quad (24)$$

  where $g_x : \mathbb{R}^{d_x} \to \mathbb{R}^{d_{g_x}}$ and $g_y : \mathbb{R}^{d_y} \to \mathbb{R}^{d_{g_y}}$.

Note that H, G_x and G_y provide fixed-dimensional, symmetry-invariant representations of all the paired and unpaired data points for any n and N.

For each value of c_n, say $c_n = k_i$, there is a different pair of vectors H in (22) and G_x in (24), which we denote by H_i and $G_{x,i}$ ($i = 1, \dots, N - n + 1$). In terms of these vectors, we propose to model (20) as

$$p_\theta(c_n = k_i|c_{1:n-1}, \mathbf{x}, \mathbf{y}) = \frac{e^{f(H_i, G_{x,i}, G_y, h_{n,i})}}{\sum_{j=1}^{N-n+1} e^{f(H_j, G_{x,j}, G_y, h_{n,j})}}$$

where $h_{n,i} \equiv h(y_n, x_{k_i})$, for $i = 1 \dots N - n + 1$. We defined the function f that takes real values, and θ denotes all the parameters in the functions f, h, g_x and g_y. By storing and updating H, G_x and G_y for successive values of n, the computational cost of a full sample of $c_{1:N}$ is $O(N^2)$. See Algorithm 2 in the Appendix for details; we call this approach the Neural Permutation Process (NPP).

4.1. Examples

Permutation of Noisy Pairs of Points in 2D

We consider N pairs of points in 2D obtained by adding Gaussian noise to each member of a group of N points:

$$\begin{aligned} c_{0:N} &\sim \text{Unif [permutations]} \\ x_i &\sim N(0, 3) \quad i = 1 \dots N \\ y_i &\sim N(x_{c_i}, 0.6\mathbf{1}_2) \quad i = 1 \dots N \end{aligned}$$

Figure 4 shows results for this model.

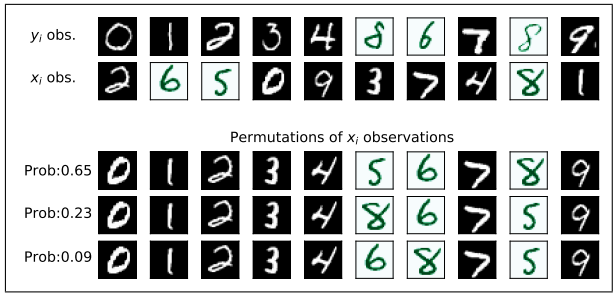

Figure 5. **Neural Permutation Process for MNIST digits.** The data are generated by sampling two sets of 10 different MNIST digits and scrambling their order, as shown in the first two rows. Each sample from the NPP posterior yields a permutation of the x_i's. Three such samples are shown in the lower three rows, indicating their probabilities. Note that the digit '5' in the y_i's resembles '8' and '6', and the digit '8' in the y_i's resembles '5'. These ambiguities, highlighted in the figure, are captured by the permutation samples.

Permutation of MNIST digits

We consider permutations among sets of MNIST digits, with the following generative model with $N = 10$,

$$\begin{aligned} c_{0:9} &\sim \text{Unif [permutations]} \\ y_i &\sim \text{Unif [MNIST digits with label } i\text{]} \quad i = 0 \dots 9 \\ x_i &\sim \text{Unif [MNIST digits with label } c_i\text{]} \quad i = 0 \dots 9 \end{aligned}$$

In words, the "noise model" for y_i in this example is simply to replace one sample of the digit in x_{c_i} with another sampled digit with the corresponding permuted label. Figure 5 shows an example from the posterior in this model, illustrating the importance of a probabilistic approach to capture the ambiguities in the observations.

5. Related Works

Many works have studied posterior inference in the models discussed. Our work differs from previous approaches in its use of neural networks to explicitly approximate the posterior over combinatorial discrete spaces in order to obtain iid approximate samples.

