Neural Clustering Processes

Ari Pakman [1] Yueqi Wang [1 2] Catalin Mitelut [1] JinHyung Lee [1] Liam Paninski [1]

Abstract

Probabilistic clustering models (or equivalently, mixture models) are basic building blocks in countless statistical models and involve latent random variables over discrete spaces. For these models, posterior inference methods can be inaccurate and/or very slow. In this work we introduce deep network architectures trained with labeled samples from any generative model of clustered datasets. At test time, the networks generate approximate posterior samples of cluster labels for any new dataset of arbitrary size. We develop two complementary approaches to this task, requiring either O(N) or O(K) network forward passes per dataset, where N is the dataset size and K the number of clusters. Unlike previous approaches, our methods sample the labels of all the data points from a well-defined posterior, and can learn nonparametric Bayesian posteriors since they do not limit the number of mixture components. As a scientific application, we present a novel approach to neural spike sorting for high-density multielectrode arrays.

1. Introduction

Probabilistic clustering models (or equivalently, mixture models) are a staple of statistical modelling in which a discrete latent variable is introduced for each observation, indicating its mixture component identity. Popular inference methods in these models fall into two main classes. When exploring the full posterior is crucial (e.g. there is irreducible uncertainty about the latent structure or many separate local optima exist), the method of choice is Markov Chain Monte Carlo (MCMC) (Neal, 2000; Jain & Neal, 2004). This method is 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. Alternatively, variational methods (Blei & Jordan, 2004; Kurihara et al., 2007; Hughes et al., 2015) are typically much faster but do not come with accuracy guarantees.

As a third alternative, in recent years there has been steady progress on amortized inference methods, and such is the spirit of this work. Concretely, we propose novel techniques to perform amortized approximate posterior inference over discrete latent variables in mixture models. We consider two possible product expansions of the mixture posteriors, and in each expansion we use neural networks to express conditional factors in terms of fixed-dimensional, distributed representations that respect the permutation symmetries imposed by the discrete variables. A major advantage of our approach, compared to previous approaches to amortized clustering, is its ability to handle an arbitrary number of clusters from a well defined posterior. This makes the methods a natural choice for nonparametric Bayesian models, such as Dirichlet process mixture models (DPMM), and their extensions. Moreover, the methods can be applied to both conjugate and non-conjugate models.

The term 'amortization' refers to the process of investing computational resources to train a model that is later used for very fast posterior inference (Gershman & Goodman, 2014). Concretely, in a model with observations x and latent variables z, the amortized approach learns a parametrized function $q_\theta(z|x)$ that approximates $p(z|x)$ for any x; learning the parameters θ may be challenging, but once θ is in hand evaluating $q_\theta(z|x)$ for new data x is fast.

The amortized inference literature can be coarsely divided into two approaches. On one side, the variational autoencoder (VAE) approach (Kingma & Welling, 2013), with roots in the wake-sleep algorithm (Hinton et al., 1995), learns $q_\theta(z|x)$ along with the generative model $p_\phi(x|z)$. Here $p(z)$ is usually a known simple distribution.

Our work corresponds to the alternative case: a generative model $p(x, z)$ is postulated, and posterior inference is the main focus of the learning phase. Amortized methods in this case usually involve a degree of specialization to the particular generative model of interest. Examples include methods developed for Bayesian networks (Stuhlmüller et al., 2013),

[1]Columbia University [2]Now at Google. Correspondence to: Ari Pakman <aripakman@gmail.com>.

Proceedings of the 37^{th} International Conference on Machine Learning, Online, PMLR 119, 2020. Copyright 2020 by the author(s).

sequential Monte Carlo (Paige & Wood, 2016), probabilistic programming (Ritchie et al., 2016; Le et al., 2016), neural decoding (Parthasarathy et al., 2017) and particle tracking (Sun & Paninski, 2018). Our work is specialized to the case where the latent variables are discrete and their range is not fixed beforehand.

After training a neural architecture using labeled 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. These samples can be used (i) to approximate expectations, (ii) as high quality importance samples, or (iii) as independent Metropolis-Hastings proposals.

In Section 2 we introduce generative mixture models and present two distinct expansions of the posterior distribution. In Section 3 and Section 4 we present neural architectures to model the factors of each expansion, along with their objective functions. In Section 5 we present two simple examples to illustrate the methods. In Section 6 we review related works. In Section 7 we discuss quantitative evaluations of the new methods. We close in Section 8 with a neuroscientific application to spike sorting for high-density multielectrode probes. The Supplementary Material (SM) contains details on the architectures, the spike-sorting application, and an extension of these ideas to particle tracking.[1]

2. Generative Mixture Models

We start by presenting mixture models from the perspective of probabilistic models for clustering (McLachlan & Basford, 1988). The latter 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

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

Here α_1, α_2 are hyperparameters. 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,

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

where $\{\sigma_i\}$ is an arbitrary permutation of the indices, and similarly for $p(\mu_{1:K}|\alpha_2)$. Our interest in this work is in cases where K can take any value $K \leq N$, such as the Chinese Restaurant Process (CRP) or its Pitman-Yor generalization (see Rodriguez & Mueller (2013) for a review). Of course, our methods will also work for models with $K < B$ with fixed B, such as Mixtures of Finite Mixtures (Miller & Harrison, 2018).

Instead of representing configurations using N labels c_i, an alternative is obtained using K sets of *indices*:

$$\begin{aligned} \mathbf{s}_k &= (s_{k,1}, \dots, s_{k,N_k}) \qquad k = 1 \dots K\,, \\ \text{where} & \qquad \forall k, \forall i, c_{s_{k,i}} = k. \end{aligned} \tag{2}$$

For example, the labels $c_{1:6} = (1,1,2,1,2,1)$ are equivalent to $\mathbf{s}_1 = (1,2,4,6)$, $\mathbf{s}_2 = (3,5)$. Note that cluster k has size N_k and $N = \sum_{k=1}^K N_k$. Given N data points $\mathbf{x} = \{x_i\}$, we would like to draw independent samples from the posterior $p(\mathbf{c}|\mathbf{x})$. For this, we consider expanding $p(\mathbf{c}|\mathbf{x})$ using either the labels representation,

$$p(c_{1:N}|\mathbf{x}) = p(c_1|\mathbf{x})p(c_2|c_1,\mathbf{x}) \dots p(c_N|c_{1:N-1},\mathbf{x}), \tag{3}$$

or the indices representation,

$$p(\mathbf{s}_{1:K}|\mathbf{x}) = p(\mathbf{s}_1|\mathbf{x})p(\mathbf{s}_2|\mathbf{s}_1,\mathbf{x}) \dots p(\mathbf{s}_K|\mathbf{s}_{1:K-1},\mathbf{x})\,. \tag{4}$$

Note that for a given cluster configuration, $p(c_{1:N}|\mathbf{x}) = p(\mathbf{s}_{1:K}|\mathbf{x})$. In the next two Sections, we present neural architectures to model the factors in each of these expansions.

