HHS Public Access

Author manuscript

J Comput Graph Stat. Author manuscript; available in PMC 2021 January 01.

Published in final edited form as:

J Comput Graph Stat. 2020 ; 29(1): 53–65. doi:10.1080/10618600.2019.1624366.

Scalable Bayesian Nonparametric Clustering and Classification

Yang Ni[1,2], **Peter Müller**[3], **Maurice Diesendruck**[2], **Sinead Williamson**[4], **Yitan Zhu**[5], **Yuan Ji**[6]

[1]Department of Statistics, Texas A&M University

[2]Department of Statistics and Data Sciences, The University of Texas at Austin

[3]Department of Mathematics, The University of Texas at Austin

[4]Department of Information, Risk, and Operations Management, The University of Texas at Austin

[5]Program for Computational Genomics and Medicine, NorthShore University HealthSystem*

[6]Department of Public Health Sciences, The University of Chicago

Abstract

We develop a scalable multi-step Monte Carlo algorithm for inference under a large class of nonparametric Bayesian models for clustering and classification. Each step is "embarrassingly parallel" and can be implemented using the same Markov chain Monte Carlo sampler. The simplicity and generality of our approach makes inference for a wide range of Bayesian nonparametric mixture models applicable to large datasets. Specifically, we apply the approach to inference under a product partition model with regression on covariates. We show results for inference with two motivating data sets: a large set of electronic health records (EHR) and a bank telemarketing dataset. We find interesting clusters and competitive classification performance relative to other widely used competing classifiers. Supplementary materials for this article are available online.

Keywords

Electronic health records; non-conjugate models; parallel computing; product partition models

1 Introduction

We propose a distributed Monte Carlo algorithm for Bayesian nonparametric clustering and classification methods that is suitable for large sample sizes. The algorithm is applicable for both conjugate and non-conjugate structures, and consists of K computationally efficient steps, where K is dynamically determined and is typically less than four. In each of the first $(K-1)$ steps, we divide the data into multiple shards and run Markov chain Monte Carlo (MCMC) simulations in each shard. The algorithm is run in parallel without any communication between the parallel jobs. Schemes with such zero communication cost are

*Present address: Data Science and Learning Division, Argonne National Laboratory, Argonne, IL 60439

Supplementary Materials

Additional figures for simulation studies. (.pdf file)

R code demonstrates simulations using the proposed SIGN algorithm. Python code preprocesses the EHR data using GAN algorithm.

known as “embarrassingly parallel” algorithms, a term that first appeared in Moler (1986). In the last step, MCMC is run again to generate approximate samples from the full posterior. We apply the algorithm for inference in a product partition model with regression on covariates (PPMx, Müller et al. 2011), and show results for a large electronic health records (EHR) dataset and a telemarketing dataset. The method is scalable, shows competitive performance compared to state-of-the-art classifiers and generates interpretable partitions of the data.

Classification and clustering.

We consider Bayesian nonparametric (BNP) methods for clustering and classification. Classification aims to assign observations into two or more categories on the basis of training data with known categories. Widely used classification algorithms include logistic regression (LR), naive Bayes, neural networks, k-nearest neighbors, support vector machines (SVM, Cortes and Vapnik 1995), decision trees, random forests (RF, Ho 1995), classification and regression trees (Breiman et al. 1984), Bayesian additive regression trees (BART, Chipman et al. 2010), and mixture models based on BNP priors. Some recent examples for the latter are Cruz-Mesía et al. (2007) who use a dependent Dirichlet process prior, Mansinghka et al. (2007) who model the distribution within each subpopulation defined by the class labels using a Dirichlet process mixture model, or Gutiérrez et al. (2014) who use a geometric-weights prior instead. For more examples, see a recent review by Singh et al. (2016) and references therein.

In contrast to supervised learning in classification, clustering methods partition the observations into latent groups/clusters in an unsupervised manner, with the aim of creating homogeneous groups such that observations in the same cluster are more similar to each other than to those in other clusters. Widely used clustering methods include hierarchical clustering, k-means, DBSCAN (Ester et al., 1996) and finite mixture models. Posterior simulation for finite mixtures was first discussed in Richardson and Green (1997) and extended to multivariate mixtures in Dellaportas and Papageorgiou (2006). See, for example, Jain (2010) and Fahad et al. (2014) for recent reviews. BNP (Hjort et al., 2010) clustering methods offer particularly flexible alternatives to the earlier mentioned clustering algorithms. Examples include Dirichlet process mixtures (DPM, Lo, 1984; MacEachern, 2000; Lau and Green, 2007) and variations with different data structures, such as Rodriguez et al. (2011) for a mixture of graphical models, Pitman-Yor process (PY) mixtures (Pitman and Yor, 1997; Ni et al., 2018), normalized inverse Gaussian process mixtures (Lijoi et al., 2005), normalized generalized Gamma process mixtures (Lijoiet al., 2007), and more general classes of BNP mixture models (Barrioset al., 2013; Favaro and Teh, 2013; Argiento et al., 2010).

Scalable methods.

Datasets that are too large to be analyzed on a single machine increasingly arise in many applications. Many of the earlier mentioned classification or clustering methods do not easily scale to large datasets, partly due to lack of straightforward parallelization. Below, we briefly review some recently proposed computation efficient strategies.

Zhang et al. (2012) developed two algorithms for parallel statistical optimization based on averaging and bootstrapping. Kleiner et al. (2014) developed a scalable bootstrap to evaluate the uncertainty of estimators. Bayesian methods naturally provide uncertainty quantification of estimators but are in general computation-intensive. Huang and Gelman (2005) proposed consensus Monte Carlo algorithms that distribute data to multiple machines running separate MCMC simulations in parallel. Various ways of eventually consolidating simulations from these subset posteriors have been proposed (Neiswanger et al., 2013; Wang and Dunson, 2013; Whiteet al., 2015; Minsker et al., 2014; Scott et al., 2016). An alternative strategy for scalable Bayesian computation is based on approximating the full likelihood/posterior using subsampling techniques (Welling and Teh, 2011; Korattikara et al., 2014; Bardenet et al., 2014; Quiroz et al., 2018); see Bardenet et al. (2015) for a review of related recent MCMC approaches. Alternatively to MCMC, Bayesian inference can be carried out by using approximation such as variational Bayes (Jaakkola and Jordan, 2000; Ghahramani and Beal, 2001; Broderick et al., 2013; Hoffman et al., 2013). For a grand overview of Bayesian computation, see also Green et al. (2015). Although variational inference is scalable to large-scale datasets and usually yields good approximations to the marginal posterior, MCMC algorithms tend to better approximate the joint posterior, due to their nature as simulation-exact methods.

Scalable classification and clustering.

Some classical classifiers like logistic regression are scalable to large datasets and easy to interpret. However, logistic regression tends to be not as accurate as other “black box” classifiers. Ideally, a good classifier should not need to sacrifice its predictive performance for interpretability and scalability. This is what we aim to achieve in this paper.

Some work has been done in this area. Payne and Mallick (2018) developed a two-stage Metropolis-Hastings algorithm for logistic regression to avoid the need for exact likelihood computation. The first stage, based on an approximate likelihood, is used to determine whether a full likelihood evaluation is necessary in the second stage. Combined with consensus Monte Carlo, the proposed method scales well to datasets with large samples. Rebentrost et al. (2014) implemented SVM on a quantum computer and showed exponential speed-up compared to classical sampling algorithms.

For clustering, Pennell and Dunson (2007) developed a two-stage approach for fitting random effects models to longitudinal data with large sample size. They first cluster subjects using a deterministic algorithm and then cluster the group-specific random effects using a DPM model. Zhao et al. (2009) proposed a parallel k-means clustering algorithm using the MapReduce framework (Dean and Ghemawat 2008). Wang and Dunson (2011) developed a single-pass sequential algorithm for conjugate DPM models. In each iteration, they deterministically assign the next subject to the cluster with the highest probability conditional on past cluster assignments and data up to the current observation. The algorithm is repeated for multiple permutations of the samples. Lin (2013) proposed a one-pass sequential algorithm for DPM models. The algorithm utilizes a constructive characterization of the posterior distribution of the mixing distribution given data and a partition. Variational inference is adopted to sequentially approximate the marginalization. Williamson et al.

(2013) introduced a parallel MCMC for DPM models which involves iteration over local updates and a global update. For the local update, they exploit the fact that a Dirichlet mixtures of Dirichlet processes (DP) again defines a DP, if the parameters of Dirichlet mixture are suitably chosen. Geet al. (2015) used a similar characterization of the DP as in Lin (2013). But instead of a variational approximation, they adapted the slice sampler for parallel computing under a MapReduce framework. Tank et al. (2015) developed two variational inference algorithms for general BNP mixture models.