The relation between permutation symmetries and neural architectures has been explored recently in (Ravanbakhsh et al., 2017; Korshunova et al., 2018; Bloem-Reddy & Teh, 2019). The representation of a set via a sum (or mean) of encoding vectors was also used in (Zaheer et al., 2017; Guttenberg et al., 2016; Ravanbakhsh et al., 2016; Edwards & Storkey, 2017; Garnelo et al., 2018a;b).

For MAP estimates, the recent work (Bengio et al., 2018) surveys deep learning techniques for combinatorial optimization.

Posteriors over random permutations have been studied using MCMC techniques in (Diaconis, 2009), and variational methods in (Linderman et al., 2018).

The works (Du, 2010; Aljalbout et al., 2018; Min et al., 2018) review clustering methods based on neural networks, and (Pehlevan et al., 2018) proposes a biologically inspired network for online clustering. An alternative approach to clustering based on probabilistic models for the data acts by maximizing information-theoretic criteria over cluster configurations (Slonim & Tishby, 2000; Slonim et al., 2005; Faivishevsky & Goldberger, 2010).

The work (Schmidt & Morup, 2013) gives an overview of relational models with Bayesian non-parametric priors. For stochastic block models, posterior inference is studied in (Decelle et al., 2011) using the cavity method and belief propagation, the work (Abbe, 2018) surveys recent MAP techniques, and (van der Pas & van der Vaart, 2018) studies the consistency of MAP estimates. Neural architectures for community detection on graphs have been studied in (Chen et al., 2019) as a classification problem for every node.

Similar amortized approaches to Bayesian inference have been explored in Bayesian networks (Stuhlmüller et al., 2013), sequential Monte Carlo (Paige & Wood, 2016), probabilistic programming (Ritchie et al., 2016; Le et al., 2016) and particle tracking (Sun & Paninski, 2018).

6. Outlook

The method we introduced can potentially be applied to many other discrete model settings (Orbanz & Roy, 2015). In each case, the symmetries should be explored in order to define the appropriate encodings for the observations.

Also, many non-exchangeable priors have been studied for both permutations (Critchlow et al., 1991; Lebanon & Lafferty, 2002) and clustering (MacEachern, 2000; Wallach et al., 2010; Blei & Frazier, 2011; Foti & Williamson, 2015; Di Benedetto et al., 2017; Lu et al., 2018). In these cases, the global symmetry discussed in Section 2.1 is not present, and some of the partial permutation symmetries we exploited in our construction are not present either, requiring different encoding schemes. Recurrent neural networks may provide useful more general encodings in these cases; we plan to explore these directions in future work.

Acknowledgements

We thank Scott Linderman and Ruoxi Sun for helpful conversations and Aaron Schein for comments on the draft. This work was supported by the Simons Foundation, the DARPA NESD program, ONR N00014-17-1-2843, NIH/NIBIB R01 EB22913, NSF NeuroNex Award DBI-1707398 and The Gatsby Charitable Foundation.

References

Abbe, E. Community Detection and Stochastic Block Models. *Foundations and Trends in Communications and Information Theory*, 14(1-2):1–162, 2018.

Airoldi, E. M., Blei, D. M., Fienberg, S. E., and Xing, E. P. Mixed membership stochastic blockmodels. *Journal of Machine Learning Research*, 9(Sep):1981–2014, 2008.

Aljalbout, E., Golkov, V., Siddiqui, Y., and Cremers, D. Clustering with Deep Learning: Taxonomy and New Methods. *arXiv preprint arXiv:1801.07648*, 2018.

Bengio, Y., Lodi, A., and Prouvost, A. Machine learning for combinatorial optimization: a methodological tour d'horizon. *arXiv preprint arXiv:1811.06128*, 2018.

Blei, D. M. and Frazier, P. I. Distance dependent Chinese restaurant processes. *Journal of Machine Learning Research*, 12(Aug):2461–2488, 2011.