3. Pointwise Sampling

We would like to model all the factors in (3) in a unified way, with a generic factor given by

$$p(c_n|c_{1:n-1},\mathbf{x}) = \frac{p(c_1 \dots c_n, \mathbf{x})}{\sum_{c'_n=1}^{K+1} p(c_1 \dots c'_n, \mathbf{x})}. \tag{5}$$

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.

Since (5) is in general difficult to compute directly, we will approximate these terms with a neural network $q_\theta(c_n|c_{1:n-1},\mathbf{x})$, that takes as inputs $(c_{1:n-1},\mathbf{x})$, then extracts features and combines them nonlinearly to output a probability distribution on c_n. Critically, we will design the network to enforce the highly symmetric structure of (5).

To make this symmetric structure more transparent, let us consider the joint distribution of the assignments of the first n data points,

$$p(c_1, \dots, c_n, \mathbf{x})\,. \tag{6}$$

[1] An early version appeared in (Pakman & Paninski, 2018; Wang et al., 2019). Similar methods were applied to amortized permutations in (Pakman et al., 2019).

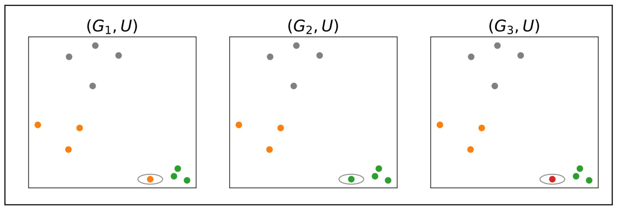

Figure 1. **Encoding cluster labels.** After assigning labels $c_{1:6}$ to $K = 2$ clusters, each of the three possible c_7 labels (for the circled point x_7) gives an encoding G_k for the set $x_{1:7}$. The vector U encodes the four gray unlabeled points (Best in color).

Note that under the model (1), this quantity depends on all the N elements of $\mathbf{x}$, not just on $x_{1:n}$. A neural representation of (6) 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 permutation-invariant representations of the observations $\mathbf{x}$. The general problem of constructing such invariant encodings was discussed recently in (Zaheer et al., 2017); to adapt this approach to our context, we consider three distinct permutation symmetries:

- **Permutations within a cluster:** (6) is invariant under permutations of x_i's in the same cluster. For each of the K clusters that have been sampled so far, we define the encoding

$$H_k = \sum_{i:c_i=k} h(x_i) \quad h : \mathbb{R}^{d_x} \to \mathbb{R}^{d_h} \tag{7}$$

  which is clearly invariant under permutations of x_i's in the same cluster. In general h is an encoding function we learn from data, unless $p(x|\mu)$ belongs to an exponential family and the prior $p(c_{1:N})$ is constant, as discussed in SM Section B.

- **Permutations between clusters:** (6) is invariant under permutations of the cluster labels. In terms of the within-cluster invariants H_k, this can be captured by

$$G = \sum_{k=1}^{K} g(H_k), \quad g : \mathbb{R}^{d_h} \to \mathbb{R}^{d_g}. \tag{8}$$

- **Permutations of the unassigned data points:** (6) is also invariant under permutations of the $N - n$ unassigned data points. This can be captured by

$$U = \sum_{i=n+1}^{N} u(x_i), \qquad u : \mathbb{R}^{d_x} \to \mathbb{R}^{d_u}. \tag{9}$$

Note that G and U provide fixed-dimensional, symmetry-invariant representations of the assigned and non-assigned data points, respectively, for any values of N and K. Encodings of this form yield arbitrarily accurate approximations of (partially) symmetric functions (Zaheer et al., 2017; Gui et al., 2019).

3.1. The Variable-input Softmax

After assigning values to $c_{1:n-1}$, each of the $K+1$ possible values for c_n corresponds to $h(x_n)$ appearing in one particular H_k in (7), and yields a separate vector G_k in (8). See Figure 1 for an example. In terms of the G_k's and U, we propose to model (5) as

$$q_\theta(c_n = k | c_{1:n-1}, \mathbf{x}) = \frac{e^{f(G_k, U)}}{\sum_{k'=1}^{K+1} e^{f(G_{k'}, U)}} \tag{10}$$

with $k = 1 \dots K+1$, where we have introduced a new real-valued function f. In other words, each value of c_n corresponds to a different channel through which the encoding $h(x_n)$ flows to the logit value f. Note that $k = K+1$ corresponds to c_n forming its own new cluster with $H_k = h(x_n)$.

Our softmax (10) differs from the usual form in, e.g., classification networks, where a fixed number of categories receive logit values f from the fixed-size final layer of a multi-layer perceptron (MLP). In our case, the discrete identity of each logit is determined by the neural path that the input $h(x_n)$ takes to G, thus allowing a flexible number of categories.

In eq. (10), θ denotes the parameters in the functions h, g, u and f, which we represent with neural networks. By storing and updating G and U for successive values of n, as shown in Algorithm 1, the computational cost of a full i.i.d. sample of $c_{1:N}$ is $O(NK)$, the same as a single Gibbs sweep; and by parallelizing steps 8-9 in Algorithm 1, the number of network forward passes becomes $O(N)$. We term this approach Neural Clustering Process (NCP). It is relatively easy to run hundreds of copies of Algorithm 1 in parallel on a GPU, with each copy yielding a different set of samples $c_{1:N}$.[2]

3.2. Objective Function

In order to train the neural networks, we use stochastic gradient descent to minimize the expected KL divergence,

$$\mathbb{E}_{p(N)p(\mathbf{x})} KL(p(c|\mathbf{x}) \| q_\theta(c|\mathbf{x})) = \tag{11}$$
$$-\mathbb{E}_{p(N)p(c_{1:N},\mathbf{x})} \left[\sum_{n=2}^{N} \log q_\theta(c_n | c_{1:n-1}, \mathbf{x}) \right] + \text{const.}$$

Samples from $p(c_{1:N}, \mathbf{x})$ are obtained from the generative model, irrespective of the model being conjugate. In cases with unlimited samples (such as the 2D Gaussian example in Section 5 and the spike-sorting application in Section 8), we can potentially train a neural network to approximate $p(c_n | c_{1:n-1}, \mathbf{x})$ arbitrarily accurately.

