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 random permutations, probabilistic clustering (such as Dirichlet process mixture models) and random communities (such as stochastic block models). The approach is based on mapping distributed, symmetry-invariant representations of discrete arrangements into conditional probabilities. The resulting algorithms parallelize easily, yield iid samples from the approximate posteriors, 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 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 conditional 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 a generating process of the form

\[
c_1 \ldots c_N \sim p(c_1, \ldots, c_N)
\]
\[
\mu_1 \ldots \mu_K \sim p(\mu_1, \ldots, \mu_K)
\]
\[
x_i \sim p(x_i|\mu_{c_i}) \quad i = 1 \ldots N
\]

The number of clusters $K$ is a random variable, indicating the number of distinct values among the sampled $c_i$'s, and $\mu_k$ denotes a parameter vector controlling the distribu-
We are interested in approximating (2) with a neural network as

\[ p(c_1, \ldots, c_n|\mathbf{x}) \simeq \varphi(c_n, c_{1:n-1}, \mathbf{x}). \]  

(3)

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,

\[ p(c_1, \ldots, c_n|\mathbf{x}). \]  

(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

\[ H_k = \sum_{i: c_i = k} h(x_i), \quad k = 1 \ldots K, \]  

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

\[ G = \sum_{k=1}^{K} g(H_k), \]  

(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

\[ Q = \sum_{i=n+1}^{N} h(x_i). \]  

(7)

Note that \( G \) and \( Q \) provide fixed-dimensional, symmetry-invariant representations of all the assigned and non-assign 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

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

(8)
for $k = 1 \ldots K + 1$, where $h_n = h(x_n)$ and we defined a new function $f$ that takes real values. While this formula may appear somewhat arbitrarily chosen, arguments similar to those in (Zaheer et al., 2017) establish that this family of distributions is general enough to yield arbitrarily good approximations to any function that satisfies the three symmetries discussed above.

In eq. (8), $\theta$ 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 $\theta$ of the neural networks, we use stochastic gradient descent to minimize the expected negative log-likelihood,

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

Samples from $p(c_{1:N}, x)$ are obtained from the generative model, irrespective of the model being conjugate. Here $p(N)$ is a prior over positive integers, and can be chosen by the user according to the expected sizes of the datasets on which inference is to be made.

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

2.1. Global symmetry from exchangeability

The assumption that the prior $p(c_{1:N})$ is exchangeable 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. In the Appendix we provide details of the network architectures used in all the examples of the paper.
We consider next a DPMM over the empirical distribution with \( \alpha = 0.7 \) 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.

**Clustering in 2D Gaussian models**

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

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

where CRP stands for the Chinese Restaurant Process, with \( \alpha = 0.7, \sigma = 1, \) 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

\[
\begin{align*}
N & \sim \text{Uniform}[5, 100] \\
c_{1:N} & \sim \text{CRP}(\alpha) \\
l_k & \sim \text{Unif}[0, 9] \quad k = 1 \ldots K \\
x_i & \sim \text{Unif}[\text{MNIST digits with label } l_{c_i}] \quad i = 1 \ldots N
\end{align*}
\]

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.

**3. Communities**

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

\[
\begin{align*}
c_1 \ldots c_N & \sim p(c_1, \ldots, c_N) \\
\phi_{k_1,k_2} & \sim \text{Beta}(\alpha, \beta) \quad k_1 \leq k_2 \\
x_{i,j} & \sim \text{Bernoulli}(\phi_{c_i,c_j}) \quad i \leq j, \quad i, j = 1 \ldots N
\end{align*}
\]

where \( k_1, k_2 = 1 \ldots K. \) The prior \( p(c_{1:n}) \) 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 \( x_{i,j} = x_{j,i}, \) and assume \( x_{ij} \in \{+1, -1\}. \)

Examples of this family of models include stochastic block models (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

\[
x = (x_{i,1}, \ldots, x_{i,N}) \quad i = 1 \ldots N.
\]

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 \( h(x_i) \) would ignore the permutation symmetries present among the components of \( x_i \) as a result of the \( c_{1:n} \) assignments. Moreover, a fixed function \( h(x_i) \) would not be able to accommodate
the fact that the length of $x_i$ changes with the size $N$ of the dataset.

Suppose that there are $K$ clusters among the $c_{1:n}$, each with $n_k$ 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 $n_{K+1} = N - n$, so we assume $c_{n+1:N} = K + 1$.