The method that is most similar to the approach in this paper is the subset nonparametric Bayesian (SNOB) clustering of Zuanetti et al. (2018), a computation-efficient alternative for model-based clustering under a DPM model with conjugate priors. SNOB is a two-step approach. It first splits data into shards and computes clusters locally in parallel. A second step combines the local clusters into global clusters. All steps are carried out using MCMC simulation under a common DPM model. All cluster-specific parameters are marginalized out in the second step in order to merge local clusters. The latter requires conjugate models and limits its applicability in a wide range of applications where non-conjugate models are desired. In addition, for large datasets, the number of local clusters may still be too large to process in a single machine. This motivates the construction of a more general multi-step algorithm that allows for possibly non-conjugate models.

Proposed method.

Inspired by the Algorithm 8 in Neal (2000) for inference in DPM models, we extend SNOB to clustering under non-conjugate BNP models, and propose a multi-step algorithm for subset inference of general nonparametric Bayesian methods (SIGN). The algorithm is a K-step approach (K is dynamically determined and will be introduced in Section 2). Each step requires clustering on small subsets only. The number of required subsets grows linearly with the sample size n, making it possible to implement posterior inference also for data that is too large for full MCMC simulation. SIGN can be applied with a large class of BNP mixture models. Particularly, we show how SIGN is implemented for inference under the PPMx model to simultaneously cluster and classify patients from a large Chinese EHR dataset with 85,021 samples and customers from a bank telemarketing dataset with 37,078 records. SIGN relies on a notion of “clustering of clusters” which, in different contexts, has been successfully used in the literature before. For example, Argiento et al. (2014) and Malsiner-Walli et al. (2017) developed clustering-of-clusters models for clustering non-Gaussian or non-convex data. In the upcoming discussion, we use the same notion to greatly reduce the computation cost of clustering large datasets.

In the context of a classification problem, SIGN still requires that all data can be accessed. This is not an inherent constraint of the proposed algorithm; rather it is due to the lack of sufficient/summary statistics for general classification models (such as probit regression). Whenever such statistics exist, SIGN does not need to access the entire dataset.

The remainder of this paper is organized as follows. In Section 2, we introduce the proposed SIGN algorithm which is applied for inference under the PPMx model in Section 3. The SIGN algorithm is evaluated with simulation studies in Section 4 and applied to EHR and bank telemarketing data in Section 5. We conclude with a discussion in Section 6.

2 The proposed SIGN algorithm

2.1 BNP clustering

We propose an algorithm for posterior inference on random partitions under BNP mixture models. To state the general model, we need some notation. A partition $\rho = \{S_1,\ldots,S_C\}$ of an index set $[n] = \{1,\ldots,n\}$ is a collection of nonempty, disjoint and exhaustive subsets $S_c \subseteq [n]$. The partition can alternatively be represented by a set of cluster membership indicators $\boldsymbol{s} = (s_1,\ldots,s_n)$ with $s_i = c$ if $i \in S_c$. Throughout the paper, we will use superscript $-i$ to represent the appropriate quantity with the ith sample or the ith item (defined later) removed. For instance, $\boldsymbol{s}^{-i} = \boldsymbol{s}\backslash s_i$ and $\rho^{-i} = \left(\rho\backslash S_{s_i}\right) \cup \left(S_{s_i}\backslash i\right)$ are the cluster memberships and partition after removing index i.

In what follows we consider a *random* partition ρ with prior probability distribution $p(\rho)$. Let $n_c = |S_c|$ denote the cardinalities of the partitioning subsets. Let $\boldsymbol{n} = (n_1, \ldots, n_C)$ and let $\boldsymbol{n}^{j+}$ denote with the jth element incremented by 1, with the convention that $\boldsymbol{n}^{(C+1)+} = (n_1, \ldots, n_C, 1)$. A random partition is called exchangeable if $p(\rho) = f(\boldsymbol{n})$ for a symmetric (in its arguments) function $f(\mathbf{n})$ and if $f(\mathbf{n}) = \sum_{j=1}^{C+1} f\left(\mathbf{n}^{j+}\right)$. The function $f(\boldsymbol{n})$ is known as the exchangeable partition probability function (EPPF). By Kingman's representation theorem (Kingman, 1978), any exchangeable random partition can be characterized as the groups formed by ties under i.i.d. sampling from a discrete probability measure $G = \sum_{h=1}^{\infty} w_h \delta_{m_h}$. That is, ρ is determined by the ties among $\theta_i \sim G$, $i = 1,\ldots,n$. We denote the unique values of θ_i's by $\theta_1^{\star}, \ldots, \theta_C^{*}$, implying $i \in S_c$ if $\theta_i = \theta_c^{\star}$. See, for example, Lee et al. (2013) for a discussion. It follows that a prior probability model for an exchangeable random partition ρ can always be defined as a prior $p(G)$ on a random discrete distribution $G = \sum_{h=1}^{\infty} w_h \delta_{m_h}$. This implicit definition of $p(\rho)$ by a BNP prior $p(G)$ on the random probability measure G is a commonly used specification of random partition models. The construction already includes cluster-specific parameters $\theta_c^{\star}$ which are useful for the construction of a sampling model conditional on the partition. We use it in the next step of the model construction.

The model on G and θ_i is completed with a sampling model for the observed data conditional on ρ. For example, the θ_i could index a sampling model $p(y_i \mid \theta_i)$, implying that all observations in a cluster share the same sampling model. In summary,

$$y_i|\theta_i \overset{ind}{\sim} p(y_i|\theta_i),\ \theta_i|G \overset{ind}{\sim} G, G \sim H,$$

where G is a discrete prior distribution for θ_i and H is the BNP prior for the random probability measure G.

There are a number of options for H. A popular choice is the DP, which yields an EPPF of the form $p(\rho) \propto \alpha^{C-1} \prod_{c=1}^{C} (n_c - 1)!$ where α is known as the concentration parameter. Other choices include the PY process, the normalized inverse Gaussian process and the normalized generalized gamma process. In many applications, the focus is on the posterior distribution of the random partition ρ. The posterior distribution on ρ can be approximated

by various MCMC algorithms, including Escobar (1994), MacEachern and Müller (1998), Neal (2000) and Walker (2007) in the case of the DP prior, and, for more general models, Barrios et al. (2013), Favaro and Teh (2013) or Argiento et al. (2010). However, MCMC is only practicable for small to moderate datasets. Directly applying MCMC to large datasets is very costly because the algorithm has to scan through all observations at each iteration, each requiring likelihood and prior evaluations.

2.2 SIGN algorithm

The proposed SIGN algorithm proceeds in steps. For illustration, an example workflow of SIGN with $K = 3$ steps is shown in Figure 1. Importantly, across all steps of the algorithm, all updates of cluster configurations (initially of observations, and of sets of observations in later steps) are based on a single underlying BNP mixture model for the data. Details of the implied probabilities for clustering sets of obserations are given later.

Step 1. In the first step, the full dataset is randomly split into M_1 shards, $M_1 = 4$ in Figure 1; the observations from each shard are denoted by a distinct symbol in the figure. A clustering method is then applied (in our implementation, using the algorithm 8 in Neal (2000) for posterior simulation) to cluster the items (initially, in the first step, the observations) in each shard separately, and can be implemented in parallel. We refer to the estimated clusters, represented by the ellipses, as “local” clusters. These local clusters are frozen, meaning that the observations within each cluster will never be divided in the subsequent steps, although merging is possible.

Step 2. The local clusters estimated from the previous step become the items to be clustered in this step. The items are distributed into M_2 shards ($M_2 = 2$ in Figure 1). We still use the same underlying BNP mixture model to cluster the items. See later for a statement of the relevant probabilities to cluster the items. At the end of the second step, the estimated clusters are again frozen as “regional” clusters.

Step 3. At the last step, all regional clusters are collected to form the items for the next, third, step. Again items are split into M_3 shards and clustered within each shard. In the example of Figure 1, $M_3 = 1$ and iteration stops.

In general, iteration continues until the number of items is sufficiently small to be clustered in a single shard. Importantly, at each step one need to only scan through a small number of items (created by previous steps), instead of all observations in a large dataset. Each step can be implemented in parallel using instances of the same MCMC algorithm which takes as input a set of (current) items, generically denoted by $\tilde{\mathbf{y}} = \{\tilde{\mathbf{y}}_1, \ldots, \tilde{\mathbf{y}}_B\}$ and outputs estimated clusters of these items. Those clusters then define the items for the next step of the algorithm. Initially, in step 1, $\tilde{\mathbf{y}}_i = \mathbf{y}_i$ are the original data. Let $r_i = |\tilde{\mathbf{y}}_i|$, $i = 1, \ldots, B$ denote the size of each item, in terms of the number of original data that form $\tilde{\mathbf{y}}_i$, and let $\boldsymbol{r} = \{r_1, \ldots, r_B\}$. In the following, we describe the posterior probabilities for clustering items, i.e., sets of original observations, in any of the steps $k = 1, \ldots, K$. To simplify notation we do not include an index for steps.