The objective function (11) can be seen as a form of Expectation Propagation (Minka, 2001), as opposed

[2]Implementation available at `https://github.com/aripakman/neural_clustering_process`

Algorithm 1 $O(NK)$ Neural Clustering Process

```
1:  $h_i \leftarrow h(x_i),\ u_i \leftarrow u(x_i)$      $i = 1 \dots N$ {Notation}
2:  $U \leftarrow \sum_{i=2}^N u_i,\ \ K \leftarrow 1$   {Initialize unassigned set}
3:  $H_1 \leftarrow h_1, G \leftarrow g(H_1), c_1 \leftarrow 1$          {First cluster}
4:  for $n \leftarrow 2 \dots N$ do
5:    $U \leftarrow U - u_n$   {Remove $x_n$ from unassigned set}
6:    $H_{K+1} \leftarrow 0$                {We define $g(0) = 0$}
7:    for $k \leftarrow 1 \dots K+1$ do
8:      $G_k \leftarrow G + g(H_k + h_n) - g(H_k)$     {Add $x_n$}
9:      $q_k \leftarrow e^{f(G_k, U)}$
10:   end for
11:   $q_k \leftarrow q_k / \sum_{k'=1}^{K+1} q_{k'},\ \ c_n \sim q_k$        {Sample}
12:   if $c_n = K+1$ then
13:     $K \leftarrow K+1$
14:   end if
15:   $G \leftarrow G - g(H_{c_n}) + g(H_{c_n} + h_n)$ {Add point $x_n$}
16:   $H_{c_n} \leftarrow H_{c_n} + h_n$
17: end for
18: Return $c_1 \dots c_N$
```

to variational inference, which would minimize instead $KL(q_\theta(c|\mathbf{x}) \| p(c|\mathbf{x}))$. Note that the gradient acts only on the variable-input softmax q_θ, not on $p(c, \mathbf{x})$, so there is no problem of backpropagating through discrete variables (Jang et al., 2016; Maddison et al., 2016).

4. Clusterwise Sampling

While the NCP algorithm is good enough for small datasets, $O(N)$ forward calls might be too many for large datasets. We consider now an $O(K)$ alternative, based on modeling the factors in the clusterwise expansion (4),

$$p(\mathbf{s}_{1:K}|\mathbf{x}) = p(\mathbf{s}_1|\mathbf{x})p(\mathbf{s}_2|\mathbf{s}_1, \mathbf{x}) \dots p(\mathbf{s}_K|\mathbf{s}_{1:K-1}, \mathbf{x}) \,. \quad (12)$$

Sampling from $p(\mathbf{s}_k|\mathbf{s}_{1:k-1}, \mathbf{x})$ can be done in two steps:

1. Sample uniformly an index d_k from the set $I_k = \{1 \dots N\} \backslash \{\mathbf{s}_{1:k-1}\}$ of available indices (those not taken by $\mathbf{s}_{1:k-1}$). The point x_{d_k} becomes the first element of cluster k.

2. Denote by $\mathbf{a}_k = (a_1 \dots a_{m_k})$ the elements of the set of remaining indices $I_k \backslash \{d_k\}$, where $m_k = |I_k \backslash \{d_k\}|$. Conditioned on $(d_k, \mathbf{s}_{1:k-1}, \mathbf{x})$, sample a binary vector

$$\mathbf{b}_k = (b_1 \dots b_{m_k}) \qquad \in \{0,1\}^{m_k}$$

with $b_i = 1$ if the point x_{a_i} joins cluster k.

These two steps (see Figure 2 for an example) are iterated until there are no available indices left, and have probability

$$p(d_k, \mathbf{b}_k|\mathbf{s}_{1:k-1}, \mathbf{x}) = p(d_k|\mathbf{s}_{1:k-1})p(\mathbf{b}_k|d_k, \mathbf{s}_{1:k-1}, \mathbf{x}) \quad (13)$$

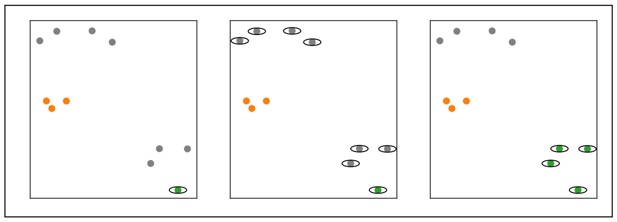

Figure 2. **Clusterwise sampling.** *Left*: After sampling cluster $\mathbf{s}_1$ (orange), the first element of $\mathbf{s}_2$, d_2, is sampled uniformly (green). *Middle:* All unassigned points $\mathbf{a}_2$ (grey) are candidates to join d_2. *Right:* By sampling $\mathbf{b}_2$, cluster $\mathbf{s}_2$ is completed. (Best in color).

where

$$p(d_k|\mathbf{s}_{1:k-1}) = \begin{cases} 1/|I_k| & \text{for } d_k \in I_k \,, \\ 0 & \text{for } d_k \notin I_k \,, \end{cases}$$

and $|I_k| = m_k + 1$. The event indicated by $\mathbf{s}_k$ is the union of N_k disjoint events $(d_k, \mathbf{b}_k)$, and we have

$$p(\mathbf{s}_k|\mathbf{s}_{1:k-1}, \mathbf{x}) = \frac{1}{|I_k|} \textstyle\sum_{d_k \in \mathbf{s}_k} p(\mathbf{b}_k|d_k, \mathbf{s}_{1:k-1}, \mathbf{x}) \quad (14)$$

where $\mathbf{b}_k$ has a '1' for each element in $\mathbf{s}_k$ except d_k. Our major challenge is therefore to model the conditional $p(\mathbf{b}_k|d_k, \mathbf{s}_{1:k-1}, \mathbf{x})$, which we address next.

4.1. Factorized posterior

The information contained in $(d_k, \mathbf{s}_{1:k-1}, \mathbf{x})$, is better represented by splitting the dataset as $\mathbf{x}_k = (\mathbf{x_a}, x_{d_k}, \mathbf{x_s})$, where

$\mathbf{x_a} = (x_{a_1} \dots x_{a_{m_k}})$	m_k available points for cluster k
x_{d_k}	First data point in cluster k
$\mathbf{x_s} = (\mathbf{x}_{\mathbf{s}_1} \dots \mathbf{x}_{\mathbf{s}_{k-1}})$	Points already assigned to clusters.