Blei, D. M. and Jordan, M. I. Variational Methods for the Dirichlet Process. In *Proceedings of the Twenty-first International Conference on Machine Learning*, ICML '04, 2004.

Bloem-Reddy, B. and Teh, Y. W. Probabilistic symmetry and invariant neural networks. *arXiv preprint arXiv:1901.06082*, 2019.

Chen, Z., Li, L., and Bruna, J. Supervised Community Detection with Line Graph Neural Networks. *ICLR*, 2019.

Critchlow, D. E., Fligner, M. A., and Verducci, J. S. Probability models on rankings. *Journal of mathematical psychology*, 35(3):294–318, 1991.

Decelle, A., Krzakala, F., Moore, C., and Zdeborová, L. Asymptotic analysis of the stochastic block model for modular networks and its algorithmic applications. *Physical Review E*, 84(6):066106, 2011.

Di Benedetto, G., Caron, F., and Teh, Y. W. Nonexchangeable random partition models for microclustering. *arXiv preprint arXiv:1711.07287*, 2017.

Diaconis, P. The Markov chain Monte Carlo revolution. *Bulletin of the American Mathematical Society*, 46(2): 179–205, 2009.

Du, K.-L. Clustering: A neural network approach. *Neural networks*, 23(1):89–107, 2010.

Edwards, H. and Storkey, A. Towards a neural statistician. *ICLR*, 2017.

Faivishevsky, L. and Goldberger, J. Nonparametric information theoretic clustering algorithm. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pp. 351–358, 2010.

Foti, N. J. and Williamson, S. A. A survey of non-exchangeable priors for Bayesian nonparametric models. *IEEE transactions on pattern analysis and machine intelligence*, 37(2):359–371, 2015.

Garnelo, M., Rosenbaum, D., Maddison, C. J., Ramalho, T., Saxton, D., Shanahan, M., Teh, Y. W., Rezende, D. J., and Eslami, S. Conditional neural processes. In *International Conference on Machine Learning*, 2018a.

Garnelo, M., Schwarz, J., Rosenbaum, D., Viola, F., Rezende, D. J., Eslami, S., and Teh, Y. W. Neural processes. In *ICML 2018 workshop on Theoretical Foundations and Applications of Deep Generative Models*, 2018b.

Geweke, J. Getting it right: Joint distribution tests of posterior simulators. *Journal of the American Statistical Association*, 99(467):799–804, 2004.

Guttenberg, N., Virgo, N., Witkowski, O., Aoki, H., and Kanai, R. Permutation-equivariant neural networks applied to dynamics prediction. *arXiv preprint arXiv:1612.04530*, 2016.

Holland, P. W., Laskey, K. B., and Leinhardt, S. Stochastic blockmodels: First steps. *Social networks*, 5(2):109–137, 1983.

Hughes, M., Kim, D. I., and Sudderth, E. Reliable and scalable variational inference for the hierarchical Dirichlet process. In *Artificial Intelligence and Statistics*, pp. 370–378, 2015.

Jain, S. and Neal, R. M. A split-merge Markov chain Monte Carlo procedure for the Dirichlet process mixture model. *Journal of computational and Graphical Statistics*, 13(1): 158–182, 2004.

Kemp, C., Tenenbaum, J. B., Griffiths, T. L., Yamada, T., and Ueda, N. Learning systems of concepts with an infinite relational model. In *AAAI*, volume 3, pp. 5, 2006.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *ICLR*, 2015.

Korshunova, I., Degrave, J., Huszar, F., Gal, Y., Gretton, A., and Dambre, J. Bruno: A deep recurrent model for exchangeable data. In *Advances in Neural Information Processing Systems 31*. 2018.

Kurihara, K., Welling, M., and Teh, Y. W. Collapsed Variational Dirichlet Process Mixture Models. In *IJCAI*, volume 7, pp. 2796–2801, 2007.

Le, T. A., Baydin, A. G., and Wood, F. Inference compilation and universal probabilistic programming. *arXiv preprint arXiv:1610.09900*, 2016.