2.2.1 MCMC—In each of the K steps, MCMC simulation iterates between (i) updating the cluster membership, and (ii) updating cluster-specific parameters given the cluster membership. The key quantity in updating the cluster membership is the conditional probability

$$p\left(\tilde{s}_i = c \mid \tilde{\boldsymbol{s}}^{-\boldsymbol{i}}, \tilde{\boldsymbol{y}}, \boldsymbol{\theta}^{\star}\right) \propto p\left(\tilde{s}_i = c \mid \tilde{\boldsymbol{s}}^{-\boldsymbol{i}}\right) p(\tilde{\boldsymbol{y}}_i \mid \theta_c^*) \tag{1}$$

for $i = 1,\ldots,B$ and $c = 1,\ldots,C^{-i} + 1$ where $\tilde{s}_i = c$ means that item i is in cluster c, i.e., all observations in $\tilde{\boldsymbol{y}}_i$ are assigned to cluster c. The definition of the items $\tilde{\boldsymbol{y}}_i$ and the number of items, B, changes across steps. Initially, $\tilde{\boldsymbol{y}}_i$ are the original data, and $B = n$ is the sample size. In step 2, the items $\tilde{\boldsymbol{y}}_i$ are the local clusters and B is the total number of local clusters, etc. Importantly, the probabilities that are evaluated under (1) and used for clustering in steps 1 through 3 are all based on the same BNP mixture model for the original observations.

Equation (1) states the probabilities for combining clusters of observations into larger clusters. The first factor can be evaluated as

$$p\left(\tilde{s}_i = c \,\middle|\, \tilde{\boldsymbol{s}}^{-\boldsymbol{i}}\right) \propto \frac{p\left(\rho^{+c}\right)}{p\left(\rho^{-i}\right)} \tag{2}$$

where $\rho^{+c} = \left(\rho^{-i} \backslash S_c^{-i}\right) \cup \left(S_c^{-i} \cup \tilde{i}\right)$ is the new partition that assigns the ith item to cluster c (that is, all original data points that make up $\tilde{\boldsymbol{y}}_i$). The partition probabilities on the right-hand side of (2) depend on $\boldsymbol{r}\,\boldsymbol{n}$ and the BNP prior H. For example, using $H = \mathrm{PY}(d, G_0)$ with concentration parameter α, discount parameter d, and baseline probability measure G_0, implies the random prior partition:

$$p(\rho) \propto (\alpha \mid d)_C \prod_{c=1}^{C} (1-d)_{n_c - 1}, \tag{3}$$

where $(x)n = x(x+1)\ldots(x+n-1)$ denotes the Pochhammer symbol of a rising factorial, and $(x \mid y)_n = x(x+y)\ldots(x+(n-1)y)$ denotes the Pochhammer symbol with increment y. Substituting (3) into (2) yields

$$p\left(\tilde{s}_i = c \mid \tilde{\boldsymbol{s}}^{-\boldsymbol{i}}\right) \propto \begin{cases} \dfrac{\Gamma\left(n_c^{-i} + r_i - d\right)}{\Gamma\left(n_c^{-i} - d\right)} & \text{if } c = 1, \ldots, C^{-i} \\ \dfrac{\left(\alpha + dC^{-i}\right)\Gamma(r_i - d)}{\Gamma(1-d)} & \text{if } c = C^{-i} + 1 \end{cases}, \tag{4}$$

where n_c^{-i} is the size of the cth cluster after removing the ith item $\tilde{\boldsymbol{y}}_i$ (recall that size is recorded in original data units). In the special case when $r_i = |\tilde{\boldsymbol{y}}_i| = 1$ for all i, equation (4) reduces to the Pólya urn representation of the PY.

The second factor in (1) is the sampling model evaluated for $\tilde{\boldsymbol{y}}_i$ given the cluster-specific parameters, which is straightforward to compute except for the missing $\theta_c^{\star}$ for $c = C^{-i} + 1$,

which could be avoided by marginalizing with respect to $\theta_c^\star$. In general, this marginalization is analytically intractable and we use instead the Algorithm 8 in Neal (2000). The algorithm includes a temporary model augmentation with m latent variables $\theta_c^\star, c = C^{-i} + 1, \ldots, C^{-i} + m$ which serve as potential new cluster parameters, and are generated from the prior for unique atoms of the BNP prior (base measure in the DP or PY prior). The prior probability for a new cluster in (4) is split equally among the m potential values. The case when resampling $\tilde{s}_i$ removes a current cluster, say S_c, by re-assigning the only element of a singleton cluster, needs careful attention. In that case, $\theta_c^\star$ becomes $\theta_{C^{-i}+1}^\star$ and only the remaining latent variables are sampled from the prior. The only remaining parameters to be sampled in the MCMC are the cluster-specific parameters $\theta_j^\star$, which are updated using a suitable MCMC transition probability. At the end of each MCMC pass, we compute a least-squares estimate of the partition (Dahl, 2006). An alternative method of summarizing partitions is also considered in Section 4.1. Algorithm 1 summarizes the MCMC simulation.

Algorithm 1 MCMC

1: **function** MCMC($\tilde{\mathbf{y}}$) // $\tilde{\mathbf{y}} := \{\tilde{\mathbf{y}}_1, \ldots, \tilde{\mathbf{y}}_B\}$
2: Initialize the partition
3: **for** iter $= 1, \ldots, N$ **do** //N: number of iterations
4: Update cluster-specific parameters $\theta_j^\star$
5: Update cluster membership, $\tilde{s}_i$, using (1) and using Neal's Algorithm 8 as described in the text
6: **end for**
7: Compute the estimated partition $\hat{\rho} = \{S_1, \ldots, S_C\}$
8: **Output:** $\tilde{\mathbf{y}}^* = \{\tilde{\mathbf{y}}_1^*, \ldots, \tilde{\mathbf{y}}_C^*\}$ // $\tilde{\mathbf{y}}_c^* := \bigcup_{i \in S_c} \tilde{\mathbf{y}}_i$
9: **end function**

2.2.2 The complete scheme—The complete SIGN algorithm simply repeatedly distributes the items (i.e., blocked observations) into shards and applies Algorithm 1 to each shard in parallel. The number K of steps is dynamically determined by specifying a maximum number R (typically a few hundred) of items that can be clustered in one processor. Simulation terminates when the total number of items is less than R. For example, suppose we set $R = 200$ and we obtain 600 local clusters from the first step. Since $600 > R$, we decide to distribute the local clusters into 3 shards each with 200 blocked observations. If 30 regional clusters are returned from the second step, we will in a final step cluster the regional clusters in one shard. Since $30 < R$, there is no need for a further split and iteration stops. Hence in this example $K = 3$. The complete scheme is summarized in Algorithm 2. SIGN implements approximate posterior inference under the BNP mixture model. The approximation arises from the fact that observations in the same item at any step will not be split in any of the subsequent steps. See the following discussion for more details.

The SIGN algorithm can be applied with a wide range of BNP mixture models. The key step is deriving the prior probabilities for combining clusters. If an EPPF for the underlying BNP prior is available, it is simply evaluated as the ratio of two EPPFs as shown in Equation (2). Combined with the sampling model, the cluster membership of each item can be drawn from its full conditional.

Algorithm 2 SIGN for BNP clustering

1: **function** SIGN(y, R) // $y := \{y_1, \ldots, y_n\}$
2: Initialize $\tilde{y} = y$
3: **while** $B > R$ **do** // B: number of blocks in $\tilde{y}$
4: Set $M = \left\lceil \frac{B}{R} \right\rceil$ // M: number of shards
5: Randomly distribute $\tilde{y}$ into M shards: $\tilde{y}_1, \ldots, \tilde{y}_M \subseteq \tilde{y}$
6 **parfor** each shard $m = 1, \ldots, M$ **do**// parallel for loop
7: $\tilde{y}^*_m = \text{MCMC}(\tilde{y}_m)$ // call Algorithm 1
8 **end parfor**
9: Set $\tilde{y} = \cup_{m=1}^{M} \tilde{y}^*_m$ and redefine B // clusters become blocks for next step
10: **end while**
11: **Output:** $\tilde{y}$
12: **end function**

2.2.3 Approximation—The described update for ρ involves an approximation of the target distribution $p(\rho \mid \boldsymbol{y})$. The SIGN algorithm reduces computation time by replacing the problem of clustering n items by essentially M_1 problems of clustering n / M_1 items. The nature of the involved approximation is similar in flavor to a variational Bayes approximation. This is seen in a simplified setup assuming $M_1 = 2$ shards, say A_1 and A_2, and assuming that iteration stops after $K = 2$ steps. Let d_{ij} denote co-clustering indicators $d_{ij} = I(s_i = s_j)$, and let $\boldsymbol{d}_1 = (d_{ij};\ i, j \in A_1)$, and similarly for $\mathbf{d}_2$ and $\boldsymbol{d}_{12} = (d_{ij};\ i \in A_1, j \in A_2)$. Then $(\boldsymbol{d}_1, \boldsymbol{d}_2, \boldsymbol{d}_3)$ is an alternative representation of a partition ρ. Let $\boldsymbol{y}_1$ denote the data in A_1 and similarly for $\boldsymbol{y}_2$. If SIGN is implemented by generating one (approximate) posterior draw of $\boldsymbol{d}_1$ and $\boldsymbol{d}_2$ in each shard, then the algorithm $p(\boldsymbol{d}_1, \boldsymbol{d}_2, \boldsymbol{d}_{12} \mid \boldsymbol{y})\ q(\boldsymbol{d}_1, \boldsymbol{d}_2, \boldsymbol{d}_{12}) = p(\boldsymbol{d}_1 \mid \boldsymbol{y}_1)\ p(\boldsymbol{d}_2 \mid \boldsymbol{y}_2)\ p(\boldsymbol{d}_{12} \mid \boldsymbol{d}_1, \boldsymbol{d}_2, \boldsymbol{y})$, that is, a distribution that is independent in $\boldsymbol{d}_1, \boldsymbol{d}_2$, but with each factor defined by the original target distribution. In contrast to a variational Bayesian approach there is no notion of optimizing the approximation.