The $n_k$ components of $x_i$ associated with each of the $K + 1$ clusters can be split into $n^+_j,k$ values $+1$ and $n^-_j,k$ values $-1$, and these two numbers are the only invariants of $x_i$ under the symmetry of permuting the indices within cluster $k$.

Additional invariants can be obtained combining $n^+_j,k$ and $n^-_j,k$ across all $x_j$’s with $c_j = c_i$, such as

$$m^+_{c_i,k} = \frac{1}{n_{c_i}} \sum_{j:c_j=c_i} n^+_j,k$$

$$m^-_{c_i,k} = \frac{1}{n_{c_i}} \sum_{j:c_j=c_i} (n^+_j,k - m^+_{c_i,k})^2$$

and similarly $m^-_{c_i,k}$ and $v^-_{c_i,k}$. We propose to encode the components of $x_i$ belonging to cluster $k$ as

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

The motivation to consider these invariants is that, if the partition corresponding to $c_{1:n}$ is correct, then for $i \leq n$ and $k \leq K$ we have $n^+_{i,k} \approx m^+_{c_i,k}$ 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 $v^+_{c_i,k} \approx 0$. Similarly for $m^-_{c_i,k}$ and $v^-_{c_i,k}$.

Therefore, these values provide learning signals to the network that estimates the probability of the assignments $c_{1:n}$ being correct.

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

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

where the encoding function is $t : \mathbb{R}^6 \rightarrow \mathbb{R}^{dt}$. 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 $x_i$ of the adjacency matrix is represented by the fixed-dimensional pair $(t_i, q_i)$ in a way that reflects the symmetries of the assignments $c_{1:n}$.

### 3.2. Clustering the rows of the adjacency matrix

We can proceed now as in regular clustering, encoding each cluster of $x_i$’s within $c_{1:n}$ as

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

with the permutation invariant, fixed-dimensional vectors

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

$$Q = \sum_{i=n+1}^{N} h(t_i, q_i). \quad (17)$$
In terms of these quantities, the conditional probabilities are defined as usual as

\[
p(y_n | x_{1:n-1}, x) = \frac{e^{f(G_k, Q, h_n)}}{\sum_{k'=1}^{K+1} e^{f(G_{k'}, Q, h_n)}}
\]  

(18)

for \( k = 1 \ldots K + 1 \), 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, \( i, h, g \) 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

\[
N \sim \text{Uniform}[5, 100] \\
c_{1:N} \sim \text{CRP}(\alpha) \\
\phi_{k1,k2} \sim \text{Beta}(0.2, 0.2) \quad k_1 \leq k_2 \\
x_{i,j} \sim \text{Bernoulli}(\phi_{c_i,c_j}), \ i \leq j \quad i,j = 1 \ldots N
\]

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

\[
c_1 \ldots c_N \sim p(c_1, \ldots, c_N) \\
x_1 \ldots x_N \sim p(x_1, \ldots, x_N) \\
\mu_1 \ldots \mu_N \sim p(\mu_1, \ldots, \mu_N) \\
y_i \sim p(y_i|x_{c_i}, \mu_i) \quad i = 1 \ldots N.
\]

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

\[
c_i \in \{1, \ldots, N\}, \ c_i \neq c_j \text{ for } i \neq j
\]

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 \( \mu_i \) is a parameter that controls the conditional distribution of \( y_i \) given \( x_{c_i} \). The distributions \( p(c_{1:N}) \), \( p(x_{1:N}) \) and \( p(\mu_{1:N}) \) are assumed to be exchangeable.

Given two sets of \( N \) data points \( x = \{x_1\}, y = \{y_1\} \), we are interested in sampling the posterior of the \( c_i \)'s, using a decomposition

\[
p(c_{1:N} | x, y) = p(c_1 | x, y)p(c_2 | c_1, x, y) \ldots p(c_N | c_{1:N-1}, x, y)
\]

(19)

\[
p(c_n | c_{1:n-1}, x, y) = \frac{p(c_1 \ldots c_n | x, y)}{\sum_{i=1}^{n} p(c_1, \ldots, c_n = k_i | x, y)}
\]

(20)

Our interest is again a generic factor in (19)

\[
p(c_n | c_{1:n-1}, x, y) = \frac{p(c_1 \ldots c_n | x, y)}{\sum_{i=1}^{n} p(c_1, \ldots, c_n = k_i | x, y)}
\]

(21)

The symmetries of this expression are:

- **Permutations among the paired variables:** (21) is invariant under permutations of the assigned pairs \((y_i, x_{c_i})\). This invariance can be encoded as

\[
H = \sum_{i=1}^{n} h(y_i, x_{c_i})
\]

(22)

where \( h : \mathbb{R}^{d_x+d_y} \rightarrow \mathbb{R}^{d_h} \).