Lebanon, G. and Lafferty, J. Cranking: Combining rankings using conditional probability models on permutations. In *ICML*, volume 2, pp. 363–370. Citeseer, 2002.

Linderman, S. W., Mena, G. E., Cooper, H., Paninski, L., and Cunningham, J. P. Reparameterizing the Birkhoff polytope for variational permutation inference. In *AISTATS*, 2018.

Lu, J., Li, M., and Dunson, D. Reducing over-clustering via the powered Chinese restaurant process. *arXiv preprint arXiv:1802.05392*, 2018.

MacEachern, S. N. Dependent Dirichlet processes. *Unpublished manuscript, Department of Statistics, The Ohio State University*, pp. 1–40, 2000.

McDaid, A. F., Murphy, T. B., Friel, N., and Hurley, N. J. Improved bayesian inference for the stochastic block model with application to large networks. *Computational Statistics & Data Analysis*, 60:12–31, 2013.

McLachlan, G. J. and Basford, K. E. *Mixture models: Inference and applications to clustering*, volume 84. Marcel Dekker, 1988.

Miller, J. W. and Harrison, M. T. Mixture models with a prior on the number of components. *Journal of the American Statistical Association*, 113(521):340–356, 2018.

Min, E., Guo, X., Liu, Q., Zhang, G., Cui, J., and Long, J. A survey of clustering with deep learning: From the perspective of network architecture. *IEEE Access*, 6: 39501–39514, 2018.

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

Nowicki, K. and Snijders, T. A. B. Estimation and prediction for stochastic blockstructures. *Journal of the American statistical association*, 96(455):1077–1087, 2001.

Orbanz, P. and Roy, D. M. Bayesian models of graphs, arrays and other exchangeable random structures. *IEEE transactions on pattern analysis and machine intelligence*, 37(2):437–461, 2015.

Paige, B. and Wood, F. Inference networks for sequential Monte Carlo in graphical models. In *International Conference on Machine Learning*, pp. 3040–3049, 2016.

Pehlevan, C., Genkin, A., and Chklovskii, D. B. A clustering neural network model of insect olfaction. *bioRxiv*, 2018.

Ravanbakhsh, S., Schneider, J., and Poczos, B. Deep learning with sets and point clouds. *arXiv preprint arXiv:1611.04500*, 2016.

Ravanbakhsh, S., Schneider, J., and Póczos, B. Equivariance through parameter-sharing. In *Proceedings of the 34th International Conference on Machine Learning*, 2017.

Ritchie, D., Horsfall, P., and Goodman, N. D. Deep amortized inference for probabilistic programs. *arXiv preprint arXiv:1610.05735*, 2016.

Rodriguez, A. and Mueller, P. NONPARAMETRIC BAYESIAN INFERENCE. *NSF-CBMS Regional Conference Series in Probability and Statistics*, 9:i–110, 2013.

Schmidt, M. N. and Morup, M. Nonparametric bayesian modeling of complex networks: An introduction. *IEEE Signal Processing Magazine*, 30(3):110–128, 2013.

Slonim, N. and Tishby, N. Document clustering using word clusters via the information bottleneck method. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 208–215. ACM, 2000.

Slonim, N., Atwal, G. S., Tkačik, G., and Bialek, W. Information-based clustering. *Proceedings of the National Academy of Sciences*, 102(51):18297–18302, 2005.

Stuhlmüller, A., Taylor, J., and Goodman, N. Learning stochastic inverses. In *Advances in neural information processing systems*, pp. 3048–3056, 2013.

Sun, R. and Paninski, L. Scalable approximate Bayesian inference for particle tracking data. In *Proceedings of the 35th International Conference on Machine Learning*, 2018.

van der Pas, S. L. and van der Vaart, A. W. Bayesian community detection. *Bayesian Analysis*, 13(3):767–796, 2018.

Wallach, H., Jensen, S., Dicker, L., and Heller, K. An alternative prior process for nonparametric Bayesian clustering. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pp. 892–899, 2010.