Based on these considerations we propose a simple diagnostic to summarize the level of approximation. Select any two of the M_1 shards in step 1, say A_1 and A_2. We then carry out Steps 1a and 2a like Steps 1 and 2 before, but restricted to the two selected shards only. Alternatively we implement (simulation exact) posterior MCMC simulation in

$A_1 \cup A_2$. For any pair (i, j) in $A_1 \cup A_2$, let $p_{ij} = E(d_{ij} \mid y)$ denote the posterior co-clustering probabilities let $\tilde{p}_{ij}$ denote the estimate based on Steps 1a and 2a, and let $\hat{p}_{ij}$ denote the same based on the MCMC simulation. We summarize the level of the approximation in SIGN by reporting the histogram of $\tilde{p}_{ij} - \hat{p}_{ij}$ and the proportion $F_{0.1}$ of pairs (i, j) with $\left|\tilde{p}_{ij} - \hat{p}_{ij}\right| < 0.1$. For the simulations and examples discussed in later sections, we find the histogram is concentrated around zero and $F_{0.1} > 70\%$.

3 Clustering and classification with PPMx

Clustering is often carried out to find more homogeneous subpopulations that can then be used, for example, to derive improved classification and prediction. One example of such approaches is the PPMx model, which allows for simultaneously partitioning of heterogeneous samples and predicting outcomes on the basis of covariates. To fix notation,

let $z_i \in \{0,1\}$ denote a binary outcome (reserving notation y_i for a later introduced augmented response). Let

$\boldsymbol{x}_i = \{\boldsymbol{w}_i, \boldsymbol{u}_i\}$ denote a set of continuous covariates $\boldsymbol{w}_i = (w_{i1}, \ldots, w_{ip})$ and categorical covariates $\boldsymbol{u}_i = (u_{i1}, \ldots, u_{iq})$ for experimental units $i = 1, \ldots, n$. Let $\boldsymbol{z} = \{z_1, \ldots, z_n\}$ and $\boldsymbol{x} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$. A product partition model (PPM) (Hartigan, 1990) assumes $p(\rho) \propto \prod_{c=1}^{C} h(S_c)$, where $h(\cdot)$ is a non-negative cohesion function that quantifies the tightness of a cluster. For example, the prior distribution on partitions that is induced under i.i.d. sampling from a DP-distributed random measure with concentration parameter α is a PPM with $h(S_c) = \alpha \times (|S_c| - 1)!$. Müller et al. (2011) define the PPMx as a variation of the PPM by introducing prior dependence on covariates by augmenting the random partition to

$$p(\rho \mid \boldsymbol{x}) \propto \prod_{c=1}^{C} h(S_c) g\left(\boldsymbol{x}_c^{\star}\right), \tag{5}$$

with a nonnegative similarity function $g(\cdot)$ indexed by covariates where $\boldsymbol{x}_c^{\star} = \{\boldsymbol{x}_i \mid i \in S_c\}$ are the covariates of observations in the cth cluster. The similarity function measures how similar the covariates are thought to be. A computationally convenient default way to define a similarity function uses the marginal probability in an auxiliary probability model q on x:

$$g\left(\boldsymbol{x}_c^{\star}\right) = \int \prod_{i \in S_c} q_x(\boldsymbol{x}_i \mid \xi_c) q_\xi(\xi_c) d\xi_c .$$

The important feature here is that the marginal distribution has higher density value for a set of very similar x_i than for a very diverse set. For continuous covariates, we use an independent normal-normal-gamma auxiliary model. Let $N(x \mid m, v)$ denote the evaluation of a normal density with moments (m, v), evaluated at x, and similar for a gamma density, $\mathrm{Ga}(x \mid a, b)$ and other distributions. We use $q_x\left(w_{ij} \mid \mu_c, \lambda_c\right) = \mathrm{N}\left(w_{ij} \mid \mu_c, \lambda_c^{-1}\right)$ and $q_\xi(\mu_c, \lambda_c) = \mathrm{N}\left(\mu_c \mid \mu_0, (\upsilon_0 \lambda_c)^{-1}\right) \times Ga(\lambda_c \mid a_\lambda, b_\lambda)$. In this case, $\boldsymbol{\xi}_c = (\mu_c, \lambda_c)$. Let $\mathrm{Cat}(x \mid \boldsymbol{\pi})$ indicate a discrete r.v. x with probabilities $p(x = l) = \boldsymbol{\pi}_l$. For categorical covariates with r categories, we use a categorical-Dirichlet auxiliary model, $q_x(u_{ij} \mid \boldsymbol{\pi}_c) = \mathrm{Cat}(u_{ij} \mid \boldsymbol{\pi}_c)$ and $q_\xi(\boldsymbol{\pi}_c) = \mathrm{Dir}(\boldsymbol{\pi}_c \mid a_\pi, \ldots, a_\pi)$ with $\boldsymbol{\pi}_c = (\boldsymbol{\pi}_{c1}, \ldots, \boldsymbol{\pi}_{cr})$. The prior $p(\rho \mid \boldsymbol{x})$ introduces the desired covariate-dependent prior on the clusters $\mathcal{S}_c$.

Conditional on ρ we introduce cluster-specific parameters $\boldsymbol{\beta}_c$ and complete the model with a probit sampling model,

$$p(\boldsymbol{z} \mid \rho, \boldsymbol{\beta}, \boldsymbol{x}) = \prod_{c=1}^{C} \prod_{i \in S_c} p(z_i \mid \boldsymbol{x}_i, \boldsymbol{\beta}_c) = \prod_{c=1}^{C} \prod_{i \in S_c} p_i^{z_i} (1 - p_i)^{1 - z_i} \tag{6}$$

with $p_i = \Phi(\boldsymbol{x}_i \boldsymbol{\beta}_c)$ and a centered multivariate normal prior on $\boldsymbol{\beta}_c \sim \mathrm{N}(0, \tau_\beta I)$.

A practical advantage of the PPMx is its simple implementation. The posterior defined by models (5) and (6) becomes

$$p(\rho, \boldsymbol{\beta}, \boldsymbol{\xi} | \boldsymbol{z}, \boldsymbol{x}) \propto \prod_{c=1}^{C} \left[\left\{ \prod_{i \in S_c} p(z_i | \boldsymbol{x}_i, \boldsymbol{\beta}_c) q_x(\boldsymbol{x}_i | \boldsymbol{\xi}_c) \right\} p(\boldsymbol{\beta}_c) q_{\xi}(\boldsymbol{\xi}_c) h(S_c) \right].$$

Letting $\boldsymbol{y}_i = \{z_i, \boldsymbol{x}_i\}, \theta_c^{\star} = \{\boldsymbol{\beta}_c, \boldsymbol{\xi}_c\}, q_y\left(y_i | \theta_c^{\star}\right) = p(z_i | \boldsymbol{x}_i, \boldsymbol{\beta}_c) q_x(\boldsymbol{x}_i | \boldsymbol{\xi}_c)$, $q_{\theta}\left(\theta_c^{\star}\right) = p(\boldsymbol{\beta}_c) q_{\xi}(\boldsymbol{\xi}_c)$ and $q_{\rho}(\rho) = \prod_{c=1}^{C} h(S_c)$ one can rewrite the posterior distribution as

$$p(\rho, \boldsymbol{\beta}, \boldsymbol{\xi} | \boldsymbol{z}, \boldsymbol{x}) \propto \prod_{c=1}^{c} \prod_{i \in S_c} q_y\left(\boldsymbol{y}_i | \theta_c^{\star}\right) \times \prod_{c=1}^{C} q_{\theta}\left(\theta_c^{\star}\right) \times q_{\rho}(\rho). \quad (7)$$

That is, posterior inference can proceed as if $\boldsymbol{y}_i$ were sampled from Equation (7). For example, in our application, we choose $q_{\rho}(\cdot)$ to be the random partition that is induced by a PY prior. The PY generalizes the DP and is more flexible in modeling the number of clusters (De Blasi et al., 2015). Posterior inference under (7) can then be carried out using Equation (2.1) (and hence Algorithms 1 and 2) with $p(y_i | \cdot) = q_y(y_i | \cdot)$, $H = \mathrm{PY}(\alpha, d, G_0)$ and $G_0 = q_{\theta}(\cdot)$. Note how (7) is identical to the posterior in a model with data $\boldsymbol{y}_i$ cluster-specific parameters $\boldsymbol{\theta}^*_c$ and prior $q_{\rho}(\rho)$, allowing for easy posterior simulation.