- **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

\[
G_y = \sum_{i=n+1}^{N} g_y(y_i),
\]

(23)

\[
G_x = \sum_{i=1}^{N-n} g_x(x_{k_i}),
\]

(24)

where \( g_x : \mathbb{R}^{d_x} \rightarrow \mathbb{R}^{d_y} \) and \( g_y : \mathbb{R}^{d_y} \rightarrow \mathbb{R}^{d_y} \).

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 \( G_{x,i} \) (\( i = 1, \ldots, N-n+1 \)). In terms of these vectors, we propose to model (20) as

\[
p(y_n | x_{1:n-1}, x, y) = \frac{e^{f(H_{i}, G_{x,i}, G_{y,i}, h_{n,i})}}{\sum_{j=1}^{N-n+1} e^{f(H_{i}, G_{x,j}, G_{y,j}, h_{n,j})}}
\]

where \( h_{n,i} \equiv h(y_n, x_{k_i}) \), for \( i = 1 \ldots N-n+1 \). We defined the function \( f \) that takes real values, and \( \theta \) 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).
Discrete Neural Processes

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 \( N(x_i, 0.612) \). 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.

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:

\[
\begin{align*}
  c_{0:N} & \sim \text{Unif [permutations]} \\
  x_i & \sim N(0, 3) \quad i = 1 \ldots N \\
  y_i & \sim N(x_{c_i}, 0.612) \quad i = 1 \ldots N
\end{align*}
\]

Figure 4 shows results for this model.

**Permutation of MNIST digits**

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

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

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

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

The work (Du, 2010) provides an overview of deterministic clustering 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 &
In 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 specialized for community detection on graphs have been studied in (Chen et al., 2019) as a classification problem for every node over a predetermined number of clusters.

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).

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).

6. Outlook

The method we introduced can potentially be applied to many other discrete model settings. 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 (Wallach et al., 2010; Blei & Frazier, 2011; 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. This work was supported by the Simons Foundation, the DARPA NESD program, and by ONR N00014-17-1-2843 and NIH/NIBIB R01 EB22913.
References


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


A. The NPP Algorithm

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

1: \(r_i, g_i \leftarrow g_y(y_i), g_x(x_i)\) \(i = 1 \ldots N\) \{Notation\}
2: \(H \leftarrow 0\)
3: \(G_y \leftarrow \sum_{i=1}^{N} r_i\)
4: \(G_x \leftarrow \sum_{i=1}^{N} q_i\)
5: for \(n = 1 \ldots N - 1\) do
6: \(G_y \leftarrow G_y - r_i\) \{Remove from unpaired set\}
7: if \(n = 1\) then
8: \(\{k_i\}_{i=1}^{N} \leftarrow \{1, \ldots, N\}\) \{Available indices\}
9: else
10: \(\{k_i\}_{i=1}^{N-n+1} \leftarrow \{1, \ldots, N\}/\{c_{1:n-1}\}\)
11: end if
12: for \(i = 1 \ldots N - n + 1\) do
13: \(G_x \leftarrow G_x - q_{k_i}\) \{Remove from unpaired set\}
14: \(H \leftarrow H + h(y_{k_i}, x_{k_i})\) \{Add to paired set\}
15: \(p_i \leftarrow e^{f(\sum_{i'=1}^{n-1} p_{i'})}\) \{Normalize probabilities\}
16: end for
17: \(i \sim p_i\) \{Sample assignment for \(y_n\)\}
18: \(c_n \leftarrow k_i\)
19: \(G_x \leftarrow G_x - q_{c_n}\) \{Remove from unpaired set\}
20: \(H \leftarrow H + h(y_n, x_{c_n})\) \{Add pair to paired set\}
21: end for
22: Return \(c_1 \ldots c_{N-1}\)

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 (± 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 $p(x|c_1:N)$. The architecture of the functions in each case were:

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

Clusters: MIST
- $h$: 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 [1024-64-64-64-1] 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 [1024-64-64-64-1] with ReLUs

Permutations: MIST
- $e$: 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 [1024-64-64-64-1] with ReLUs