Xu, Z., Tresp, V., Yu, K., and Kriegel, H.-P. Learning infinite hidden relational models. *Uncertainity in Artificial Intelligence (UAI2006)*, 2006.

Zaheer, M., Kottur, S., Ravanbakhsh, S., Póczos, B., Salakhutdinov, R., and Smola, A. J. Deep sets. In *Advances in neural information processing systems*, 2017.

A. The NPP Algorithm

Algorithm 2 $O(N^2)$ Neural Permutation Process Sampling

$r_i, q_i \leftarrow g_y(y_i), g_x(x_i) \quad i = 1 \dots N$ {Notation}
$H \leftarrow 0$
$G_y \leftarrow \sum_{i=1}^{N} r_i$
$G_x \leftarrow \sum_{i=1}^{N} q_i$
for $n \leftarrow 1 \dots N-1$ **do**
  $G_y \leftarrow G_y - r_i$ {Remove from unpaired set}
  if $n = 1$ **then**
    $\{k_i\}_{i=1}^{N} \leftarrow \{1, \dots, N\}$ {Available indices}
  else
    $\{k_i\}_{i=1}^{N-n+1} \leftarrow \{1, \dots, N\} / \{c_{1:n-1}\}$
  end if
  for $i \leftarrow 1 \dots N-n+1$ **do**
    $G_x \leftarrow G_x - q_{k_i}$ {Remove from unpaired set}
    $H \leftarrow H + h(y_n, x_{k_i})$ {Add to paired set}
    $p_i \leftarrow e^{f(H, G_x, G_y, h_n)}$
    $G_x \leftarrow G_x + q_{k_i}$ {Undo}
    $H \leftarrow H - h(y_n, x_{k_i})$ {Undo}
  end for
  $p_i \leftarrow p_i / \sum_{i'=1}^{N-n+1} p_{i'}$ {Normalize probabilities}
  $i \sim p_i$ {Sample assignment for y_n}
  $c_n \leftarrow k_i$
  $G_x \leftarrow G_x - q_{c_n}$ {Remove from unpaired set}
  $H \leftarrow H + h(y_n, x_{c_n})$ {Add pair to paired set}
end for
Return $c_1 \dots c_{N-1}$

B. Diagnostics

B.1. Geweke's test

A popular test to verify the correctness of MCMC implementations, proposed by Geweke (Geweke, 2004), can also be performed in our case. In our setting, the test amounts to comparing samples from

$$p_\theta(c_{1:N}) \equiv \int d\mathbf{x}\, p_\theta(c_{1:N}|\mathbf{x})\, p(\mathbf{x})\,, \tag{25}$$

where $p(\mathbf{x})$ is the marginal from the generative model, with samples from the prior $p(c_{1:N})$. Figure 6 shows such a comparison for the cases of 2D clustering and the relational model. In the former case, the agreement is excellent. In the latter, the result suggests some modest bias, which might be corrected using bigger neural architectures.

B.2. Monitoring global permutation invariance

As mentioned in Section 2.1, we must verify the symmetry of the posterior likelihood under global permutations of all the data points. We show such a check in Figure 7.