One of the goals in our later applications is to classify a new subject, i.e., predict the binary outcome $\boldsymbol{z}_{n+1}$, on the basis of covariates $\boldsymbol{x}_{n+1}$. It is straightforward to predict z_{n+1} using posterior averaging with respect to partitions, cluster allocation and model parameters. Let $q\left(\boldsymbol{x}_{n+1} | \boldsymbol{x}_c^{\star}\right) = g\left(\boldsymbol{x}_c^{\star}, \boldsymbol{x}_{n+1}\right) / g\left(\boldsymbol{x}_c^{\star}\right)$. The posterior predictive distribution is given by

$$p(z_{n+1} | \boldsymbol{x}_{n+1}, \boldsymbol{z}, \boldsymbol{x}) \propto \int \{(n_c - d) \sum_{c=1}^{C} p(z_{n+1} | \boldsymbol{x}_{n+1}, \boldsymbol{\beta}_c, s_{n+1} = c) q\left(\boldsymbol{x}_{n+1} | \boldsymbol{x}_c^{\star}\right) + (\alpha + dC) p(z_{n+1} | \boldsymbol{x}_{n+1}, \boldsymbol{\beta}_{C+1}) g(\boldsymbol{x}_{n+1}) \Big\} p(\rho | \boldsymbol{z}, \boldsymbol{x}) d\rho,$$

which can be approximated by

$$p(z_{n+1} | \boldsymbol{x}_{n+1}, \boldsymbol{z}, \boldsymbol{x}) \propto \frac{1}{T} \sum_{t=1}^{T} \{\left(n_c^{(t)} - d\right) \sum_{c=1}^{C^{(t)}} p\left(z_{n+1} | \boldsymbol{x}_{n+1}, \boldsymbol{\beta}_c^{(t)}, s_{n+1}^{(t)} = c\right) q\left(\boldsymbol{x}_{n+1} | \boldsymbol{x}_c^{\star}\right) + \left(\alpha + dC^{(t)}\right) p\left(z_{n+1} | \boldsymbol{x}_{n+1}, \boldsymbol{\beta}_{C+1}^{(t)}\right) g(\boldsymbol{x}_{n+1}) \Big\},$$

with superscript (t) indexing the tth MCMC samples t = 1,…, T, and $\boldsymbol{\beta}_{C+1}^{(t)}$ is drawn from its prior.

4 Simulation

We conduct four simulation studies. The first two simulations focus on clustering. We use a PY mixture (PYM) model. The last two simulations consider clustering and classification

based on the PPMx. We use relatively small datasets with $n = 800$ in the simulations, so that we can compare with standard MCMC implementations of inference under the PYM and PPMx models. Scalability is later explored, in two case studies. We report frequentist summaries based on 50 repetitions. For each repeat simulation, MCMC is run for 10,000 iterations, discarding the first 50% of MCMC samples as burn-in and thinning the chain by keeping every 5th sample.

4.1 Simulation I: Mixture of five normals

The first simulation considers a SIGN approximation of posterior inference in a PYM model for a $p = 5$ dimensional outcome $\boldsymbol{y}_i$:

$$\boldsymbol{y}_i|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i \overset{\text{ind}}{\sim} p(\boldsymbol{y}_i|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i),\ \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i|G \overset{\text{ind}}{\sim} G,\ G \sim \text{PY}(\alpha, d, G_0),$$

where $G_0(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = N_p(\boldsymbol{\mu}|0, \boldsymbol{\Sigma}/\kappa_0) \times \text{IW}(\boldsymbol{\Sigma}|b, \boldsymbol{I}_p)$. The hyperparameters are $\alpha = 1$, $d = 0.5$, $\kappa_0 = 0.01$, $b = p$.

We construct a simulation truth with $C_0 = 5$ true clusters with equal sizes. We generate s_i using $p(s_i = l) = 1/5$, and data $\boldsymbol{y}_i|s_i = c \sim N_p(\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)$, $i = 1,\ldots,n$. where $\boldsymbol{\mu}_1 = (-2,1.5,0,0,0)^T$, $\boldsymbol{\mu}_2 = (0,3,0,0,0)^T$, $\boldsymbol{\mu}_3 = (0,0,0,1,-2)^T$, $\boldsymbol{\mu}_4 = (1,2,0,0,0)^T$, $\boldsymbol{\mu}_5 = (0,0,0,-2,-2)^T$, $\boldsymbol{\Sigma}_1 = \text{diag}(0.25,0.1,1,1,1)$, $\boldsymbol{\Sigma}_2 = \text{diag}(1.25^2,0.1,1,1,1)$, $\boldsymbol{\Sigma}_3 = \text{diag}(1,1.1,0.1,0,0.25)$, $\boldsymbol{\Sigma}_4 = \text{blkdiag}\left(\begin{bmatrix} 0.1 & 0.05 \\ 0.05 & 0.1 \end{bmatrix}, I_3\right)$ and $\boldsymbol{\Sigma}_5 = \text{blkdiag}\left(\text{I}_3, \begin{bmatrix} 0.25 & 0.125 \\ 0.125 & 0.25 \end{bmatrix}\right)$.

In words, clusters 1, 2, and 4 are characterized by a shift in the distribution for the first two variables y_{i1} and y_{i2} with different correlation structures, whereas clusters 3 and 5 are characterized by a shift in the third and fourth variables y_{i4} and y_{i5}. And y_{i3} plays the role of a "noisy" response with the same distribution across all clusters. Variables that do not characterize clusters (such as y_{i3}, y_{i4}, y_{i5} in clusters 1, 2 and 4) are independently sampled from standard normal distributions. The data of one randomly selected simulation is shown in Supplementary Materials (Figure 1).

We implement SIGN for approximate posterior inference, using $K = 2$ steps. In the first step, the data are randomly split into $M_1 = 4$ shards with each shard processing 200 samples. We compare the SIGN approximation with a (simulation exact) MCMC implementation for PYM and with DBSCAN (Ester et al. 1996). DBSCAN is a clustering-algorithm designed to discover clusters of arbitrary shape. It has two tuning parameters: neighborhood size ϵ and the minimum number m of points in the ϵ-neighborhood. We use the default choice $m = 5$ as implemented in the *R* package "dbscan" (Hahsler and Piekenbrock 2018). The value of ϵ is set to 1, which gives the best performance in the spiral data (Section 4.2), a typical example showcasing the strength of DBSCAN. We will also examine a couple of other values of ϵ. In addition, we include a variation of SIGN with a different loss function instead of squared error loss (to summarize the partition at the end of each MCMC pass). Specifically, we use variation of information loss (Meilă 2007; Wade and Ghahramani 2018; Rastelli and Friel 2018), and refer to this variation as SIGN-VI.

The average estimated number of clusters, C, and the misclustering rate are reported in Table 1. SIGN and SIGN-VI slightly underestimate C (relative to the posterior mean under full MCMC simulation). In terms of the misclustering rate, full MCMC posterior inference is closely matched by SIGN and SIGN-VI. DBSCAN shows a 41% misclustering rate. A similar comparison holds for parameter estimation: SIGN and SIGN-VI closely match the average mean squared error for cluster-specific means $\boldsymbol{\mu}$ that is reported under full posterior MCMC, which in turn is lower than under DBSCAN (Table 1).

To further evaluate the approximation accuracy of SIGN relative to full posterior simulation for the PYM, we evaluate the posterior co-clustering probabilities $p_{ij} = p(s_i = s_j \mid \mathbf{y})$, $i < j$, using SIGN and full posterior MCMC in the PYM, and then take the differences between the two estimates across all pairs (i, j) of observations. The histogram of the differences across 50 simulations (Supplementary Material, Figure 2) indicates good approximation accuracy of SIGN.

Sensitivity analysis.—Next, we evaluate the sensitivity of the SIGN approximation with respect to different random splits of the data in the first step. We randomly pick one simulation and run SIGN 50 times, each time with a different initial split. The average estimated number of clusters is 4.96 with standard deviation 0.20 and misclustering rate 0.09 with standard deviation 0.03. The small standard deviations indicate stable performance of SIGN under different splits of the data.

To assess sensitivity with respect to the local sample size in each shard in the first step, we increase the sample size to $n = 4000$ and compare with full posterior MCMC under the PYM and with DBSCAN. Full posterior inference finds 21.8 clusters on average. As expected, larger total sample size gives rise to more clusters, a typical property of many BNP mixture models. DBSCAN finds 126.38 clusters on average using $\epsilon = 1$. DBSCAN is quite sensitive to the choice of ϵ. For example, in a randomly selected simulation, the estimated numbers of clusters are 123, 77 and 5 for $\epsilon = 1.0,1.1,1.7$. However, we note that DBSCAN is particularly designed to work well with unusual clusters, as we shall see later, in Section 4.2.