Thus $p(\mathbf{b}_k|\mathbf{x}_k) \equiv p(\mathbf{b}_k|d_k, \mathbf{s}_{1:k-1}, \mathbf{x})$. Note now that this factor has a form of conditional exchangeability

$$p(b_1 \dots b_{m_k}|x_{a_1}, \dots, x_{a_{m_k}}, x_{d_k}, \mathbf{x_s}) = \\ p(b_{\sigma_1} \dots b_{\sigma_{m_k}}|x_{\sigma_{a_1}} \dots x_{\sigma_{a_{m_k}}}, x_{d_k}, \mathbf{x_s}) \,,$$

where σ is an arbitrary permutation of the elements of $\mathbf{b}_k$ and $\mathbf{x_a}$. Based on this, we assume a conditional version of de Finetti's theorem and propose[3]

$$p(\mathbf{b}_k|\mathbf{x}_k) \simeq \int d\mathbf{z}_k \prod_{i=1}^{m_k} p_i(b_i|\mathbf{z}_k, \mathbf{x}_k) p(\mathbf{z}_k|\mathbf{x}_k) \,, \quad (15)$$

[3]More precisely, de Finetti's theorem (de Finetti, 1931; Hewitt & Savage, 1955) holds for infinite sequences. For finite sequences, as in our case, the result has been shown to hold only approximately and for discrete variables, both in the unconditional (Diaconis, 1977; Diaconis & Freedman, 1980) and conditional cases (Christandl & Toner, 2009).

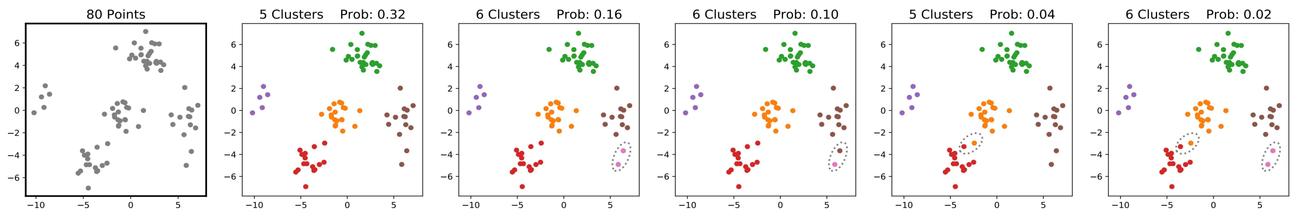

Figure 3. **Mixture of 2D Gaussians:** Given the observations in the first panel, we show samples from the NCP posterior. Note that less-reasonable samples are assigned lower probability by the NCP. The dotted ellipses indicate departures from the first, highest-probability sample. Our GPU implementation gives thousands of samples in less than a second. CCP results are similar. (Best in color.)

and approximate the integrands as

$$p_\theta(\mathbf{z}_k|\mathbf{x}_k) = \mathcal{N}(\mathbf{z}_k|\mathbf{x}_k) \quad (16)$$

$$p_{\theta,i}(b_i|\mathbf{z}_k,\mathbf{x}_k) = \text{sigmoid}[\rho_i(\mathbf{z}_k,\mathbf{x}_k)]\,. \quad (17)$$

Crucially, the posterior distributions of the b_i's are conditionally independent. Therefore, after sampling $p(\mathbf{z}_k|\mathbf{x}_k)$, all the b_i's can be sampled in parallel. Thus, while a full sample of (12) of course has cost $O(N)$, the heaviest computational burden, from network evaluations, scales as $O(K)$, since each factor in (12) needs $O(1)$ forward calls. As in NCP, we can get hundreds of full samples via GPU parallelization.

To summarize, the elements of $\mathbf{s}_k$ are generated in a process with latent variables $d_k, \mathbf{z}_k$ and joint distribution

$$p_\theta(\mathbf{s}_k,\mathbf{z}_k,d_k|\mathbf{s}_{1:k-1},\mathbf{x}) = p_\theta(\mathbf{b}_k|\mathbf{z}_k,\mathbf{x}_k)p_\theta(\mathbf{z}_k|\mathbf{x}_k)p(d_k|\mathbf{s}_{1:k-1})$$

where

$$p_\theta(\mathbf{b}_k|\mathbf{z}_k,\mathbf{x}_k) = \textstyle\prod_{i=1}^{m_k} p_{\theta,i}(b_i|\mathbf{z}_k,\mathbf{x}_k)\,. \quad (18)$$

In order to learn these functions, we introduce an encoder $q_\phi(\mathbf{z}_k, d_k|\mathbf{s}_{1:k},\mathbf{x})$ to approximate the intractable posterior, and train the functions as a conditional variational autoencoder (VAE) (Sohn et al., 2015) (as we condition everything on $\mathbf{x}$). The dependence of all the functions on the components of $\mathbf{x}$ should respect the symmetries imposed by the conditioning clusters $\mathbf{s}_{1:k-1}$ (or $\mathbf{s}_{1:k}$ for q_ϕ). This can be achieved using encodings similar to those we used above in Section 3; see SM Section A for details.

Let us stress the double role of $p_\theta(\mathbf{z}_k|\mathbf{x}_k)p(d_k|\mathbf{s}_{1:k-1})$ and $p_\theta(\mathbf{b}_k|\mathbf{z}_k,\mathbf{x}_k)$. In the VAE framework, they are the priors and likelihood of a generative model for $\mathbf{s}_k$. On the other hand they represent, after $d_k, \mathbf{z}_k$ marginalization (14)-(15), a factor of the posterior expansion (12). We call this approach Clusterwise Clustering Process (CCP).

4.2. Objective Function

Similar to the NCP case in (11), we want an approximation $p_\theta(\mathbf{s}_{1:K}|\mathbf{x})$ to $p(\mathbf{s}_{1:K}|\mathbf{x})$ that maximizes

$$-\mathbb{E}_{p(\mathbf{x})} KL[p(\mathbf{s}_{1:K}|\mathbf{x})||p_\theta(\mathbf{s}_{1:K}|\mathbf{x})] \quad (19)$$
$$= \mathbb{E}_{p(\mathbf{x},\mathbf{s}_{1:K})} \textstyle\sum_{k=1}^{K} \log p_\theta(\mathbf{s}_k|\mathbf{s}_{1:k-1},\mathbf{x}) + \text{const.}$$

where we expanded $p_\theta(\mathbf{s}_{1:K}|\mathbf{x})$ as in (12). Using now the variational posterior q_ϕ, we can bound (19) from below, which leads us to maximize the ELBO

$$\mathbb{E}_{p(\mathbf{x},\mathbf{s}_{1:K})} \sum_{k=1}^{K} \mathbb{E}_{q_\phi(\mathbf{z}_k,d_k|\mathbf{s}_{1:k},\mathbf{x})} \log \left[\frac{p_\theta(\mathbf{s}_k,\mathbf{z}_k,d_k|\mathbf{s}_{1:k-1},\mathbf{x})}{q_\phi(\mathbf{z}_k,d_k|\mathbf{s}_{1:k},\mathbf{x})} \right]$$

To use the reparametrization trick (Kingma & Welling, 2013), we use a Gumbel-Softmax relaxation for d_k (Jang et al., 2016; Maddison et al., 2016). See SM Section A.