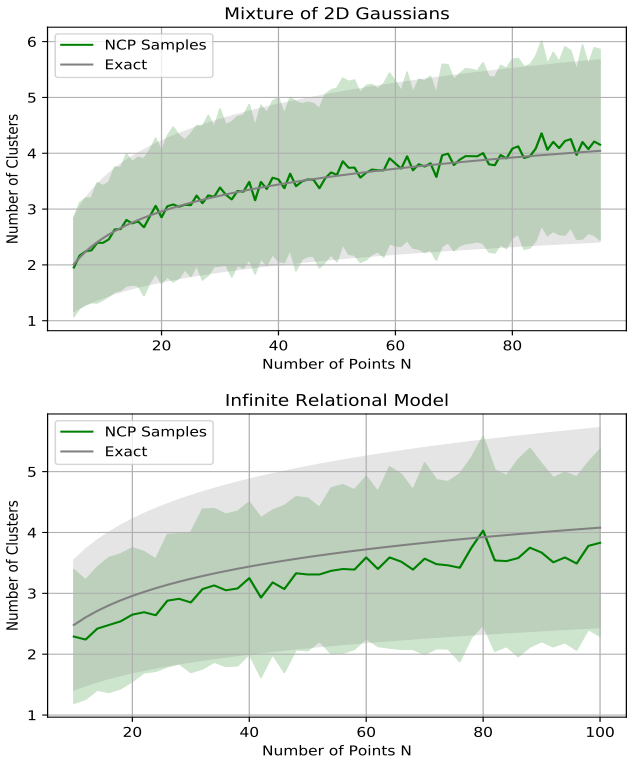

Figure 6. **Geweke's Test.** The curves compare the exact mean ($\pm$ one standard deviation) of the number of clusters for different N's from the Chinese Restaurant Process prior (with $\alpha = 0.7$), with sampled estimates using equation (25). *Above:* the 2D Gaussian mixture of Section 2.2, showing good agreement. *Below:* the IRM model of Section 3.3, exhibiting a small bias.

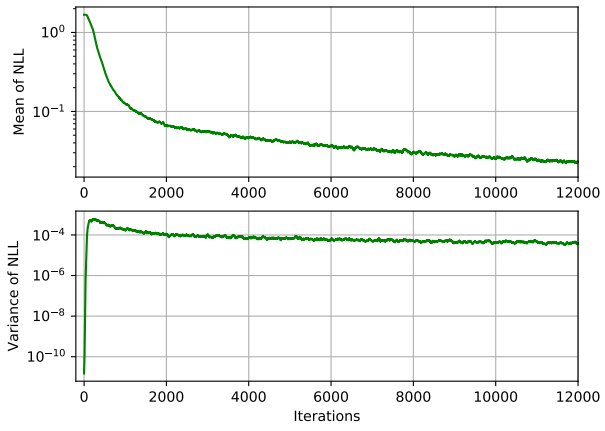

Figure 7. **Global permutation invariance.** The curves, corresponding to the model of permutation of MNIST digits in Section 4.1, show estimates of the mean and variance of the posterior NLL under global permutations, as a function of the learning iterations. As discussed in Section 2.1, the variance should be small.

C. Neural architectures in the examples

To train the networks in the examples, we used stochastic gradient descent with ADAM (Kingma & Ba, 2015), with learning rate 10^{-4}. The number of samples in each mini-batch were: 1 for $p(N)$, 1 for $p(c_{1:N})$, 64 for $p(\mathbf{x}|c_{1:N})$. The architecture of the functions in each case were:

Clusters: 2D Gaussians

- h: MLP [2-128-128-128-128-256] with ReLUs
- g: MLP [256-128-128-128-128-512] with ReLUs
- f: MLP [1024-128-128-128-128-1] with ReLUs

Clusters: MNIST

- h: 2 layers of [convolutional + maxpool + ReLU] + MLP [256-256] with ReLUs
- h and f: same as above

Communities: IRL

- t: MLP [6-64-64-64-256] with ReLUs
- h: MLP [256-64-64-64-256] with ReLUs
- g: MLP [256-64-64-64-256] with ReLUs
- f: MLP [1024-64-64-64-64-1] with ReLUs

Permutations: 2D

- h: MLP [4-64-64-64-256] with ReLUs
- $g_x = g_y$: MLP [2-64-64-64-256] with ReLUs
- f: MLP [1024-64-64-64-64-1] with ReLUs

Permutations: MNIST

- e: feature extractor with 2 layers of [convolutional + maxpool + ReLU] + MLP [256-256] with ReLUs
- h: MLP [256-64-64-64-256] with ReLUs
- $g_x = g_y$: MLP [256-64-64-64-256] with ReLUs
- f: MLP [1024-64-64-64-64-1] with ReLUs