We then run SIGN with $M_1 = 4,8,10,20$ corresponding to 1000,500,400,200 data points per shard. The average (over repeat simulations) estimated numbers of clusters are 8.54,6.46,5.74,5.44 with standard deviations 1.76,1.03,0.75,0.70 The average misclustering rates are 0.06,0.05,0.05,0.09 with standard deviations 0.02,0.01,0.01,0.02. In summary, the SIGN approximation tends to under-estimate the number of clusters relatively to full MCMC. This is probably due to the fact that local clusters in earlier steps are frozen and can only grow by later merging, but can never shrink. This biases the number of estimated clusters toward smaller numbers. However, in most applications the many small and singleton clusters that are generated by some BNP mixture models are not of interest, making this approximation error less critical.

4.2 Simulation II: Two spirals

The main strength of DBSCAN is its flexibility to adjust to clusters of different shapes. Here, we consider data with two highly non-convex clusters. A typical view of the data is shown in Figure 2. Applying full MCMC for the PYM model, the SIGN approximation, and

DBSCAN for 50 randomly generated two-spiral datasets, we find the average (over the repeat simulations) estimated number of clusters to be 14.10,9.28,1.98 with standard deviations 1.37,0.93,0.14 for full posterior inference, the SIGN approximation, and for DBSCAN, respectively. However, the tuning parameter ϵ of DBSCAN is tuned in an "oracle" way, that is, we set

$\epsilon = 1$ so that the average number of clusters is close to the truth $C_0 = 2$. Not surprisingly, due to the convexity of normal density/contours, the PYM model requires more clusters to compensate the non-convex shape of the true clusters, using both, full posterior MCMC as well as using the SIGN approximation. The estimated clusters for one simulation are shown in Figure 2.

4.3 Simulation III: cluster-specific probit regression

Next we assess the performance of the SIGN implementation for inference in the PPMx. We first consider a scenario where the simulation truth includes underlying clusters. We assume a simulation truth with $C_0 = 5$ clusters, $p = q = 5$ continuous and discrete covariates, and all clusters having the same size. Discrete covariates $\boldsymbol{u}_i$ are generated as $u_{ij} \sim Cat(1/3, 1/3, 1/3)$, independently, $j = 1, \ldots, q$. Continuous covariates $\boldsymbol{w}_i$ are generated in the same way as $\boldsymbol{y}_i$ in Simulation I. The binary response z_i is generated from a cluster-specific probit regression, $z_i \sim Bernoulli(p_i)$ with

$$\Phi^{-1}(p_i) = \begin{cases} -1 - w_{i5} & \text{if } s_i = 1 \\ -1 + 2w_{i3} & \text{if } s_i = 2 \\ -1 + w_{i4} & \text{if } s_i = 3 \\ -1 + 1.5w_{i1} - I(u_{i1} = 2) + I(u_{i1} = 3) & \text{if } s_i = 4 \\ -1 - 1 - 1.5w_{i1} - I(u_{i2} = 2) + I(u_{i3} = 3) & \text{if } s_i = 5. \end{cases}$$

We fix the hyperparameters as

$a = 1, d = 0.5, \tau_\beta = 1, \mu_0 = 0, v_0 = a_\lambda = b_\lambda = 0.01, a_\pi = 1/r$, and carry out inference under the PPMx model using the default similarity functions (simply PPMx hereafter). For comparison, we also carry out inference using k-means (Hartigan and Wong, 1979) for the continuous covariates (which define the clusters) with $k = 5$ (the true number of clusters) and 20 random starting points. PPMx is always able to correctly identify the number of clusters with 2.5% average misclustering rate (with respect to cluster assignment). The SIGN approximation selects the correct number of clusters in 48 (out of 50) simulations, with average misclustering rate 7.5%. In contrast, with k-means we find a misclustering rate of 25.5%.

To assess the out-of-sample predictive performance, that is, prediction of z_{n+1}, we compute the area under the ROC curve (AUC) based on 50 independent test samples generated from the same simulation truth as the training data. In addition to the previous comparison, we also benchmark against four more alternative classifiers: sparse LR with lasso (R package "glmnet", Friedman et al., 2010), SVM ("e1071", Meyer et al., 2018), RF ("randomForest", Liaw and Wiener, 2002), and BART ("BayesTree", Chipman and McCulloch, 2016). For SVM, we transform the discrete covariates using dummy variables,

fit with linear, cubic and Gaussian radial bases and report the best performance of the three. We grow 50,000 trees for RF and 200 trees for BART. For a fair comparison, we run BART using the same MCMC configuration as ours (i.e. 10,000 iteration, 50% burn-in and save every 5th sample). The results are reported in the first column of Table 2, where we find that full posterior inference in the PPMx and the SIGN approximation have almost the same AUC's and both compare favorably with the competing classifiers.

4.4 Simulation IV: non-linear probit regression

The favorable results for SIGN and PPMx in simulation III may be partially due to the chosen simulation truth. For an alternative comparison, in this example we use a simulation truth different from the PPMx model. Particularly, we assume a simulation truth without an underlying clustering structure, generating the binary response by a nonlinear probit regression, $z_i \sim Bernoulli(p_i)$ with

$$\Phi^{-1}(p_i) = -1 + w_{i1}^2 - w_{i2}^2 + \sin(w_{i3}w_{i4}) + I(u_{i1}=2) - I(u_{i1}=3) - I(u_{i2}=2) + I(u_{i2}=3).$$

The AUC summaries for the classification are shown in the second column of Table 2. SIGN, PPMx and RF have the same AUC, $AUC = 0.84$, which is slightly lower than the AUC of BART, $AUC = 0.87$. LR and SVM do not perform well in both simulations possibly due to the parametric (linear or cubic) decision boundary in LR, and the use of SVM with linear and cubic bases, and the difficulty in tuning the model parameters in SVM with radial bases.

5 Case studies

5.1 Electronic health records data: detecting diabetes

The emergence of EHR data gives rise to great opportunities as well as challenges for data-driven approaches in early disease detection. Large sample sizes allow more efficient statistical inference but at the same time impose computational challenges, especially for flexible but computation-intensive BNP models.

We consider EHR data for $n = 85,021$ individuals in China. The dataset is based on a physical examination of residents in some districts of a major city in China conducted in 2016. We use the data to develop a model for chronic disease prediction, specifically for diabetes. We extract data on diabetes from the items "medical history" and "other current diseases" in the physical examination form. If either of the two items of a subject mention diabetes, that subject is considered as having diabetes. We denote the diabetes status by z_i (1: diabetic and 0: normal) for subjects $i = 1,\ldots,n$. Blood samples were drawn from each subject and sent to a laboratory for subsequent tests. We consider test results that are thought to be relevant to diabetes. These include white blood cell count (WBC), red blood cell count (RBC), hemoglobin (HGB), platelets (PLT), fasting blood glucose (FBG), low density lipoproteins (LDL), total cholesterol (TC), triglycerides (Trig), triketopurine (Trik), high density lipoproteins (HDL), serum creatinine (SCr), serum glutamic oxaloacetic transaminase (SGOT), and total bilirubin (TB). We also include 5 additional covariates:

gender, height, weight, blood pressure, and waist. Our goal is twofold: (1) predicting diabetes; and (2) clustering a heterogeneous population into homogeneous subpopulations.

To comply with Chinese policy, we report inference for data generated by a Generative Adversarial Network (GAN, Goodfellow et al. 2014), which replicates the distribution underlying the raw data. GAN is a machine learning algorithm which simultaneously trains a generative model and a discriminative model on a training dataset (in our case, the raw EHR dataset). The generative model simulates the training data distribution in order to simulate hypothetical additional data, which is then merged with the original data. Meanwhile, the discriminative model learns to optimally distinguish between original and simulated data. During training, the generative model uses gradient information from the discriminative model to produce better simulations. After training, the generative model can be used to generate an arbitrary number of simulations which are similar in distribution to the original dataset. In our case, we generate a simulated dataset of the same size as the raw EHR dataset.

For this application, we train on a dataset where columns of continuous variables are standardized, and corresponding output are then re-scaled at simulation time. To accommodate binary variables, we allow the GAN to simulate continuous values, and round corresponding outputs to 0 or 1. We use the architecture of MMD-GAN (Li et al., 2017), which uses maximum mean discrepancy (MMD, Gretton et al., 2012), a distributional distance, to compare real data and simulations. Our implementation uses encoder and decoder networks each containing three layers of 100 nodes, connected by a bottleneck layer of 64 nodes, and with exponential linear unit activations. In the optimization, we use RMSProp with a learning rate of 0.001, and we weight the MMD in our discriminator loss function by 0.1.

Our model reaches a stable point, where both marginal distributions and pairwise correlations agree with the raw data (see Figure 3). Moreover, the classifiers we consider have similar prediction performance on the two datasets. Therefore, we only report the results based on the replicated EHR data (referred to as EHR data hereafter). To the extent to which the preprocessed data set retains all information and structure of the original data, any inference other than subject-specific summaries remains practically unchanged. See the Appendix for more details.