4.3. Estimating sample probabilities

Unlike NCP, CCP samples do not come with a probability estimate. The latter can be estimated using (12) and

$$p(\mathbf{b}_k|\mathbf{x}_k) \simeq \frac{1}{M}\sum_{j=1}^{M} p_\theta(\mathbf{b}_k|\mathbf{z}_{k,j},\mathbf{x}_k) \quad (20)$$

where $\mathbf{z}_{k,j} \sim p_\theta(\mathbf{z}_k|\mathbf{x}_k)$.

5. Examples

2D Gaussian models: The generative model is

$$\begin{array}{lll} \alpha \sim \text{Exp}(1) & c_{1:N} \sim \text{CRP}(\alpha) & \mu_k \sim N(0,\sigma_\mu^2\mathbf{1}_2) \\ N \sim \text{Uniform}[5,100] & & x_i \sim N(\mu_{c_i},\sigma^2\mathbf{1}_2) \end{array}$$

where CRP stands for the Chinese Restaurant Process, with concentration parameter α, $\sigma_\mu = 10$, and $\sigma = 1$. Figure 3 shows that the NCP captures the posterior uncertainty inherent in clustering this data. Since we have unlimited samples, there is no distinction here between training and test sets.

MNIST digits: We consider next a DPMM over MNIST digits, with generative model

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

where CRP_{10} is a Chinese Restaurant Process truncated to up to 10 clusters, and $d_x = 28 \times 28$. Training was

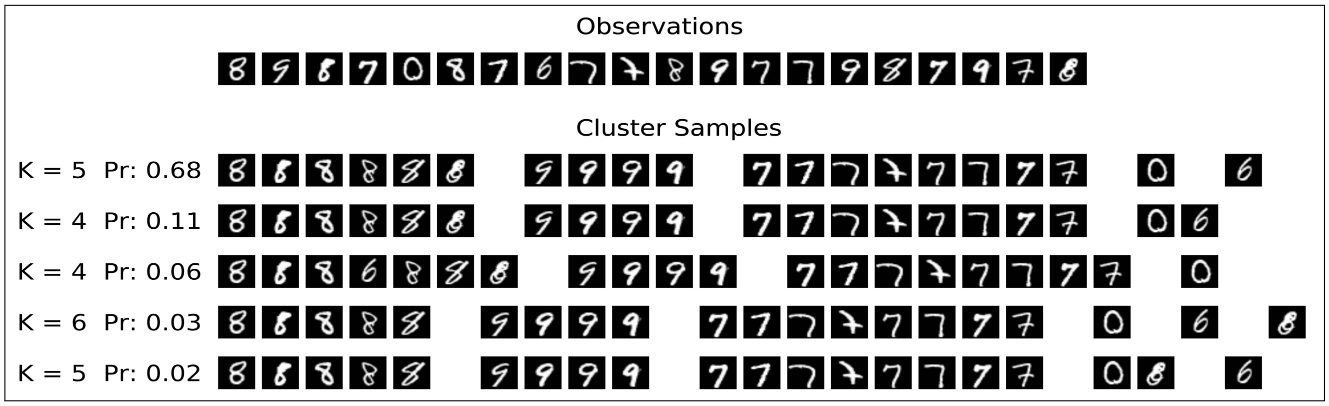

Figure 4. **NCP trained on MNIST clusters.** Top row: 20 images from the MNIST test set. Below: five samples of $c_{1:20}$ from the NCP posterior. Note that each sample captures some ambiguity suggested by the form of particular digits. CCP results are similar.

performed by sampling x_i from the MNIST training set. Figure 4 shows posterior samples for a set of digits from the MNIST test set, illustrating how the estimated model correctly captures the shape ambiguity of some of the digits. Note that in this case the generative model has no analytical expression, but this presents no problem; a set of labelled samples is all we need for training. See SM Section G for details of the network architectures used.

6. Related works

Most works on neural network-based clustering focus on learning features as inputs to traditional clustering algorithms, as reviewed in (Du, 2010; Aljalbout et al., 2018; Min et al., 2018). Our approach differs from these works because it leverages deep learning to improve *algorithmic* aspects of clustering, via amortization.

Permutation-invariant neural architectures have been explored recently in (Ravanbakhsh et al., 2017; Korshunova et al., 2018; Lee et al., 2018; Bloem-Reddy & Teh, 2019; Wagstaff et al., 2019). The representation of a set via a sum (or mean) of encoding vectors was also used in (Guttenberg et al., 2016; Ravanbakhsh et al., 2016; Edwards & Storkey, 2017; Zaheer et al., 2017; Garnelo et al., 2018a; Kim et al., 2019).

A conditional form of de Finetti's theorem was also assumed for Neural Processes (NP) (Garnelo et al., 2018b), but differs from our assumed form in (15) in that our prior $p_\theta(\mathbf{z}_k|\mathbf{x}_k)$ depends symmetrically on the available points $\mathbf{x_a}$, in order to keep the correct dependency of the marginal $p(\mathbf{c}_{1:n}, \mathbf{x})$ on all the N components of $\mathbf{x}$, while for NPs the prior is independent of the available data points.

Amortized inference of Gaussian mixtures has been studied recently in (Le et al., 2016; Lee et al., 2018; Kalra et al., 2019). In these works the output of the network are the mixture parameters instead of sampled discrete labels, and the number of components is either bounded (Le et al., 2016) or fixed (Lee et al., 2018; Kalra et al., 2019). Closer to our CCP is the DAC approach (Lee et al., 2019), that uses the set attention mechanism of (Lee et al., 2018) in the encoder to iteratively isolate and eliminate one cluster per iteration, in $O(K)$ network evaluations. But the clusters have no clear interpretation in terms of the generative model, as they come from hard thresholding of sigmoids and the eliminated clusters do not appear as a conditioning context to find new clusters. We summarize these comparisons in Table 1.