Results.—We randomly sample 84,750 subjects as training data and use the remaining 271 subjects as test data to evaluate out-of-sample classification performance. We implement inference under the PPMx model using the proposed SIGN algorithm. In the implementation, we use 250 compute cores (equivalent to 11 compute nodes with 24 cores per node) at the Texas Advanced Computing Center (TACC, http://www.tacc.utexas.edu) for computation. In the first step, the training data are randomly split into $M_1 = 250$ compute cores/shards with 339 samples in each shard. Across all shards, we obtain 1351 local clusters. In the second step, the 1351 local clusters are distributed to $M_2 = 5$ shards with each shard taking about 270 items. In step 2, the local clusters are grouped into 25 regional clusters. Iteration stops there since 25 items need not be further split, i.e., $K = 3$. In a final

step, the 25 regional clusters are merged to 5 global clusters with sizes 26892, 26453, 18778, 11474, and 1153, respectively.

The AUC summaries based on the test dataset are provided in Table 2. SIGN reports the highest AUC (0.880), followed by RF, and BART. As expected, the most important covariate for predicting diabetes is FBG. Regressing on FBG alone achieves $AUC = 0.829$. In terms of computation time, SIGN, BART, RF and SVM take 0.9, 18.7, 3.5 and 2.5 hours with 2.6 GHz Xeon E5–2690 v3 CPU, respectively, whereas LR is several magnitude faster at the price of accuracy. We do not implement PPMx with standard MCMC, as this is not feasible with the large sample size. The good performance of SIGN may be explained by its ability to explicitly accommodate the heterogeneous nature of the subject population and allow for cluster-specific probit models in each subpopulation, while leveraging model averaging to classify new subjects. For example, the estimated intercept is −1.5 for cluster 2 and −0.95 for cluster 4. The coefficient of the important covariate FBG also exhibits heterogeneity, 0.97 for cluster 3 and 0.76 for cluster 4.

Finally, we evaluate the $F_{0.1}$ diagnostic proposed in Section 2.2.3. Specifically, we sample $m = 4$ out of the $M_1 = 250$ shards. We then run the SIGN approximation as well as full MCMC posterior simulation on the merged dataset of the 4 selected shards. Repeating the same procedure 62 times, we find an average $F_{0.1}$ value of 0.73.

5.2 Predicting the success of telemarketing

Direct marketing is a form of advertising where the salesperson directly communicates with the customers to promote business. In 2011, marketers are estimated to have spent $163 billion on direct marketing which accounted for 52.1% of total US advertising expenditures in that year (Direct Marketing Association INC., 2012). A common direct marketing practice is by phone, known as telemarketing. In this study, we focus on predicting the success of telemarketing in selling long-term bank deposits.

We analyze a dataset collected from a Portuguese retail bank (Moro et al., 2014) with $n = 41,188$ records. The outcome of interest is whether the customer eventually subscribed a long-term deposit: $z_i = 1$ if yes, and $z_i = 0$ otherwise, $i = 1,\ldots,n$. Associated with each record/customer are 20 covariates which are listed in Table 3. We follow Moro et al. (2014) and remove the covariate “last contact duration,” since the duration is unknown before a call is performed and therefore can not be used to predict the outcome of the next customer. After removing records that are inconsistent with the data description, the resulting dataset contains 37,078 records. We randomly sample $n = 36,750$ as training data and use the remaining 328 for testing purpose. Similarly to the analysis in Section 5.1, we apply PPMx using SIGN with $K = 3$ steps. In the first step, we randomly split the training data into $M_1 = 150$ shards (distributed on 7 compute nodes) with each shard taking 245 samples. We find 1,474 local clusters in the first step. Next, the 1,474 local clusters are then split to $M_2 = 5$ shards, with each shard processing about 295 blocks of customers. In this step, the local clusters are merged into 64 regional clusters. Finally, the 64 regional clusters are grouped into 14 global clusters with cluster sizes 7,687, 6,042, 5,725, 5,130, 3,950, 2,815, 2,101, 1,484, 975, 689, 56, 48, 26 and 22.

The classification performance evaluated on the testing dataset is reported in the last column of Table 2 for SIGN, BART, RF, LR and SVM. We find SIGN outperforms all other methods with AUC = 0.825. The second best algorithm is BART with AUC = 0.792.

6 Discussion

We have introduced SIGN as a scalable algorithm for inference on clustering under BNP mixture models. SIGN can be thought of as a parallelizable extension of the Algorithm 8 in Neal (2000), which is applicable to both conjugate and non-conjugate models. We use SIGN to implement inference under a PPMx model for a Chinese EHR dataset with 85,021 individuals and a bank telemarketing dataset with 37,078 customers. We find good classification performance compared with state-of-the-art competing methods. For the EHR study, we find five meaningful clusters in the study population. We anticipate that this study will continue to collect many more subjects over the coming years. The use of algorithms that are scalable to millions of observations in terms of both, computing time and memory is therefore imperative. The computing time for the proposed algorithm remains practicable with increasing sample size, as long as enough computing resources are available. For example, with 1,000,000 observations, SIGN runs for about 1 hour on 2,000 cores or equivalently on 80 compute nodes. This is feasible on many high performance computing centers such as TACC. Memory is less of a critical limitation. If needed, one can use one large-memory compute node (192GB on TACC) in the last step where we have to access the entire dataset.

In this paper, we only consider “large n, small p” problems. The two motivating applications include only $p = 18$ and $p = 19$ covariates. Extension to “large n, large p” problems is of practical interest for other potential applications. Another limitation of inference for the PPMx model with the current SIGN implementation is the need to access the entire dataset in the last step, which becomes computationally prohibitive for big n or p. For the PPMx, one possible strategy is to replace the cluster-specific probit model by a simpler cluster-specific Bernoulli model with the binary response. The desired dependence between response and covariate is introduced marginally, after marginalizing with respect to the partition. Under this construction the algorithm depends on the data only through low dimensional summary statistics and could handle arbitrarily large data. A similar strategy was explored in Zuanetti et al. (2018). However, introducing the dependence between response and covariates through the partition only, we find less favorable classification performance than in the current implementations (results not shown).

Supplementary Material

Refer to Web version on PubMed Central for supplementary material.

Acknowledgment

Yang Ni, Peter Müller and Yuan Ji’s research were partially supported by NIH/NCI grant a R01 CA 132897. Maurice Diesendruck and Sinead Williamson were partially supported by NSF IIS-1447721. The authors acknowledge the TACC at The University of Texas at Austin for providing high performance computing resources that have contributed to the research results reported within this paper.

Appendix: GAN preprocessing details

To evaluate the privacy of the simulated set, we measure two types of risk: presence disclosure and attribute disclosure (Choi et al., 2017). Presence disclosure is the ability to determine whether a candidate point was included in the training dataset. Attribute disclosure is the ability to predict other attributes of a candidate point, given partial information about that point. For both settings, we choose three sets of equal size – 5% of the training data, a heldout set for testing, and a heldout set for baseline comparison – then estimate the sensitivity and precision of classification schemes.

For presence disclosure, we sample a candidate from the union of training and testing sets, and classify whether the candidate was included in the training set based on the presence of an ϵ-close neighbor in the simulated set. For large ϵ, the notion of ϵ-closeness is not informative, since many points are returned as neighbors, and precision scores hover around 50% – no better than random guessing. For small ϵ, few points are returned as neighbors, and neighbors are more likely to be correctly guessed, since the requirement is for a neighbor to be nearly identical to the candidate point. To reflect the optimal privacy standard, we report scores using the largest ϵ for which precision exceeds 55%. This yields the largest sensitivity under nontrivial risk, where a higher sensitivity indicates greater ability to identify a participant. At $\epsilon = 9.5$, the sensitivity of this classification is 0.0005, indicating that compromised training points would be identifiable only 0.05% of the time.

For attribute disclosure, we sample as above, and classify whether unknown features of a candidate point can be correctly estimated to within 5% of the true value, by averaging each feature over the candidate's five nearest neighbors in the simulated set. We report values for the case in which half of the candidate's features are known, and the other half are imputed, and note that performance did not change significantly when the percentage of known values differed. The sensitivity and precision scores of this classification are 0.31 and 0.72, respectively, indicating that unknown features would be correctly guessed 31% of the time, and features claiming to be within 5% of the true value would in fact be 72% of the time.

We note that privacy and accuracy goals are inherently opposed. An increase in privacy corresponds to a simulated set with less information about individual data points, and vice versa. As a general guideline, we aim to satisfy privacy requirements while preserving as much as possible the utility of the simulations. In the specific case of attribute risk, we recognize that scores depend on the correlation structure of the data, where highly correlated features are more susceptible to attribute disclosure. As a baseline, we compared attribute risk scores of simulations to those of the final heldout set, and found that both were approximately 30% and 70%, respectively.