Property	CCP	NCP	DAC	MoG
Unlimited components	✓	✓	✓	✗
Amortized labels	✓	✓	✓	✗
Any generative model	✓	✓	✓	✗
Well defined posterior	✓	✓	✗	-
Forward passes	$O(K)$	$O(N)$	$O(K)$	$O(1)$

Table 1. **Comparing amortized clustering approaches.** We compare NCP/CCP (our methods) with DAC (Lee et al., 2019) and amortization for mixtures of Gaussians (MoG) (Le et al., 2016; Lee et al., 2018; Kalra et al., 2019).

7. Evaluations and diagnostics

The examples in Section 5 provide strong qualitative evidence that our approximations to the true posteriors in these models capture the uncertainty inherent in the observed data. But we would like to go further and ask quantitatively how well our approximations match the exact posterior. Unfortunately, for sample sizes much larger than $N = O(10)$ it is impossible to compute the exact posterior in these models. Nonetheless, there are several quantitative metrics we can examine to check the accuracy of the model output. Note that the diagnostics below that rely on the probabilistic nature of the inferred clusters are not applicable to the other

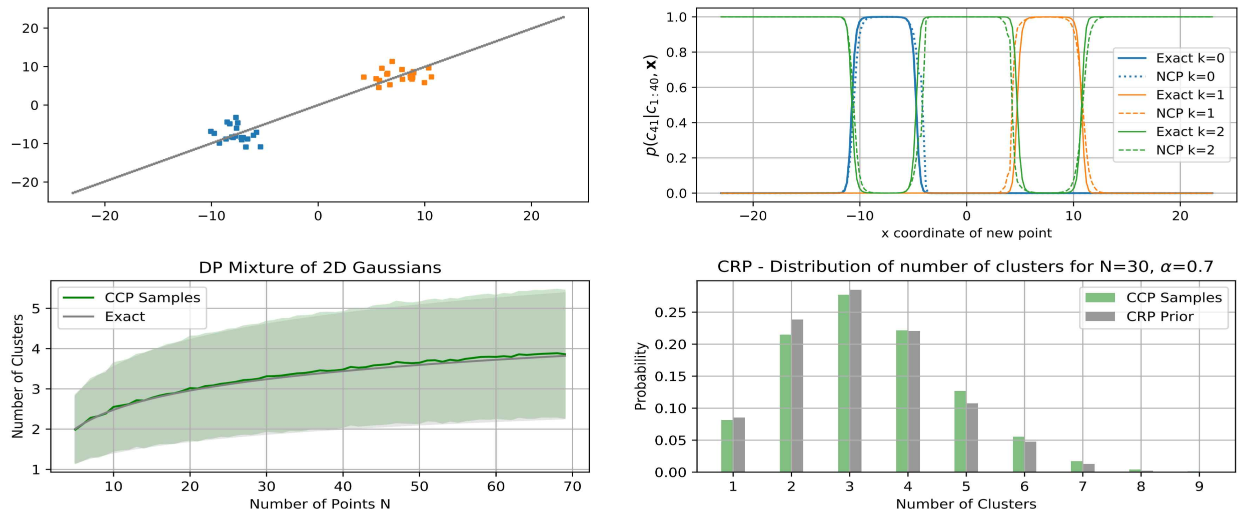

Figure 5. **Quantitative Evaluations.** *Upper left:* Two clusters of 20 points each and a line over possible locations of a 41st last point. *Upper right:* Assuming the 2D model from (21), the posterior $p(c_{41}|c_{1:40}, \mathbf{x})$ can be computed exactly, and we compare it to the NCP estimate as a function of the horizontal coordinate of x_{41}, as this point moves over the gray line on the upper left panel. *Geweke's Tests. Lower left:* The curves compare the exact mean ($\pm$ std.) of the number of clusters K for different N's from the CRP prior ($\alpha = 0.7$), with CCP sampled estimates using eq. (21). *Lower right:* Similar comparison for the histogram of K for $N = 30$ points.

methods compared in Table 1.

NCP vs. CCP: The results from the two approaches were similar in all the examples we considered, such as those in Section 5. Training CPP, however, presents the usual challenges of VAEs. We found it useful to use multiple sample objectives (Burda et al., 2015) and estimate the gradient using double-reparametrization (Tucker et al., 2019).

Global symmetry from exchangeability: From the exchangeability of $p(c_{1:N}|\alpha_1)$, the expansion (3) should not depend on the order of the data points, but this symmetry is not enforced explicitly. If our model learns the conditional probabilities correctly, this symmetry should be (approximately) satisfied, as we show in SM Section C.

Estimated vs. Analytical Probabilities: Some conditional probabilities can be computed analytically and compared with the estimates output by the network; in the example shown in Figure 5, upper-right, the estimated probabilities are in close agreement with their exact values.

Geweke's Tests: A popular family of tests that check the correctness of MCMC implementations (Geweke, 2004) can also be applied in our case: verify the (approximate) identity between the prior $p(c_{1:N})$ and

$$q_\theta(c_{1:N}) \equiv \int d\mathbf{x}\, q_\theta(c_{1:N}|\mathbf{x})\, p(\mathbf{x})\,, \qquad (21)$$

where $p(\mathbf{x})$ is the marginal from the generative model. Figure 5 shows such a comparison for the 2D Gaussian DPMM from Section 5, showing excellent agreement.

Comparison with MCMC: NCP/CCP have some advantages over MCMC approaches. First, unlike MCMC, we get a probability estimate for each sample, either directly (NCP) or with minimal computation (CCP). Secondly, NCP/CCP enjoy higher efficiency, due to parallelization of iid samples. For example, in the Gaussian 2D example in eq.(21), in the time a collapsed Gibbs sampler produces one (correlated) sample, our GPU-based NCP implementation produces more than 100 iid approximate samples. Finally, NCP/CCP do not need a burn-in period.

Comparison with Variational Inference: Below we compare NCP with a variational approach on spike sorting. For 2000 spikes, the latter returned one clustering estimate in 0.76 secs., but does not properly handle the uncertainty about the number of clusters. NCP produced 150 clustering configurations in 10 secs., efficiently capturing clustering uncertainty. In addition, the variational approach requires a preprocessing step that projects the samples to lower dimensions, whereas NCP directly consumes the high-dimensional data by learning an encoder function h.

8. Application: spike sorting with NCP

Large-scale neural population recordings using multi-electrode arrays (MEA) are crucial for understanding neural circuit dynamics. Each MEA electrode reads the signals from many neurons, and each neuron is recorded by multiple nearby electrodes. As a key analysis step, spike sorting converts the raw signal into a set of spike trains belonging to individual neurons (Pachitariu et al., 2016; Chung et al., 2017; Jun et al., 2017; Lee et al., 2017; Chaure et al., 2018; Carlson & Carin, 2019). At the core of many spike sorting pipelines is a clustering algorithm that groups the detected spikes into clusters, each representing a putative neuron (Figure 6). However, clustering spikes can be challenging: (1) Spike waveforms form highly non-Gaussian clusters in spatial and temporal dimensions, and it is unclear what are the optimal features for clustering. (2) It is unknown *a priori* how many clusters there are. (3) Existing methods do not perform well on spikes with low signal-to-noise ratios (SNR) due to increased clustering uncertainty, and fully-Bayesian approaches proposed to handle this uncertainty (Wood & Black, 2008; Carlson et al., 2013) do not scale to large datasets.

To address these challenges, we propose a novel approach to spike clustering using NCP. We consider the spike waveforms as generated from a Mixture of Finite Mixtures (MFM) distribution (Miller & Harrison, 2018), which can be effectively modeled by NCP. (1) Rather than selecting arbitrary features for clustering, the spike waveforms are encoded with a convolutional neural network (ConvNet), which is learned end-to-end jointly with the NCP network to ensure optimal feature encoding. (2) Using a variable-input softmax function, NCP is able to perform inference on cluster labels without assuming a fixed or maximum number of clusters. (3) NCP allows for efficient probablistic clustering by GPU-parallelized posterior sampling, which is particularly useful for handling the clustering uncertainty of ambiguous small spikes. (4) The computational cost of NCP training can be highly amortized, since neuroscientists often sort spikes form many statistically similar datasets.

We trained NCP for spike clustering using synthetic spikes from a simple yet effective generative model that mimics the distribution of real spikes, and evaluated the spike sorting performance on labeled synthetic data, unlabeled real data and hybrid test data by comparing NCP against two other methods: (1) **vGMFM**, variational inference on Gaussian MFM (Hughes & Sudderth, 2013). (2) **Kilosort**, a state-of-the-art spike sorting pipeline described in (Pachitariu et al., 2016). In the Supplementary Material (SM) Section D, we describe the dataset, neural architecture, and the training/inference pipeline of NCP spike sorting. In SM Section E, we show that NCP spike sorting achieves high clustering quality, and matches or outperforms a state-of-the-art method on synthetic, real and hybrid data.

Probabilistic clustering of ambiguous small spikes. Sorting small spikes has been challenging due to the low SNR and increased uncertainty of cluster assignment. By efficient GPU-parallelized posterior sampling of cluster labels, NCP is able to handle the clustering uncertainty by producing multiple plausible clustering configurations. Figure 7 shows examples where NCP separates spike clusters with amplitude as low as 3-4× the standard deviation of the noise into plausible units that are not mere scaled version of each other but have distinct shapes on different channels. f

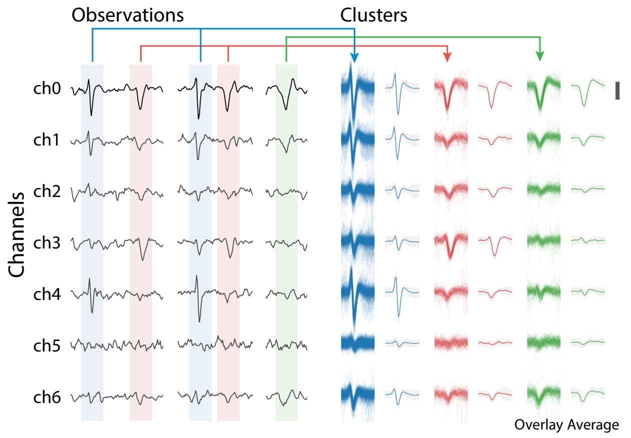

Figure 6. **Clustering multi-channel spike waveforms using NCP.** Each row is an electrode channel. Spikes of the same color belong to the same cluster. (Scale bar: 5× noise s.d.).

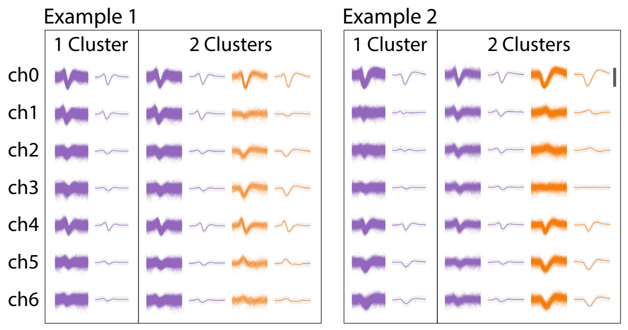

Figure 7. **Clustering ambiguous small spikes.** In both examples, multiple plausible clustering results of small spikes were produced by sampling from the NCP posterior (scale bar = 5× noise s.d.).

9. Conclusion

We introduced neural architectures to amortize posterior sampling of generative clustering models in $O(N)$ and $O(K)$ forward passes. The performance is excellent in simple examples. In a realistic spike-sorting application, our results show that NCP spike sorting provides high clustering quality, matches or outperforms a state-of-the-art method, and handles clustering uncertainty by efficient posterior sampling (a task that is not solved by currently available methods), demonstrating substantial promise for incorporating these methods into production-scale pipelines.

Acknowledgements

We thank Sean Bittner, Alessandro Ingrosso, Scott Linderman, Aaron Schein and Ruoxi Sun for helpful conversations. 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

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

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, 2004.

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

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

Calabrese, A. and Paninski, L. Kalman filter mixture model for spike sorting of non-stationary data. *Journal of neuroscience methods*, 196(1):159–169, 2011.

Carlson, D. and Carin, L. Continuing progress of spike sorting in the era of big data. *Current opinion in neurobiology*, 55:90–96, 2019.

Carlson, D. E., Vogelstein, J. T., Wu, Q., Lian, W., Zhou, M., Stoetzner, C. R., Kipke, D., Weber, D., Dunson, D. B., and Carin, L. Multichannel electrophysiological spike sorting via joint dictionary learning and mixture modeling. *IEEE Transactions on Biomedical Engineering*, 61(1):41–54, 2013.

Chaure, F. J., Rey, H. G., and Quian Quiroga, R. A novel and fully automatic spike-sorting implementation with variable number of features. *Journal of neurophysiology*, 120(4):1859–1871, 2018. doi: 10.1152/jn.00339.2018.

Chichilnisky, E. J. and Kalmar, R. S. Functional asymmetries in on and off ganglion cells of primate retina. *Journal of Neuroscience*, 22(7):2737–2747, 2002. ISSN 0270-6474. doi: 10.1523/JNEUROSCI.22-07-02737.2002. URL http://www.jneurosci.org/content/22/7/2737.

Christandl, M. and Toner, B. Finite de Finetti theorem for conditional probability distributions describing physical theories. *Journal of Mathematical Physics*, 50(4):042104, 2009.

Chung, J. E., Magland, J. F., Barnett, A. H., Tolosa, V. M., Tooker, A. C., Lee, K. Y., Shah, K. G., Felix, S. H., Frank, L. M., and Greengard, L. F. A fully automated approach to spike sorting. *Neuron*, 95(6):1381–1394, 2017.

de Finetti, B. Funzione caratteristica di un fenomeno aleatorio. *Atti della R. Academia Nazionale dei Lincei, Serie 6. Memorie, Classe di Scienze Fisiche, Mathematice e Naturale,4*, pp. 251–299, 1931.

Diaconis, P. Finite forms of de Finetti's theorem on exchangeability. *Synthese*, 36(2):271–281, 1977.

Diaconis, P. and Freedman, D. Finite exchangeable sequences. *The Annals of Probability*, pp. 745–764, 1980.

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.

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.

Gershman, S. and Goodman, N. Amortized inference in probabilistic reasoning. In *Proceedings of the annual meeting of the cognitive science society*, volume 36, 2014.

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

Graves, A. Sequence transduction with recurrent neural networks. *CoRR*, abs/1211.3711, 2012.

Gui, S., Zhang, X., Zhong, P., Qiu, S., Wu, M., Ye, J., Wang, Z., and Liu, J. Pine: Universal deep embedding for graph nodes via partial permutation invariant set functions. *arXiv preprint arXiv:1909.12903*, 2019.

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.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

Hewitt, E. and Savage, L. J. Symmetric measures on cartesian products. *Transactions of the American Mathematical Society*, 80(2):470–501, 1955.

Hinton, G. E., Dayan, P., Frey, B. J., and Neal, R. M. The" wake-sleep" algorithm for unsupervised neural networks. *Science*, 268(5214):1158–1161, 1995.

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.

Hughes, M. C. and Sudderth, E. Memoized online variational inference for dirichlet process mixture models. In *Advances in Neural Information Processing Systems 26*, pp. 1133–1141. 2013.

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.

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

Jun, J. J., Mitelut, C., Lai, C., Gratiy, S. L., Anastassiou, C. A., and Harris, T. D. Real-time spike sorting platform for high-density extracellular probes with ground-truth validation and drift correction. *bioRxiv*, 2017.

Kalra, S., Adnan, M., Taylor, G., and Tizhoosh, H. Learning permutation invariant representations using memory networks. *arXiv preprint arXiv:1911.07984*, 2019.

Kim, H., Mnih, A., Schwarz, J., Garnelo, M., Eslami, A., Rosenbaum, D., Vinyals, O., and Teh, Y. W. Attentive neural processes. *ICLR*, 2019.

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

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

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.

Lee, J., Lee, Y., Kim, J., Kosiorek, A. R., Choi, S., and Teh, Y. W. Set transformer. *arXiv preprint arXiv:1810.00825*, 2018.

Lee, J., Lee, Y., and Teh, Y. W. Deep Amortized Clustering. *arXiv:1909.13433*, 2019.

Lee, J. H., Carlson, D. E., Razaghi, H. S., Yao, W., Goetz, G. A., Hagen, E., Batty, E., Chichilnisky, E., Einevoll, G. T., and Paninski, L. Yass: Yet another spike sorter. In *Advances in Neural Information Processing Systems*, pp. 4002–4012, 2017.

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.

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.

Minka, T. P. Expectation propagation for approximate Bayesian inference. In *Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence*, pp. 362–369. Morgan Kaufmann Publishers Inc., 2001.

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

Pachitariu, M. Kilosort2. `https://github.com/MouseLand/Kilosort2`, 2019.

Pachitariu, M., Steinmetz, N., Kadir, S., Carandini, M., and Harris, K. D. Kilosort: realtime spike-sorting for extracellular electrophysiology with hundreds of channels. *BioRxiv*, pp. 061481, 2016.

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

Pakman, A. and Paninski, L. Amortized Bayesian inference for clustering models. *BNP@NeurIPS 2018 Workshop All of Bayesian Nonparametric*, 2018.

Pakman, A., Wang, Y., and Paninski, L. Neural Permutation Processes. In *Symposium on Advances in Approximate Bayesian Inference*, pp. 1–7, 2019.

Parthasarathy, N., Batty, E., Falcon, W., Rutten, T., Rajpal, M., Chichilnisky, E., and Paninski, L. Neural Networks for Efficient Bayesian Decoding of Natural Images from Retinal Neurons. In *Advances in Neural Information Processing Systems 30*, pp. 6434–6445. 2017.

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.

Shan, K. Q., Lubenov, E. V., and Siapas, A. G. Model-based spike sorting with a mixture of drifting t-distributions. *Journal of neuroscience methods*, 288:82–98, 2017.

Sohn, K., Lee, H., and Yan, X. Learning structured output representation using deep conditional generative models. In *Advances in neural information processing systems*, pp. 3483–3491, 2015.

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.

Sutskever, I., Vinyals, O., and Le, Q. V. Sequence to sequence learning with neural networks. In *NIPS*, 2014.

Tucker, G., Lawson, D., Gu, S., and Maddison, C. J. Doubly reparameterized gradient estimators for monte carlo objectives. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=HkG3e205K7`.

Vinh, N. X., Epps, J., and Bailey, J. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *Journal of Machine Learning Research*, 11(Oct):2837–2854, 2010.

Wagstaff, E., Fuchs, F. B., Engelcke, M., Posner, I., and Osborne, M. On the limitations of representing functions on sets. *arXiv preprint arXiv:1901.09006*, 2019.

Wang, Y., Pakman, A., Mitelut, C., Lee, J., and Paninski, L. Spike sorting using the neural clustering process. *Real Neurons & Hidden Units Workshop @ NeurIPS 2019*, 2019.

Wood, F. and Black, M. J. A nonparametric bayesian alternative to spike sorting. *Journal of neuroscience methods*, 173(1):1–12, 2008.

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