References

Argiento R, Cremaschi A, and Guglielmi A (2014). A "density-based" algorithm for cluster analysis using species sampling Gaussian mixture models. Journal of Computational and Graphical Statistics, 23(4):1126–1142.

Argiento R, Guglielmi A, and Pievatolo A (2010). Bayesian density estimation and model selection using nonparametric hierarchical mixtures. Computational Statistics & Data Analysis, 54(4):816–832.

Bardenet R, Doucet A, and Holmes C (2014). Towards scaling up Markov chain Monte Carlo: an adaptive subsampling approach. In Proceedings of the 31st International Conference on Machine Learning, pages 405–413.

Bardenet R, Doucet A, and Holmes C (2015). On Markov chain Monte Carlo methods for tall data. arXiv preprint arXiv:1505.02827.

Barrios E, Lijoi A, Nieto-Barajas LE, and Prünster I (2013). Modeling with normalized random measure mixture models. Statist. Sci, 28(3):313–334.

Breiman L, Friedman J, Stone CJ, and Olshen RA (1984). Classification and regression trees. Wadsworth, Belmont, CA.

Broderick T, Boyd N, Wibisono A, Wilson AC, and Jordan MI (2013). Streaming variational Bayes. In Advances in Neural Information Processing Systems, pages 1727–1735.

Chipman H and McCulloch R (2016). BayesTree: Bayesian Additive Regression Trees. R package version 0.3–1.4

Chipman HA, George EI, McCulloch RE, et al. (2010). BART: Bayesian additive regression trees. The Annals of Applied Statistics, 4(1):266–298.

Choi E, Biswal S, Malin B, Duke J, Stewart WF, and Sun J (2017). Generating multi-label discrete patient records using generative adversarial networks. In Proceedings of the 2nd Machine Learning for Healthcare Conference, pages 286–305.

Cortes C and Vapnik V (1995). Support-vector networks. Machine Learning, 20(3):273–297.

Cruz-Mesía R. D. l., Quintana FA, and Müller P (2007). Semiparametric Bayesian classification with longitudinal markers. Journal of the Royal Statistical Society: Series C, 56(2):119–137.

Dahl DB (2006). Model-based clustering for expression data via a Dirichlet process mixture model In Do K-A, Müller P, and Vannucci M, editors, Bayesian Inference for Gene Expression and Proteomics, pages 201–218. Cambridge University Press, Cambridge.

De Blasi P, Favaro S, Lijoi A, Mena RH, Prünster I, and Ruggiero M (2015). Are Gibbs-type priors the most natural generalization of the Dirichlet process? IEEE Transactions on Pattern Analysis and Machine Intelligence, 37(2):212–229. [PubMed: 26353237]

Dean J and Ghemawat S (2008). MapReduce: simplified data processing on large clusters. Communications of the ACM, 51(1):107–113.

Dellaportas P and Papageorgiou I (2006). Multivariate mixtures of normals with unknown number of components. Statistics and Computing, 16(1):57–68.

Direct Marketing Association INC. (2012). The power of direct marketing: ROI, sales, expenditures and employment in the US, 2011–2012. Direct Marketing Association, Washington, DC.

Escobar MD (1994). Estimating normal means with a Dirichlet process prior. Journal of the American Statistical Association, 89(425):268–277.

Ester M, Kriegel H-P, Sander J, Xu X, et al. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In KDD, pages 226–231.

Fahad A, Alshatri N, Tari Z, Alamri A, Khalil I, Zomaya AY, Foufou S, and Bouras A (2014). A survey of clustering algorithms for big data: Taxonomy and empirical analysis. IEEE Transactions on Emerging Topics in Computing, 2(3):267–279.

Favaro S and Teh YW (2013). MCMC for normalized random measure mixture models. Statist. Sci, 28(3):335–359.

Friedman J, Hastie T, and Tibshirani R (2010). Regularization paths for generalized linear models via coordinate descent. Journal of Statistical Software, 33(1):1–22. [PubMed: 20808728]

Ge H, Chen Y, Wan M, and Ghahramani Z (2015). Distributed inference for Dirichlet process mixture models. In Proceedings of the 32nd International Conference on Machine Learning, pages 2276–2284.

Ghahramani Z and Beal MJ (2001). Propagation algorithms for variational Bayesian learning. In Advances in Neural Information Processing Systems, pages 507–513.

Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, and Bengio Y (2014). Generative adversarial nets. In Advances in Neural Information Processing Systems, pages 2672–2680.

Green PJ, Łatuszyński K, Pereyra M, and Robert CP (2015). Bayesian computation: a perspective on the current state, and sampling backwards and forwards. arXiv preprint arXiv:1502.01148.

Gretton A, Borgwardt KM, Rasch MJ, Schölkopf B, and Smola A (2012). A kernel two-sample test. Journal of Machine Learning Research, 13:723–773.

Gutiérrez L, Gutiérrez-Peña E, and Mena RH (2014). Bayesian nonparametric classification for spectroscopy data. Computational Statistics & Data Analysis, 78:56–68.

Hahsler M and Piekenbrock M (2018). dbscan: Density Based Clustering of Applications with Noise (DBSCAN) and Related Algorithms. R package version 1.1–3.

Hartigan JA (1990). Partition models. Communications in Statistics, 19(8):2745–2756.

Hartigan JA and Wong MA (1979). Algorithm as 136: A k-means clustering algorithm. Journal of the Royal Statistical Society. Series C, 28(1):100–108.

Hjort NL, Holmes C, Müller P, and Walker SG (2010). Bayesian Nonparametrics Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, Cambridge.

Ho TK (1995). Random decision forests. In Proceedings of the Third International Conference on Document Analysis and Recognition, pages 278–282.

Hoffman MD, Blei DM, Wang C, and Paisley J (2013). Stochastic variational inference. The Journal of Machine Learning Research, 14(1):1303–1347.

Huang Z and Gelman A (2005). Sampling for Bayesian computation with large datasets Technical report, Department of Statistics, Columbia University.

Jaakkola TS and Jordan MI (2000). Bayesian parameter estimation via variational methods. Statistics and Computing, 10(1):25–37.

Jain AK (2010). Data clustering: 50 years beyond k-means. Pattern Recognition Letters, 31(8):651–666.

Kingman JFC (1978). The representation of partition structures. Journal of the London Mathematical Society, s2–18(2):374–380.

Kleiner A, Talwalkar A, Sarkar P, and Jordan MI (2014). A scalable bootstrap for massive data. Journal of the Royal Statistical Society: Series B, 76(4):795–816.

Korattikara A, Chen Y, and Welling M (2014). Austerity in MCMC land: Cutting the Metropolis-Hastings budget. In Proceedings of the 31st International Conference on Machine Learning, pages 181–189.

Lau JW and Green PJ (2007). Bayesian model-based clustering procedures. Journal of Computational and Graphical Statistics, 16(3):526–558.

Lee J, Quintana FA, Müuller P, and Trippa L (2013). Defining predictive probability functions for species sampling models. Statistical Science, 28(2):209–222. [PubMed: 24368874]

Li C-L, Chang W-C, Cheng Y, Yang Y, and Póczos B (2017). MMD GAN: Towards deeper understanding of moment matching network. In Advances in Neural Information Processing Systems, pages 2200–2210.

Liaw A and Wiener M (2002). Classification and regression by randomforest. R News, 2(3):18–22.

Lijoi A, Mena RH, and Prünster I (2005). Hierarchical mixture modeling with normalized inverse-Gaussian priors. Journal of the American Statistical Association, 100(472):1278–1291.

Lijoi A, Mena RH, and Prünster I (2007). Controlling the reinforcement in Bayesian non-parametric mixture models. Journal of the Royal Statistical Society: Series B, 69(4):715–740.

Lin D (2013). Online learning of nonparametric mixture models via sequential variational approximation. In Advances in Neural Information Processing Systems, pages 395–403.

Lo AY (1984). On a class of Bayesian nonparametric estimates: I. Density estimates. The Annals of Statistics, 12(1):351–357.

MacEachern SN (2000). Dependent Dirichlet processes. Unpublished manuscript, Department of Statistics, The Ohio State University, pages 1–40.

MacEachern SN and Müller P (1998). Estimating mixture of Dirichlet process models. Journal of Computational and Graphical Statistics, 7(2):223–238.

Malsiner-Walli G, Frühwirth-Schnatter S, and Grün B (2017). Identifying mixtures of mixtures using Bayesian estimation. Journal of Computational and Graphical Statistics, 26(2):285–295. [PubMed: 28626349]

Mansinghka VK, Roy DM, Rifkin R, and Tenenbaum J (2007). AClass: An online algorithm for generative classification. In Proceedings of the 11th International Conference on Artificial Intelligence and Statistics, pages 315–322.

Meilă M (2007). Comparing clusterings—an information based distance. Journal of Multivariate Analysis, 98(5):873–895.

Meyer D, Dimitriadou E, Hornik K, Weingessel A, and Leisch F (2018). e1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien. R package version 1.7–0.

Minsker S, Srivastava S, Lin L, and Dunson D (2014). Scalable and robust Bayesian inference via the median posterior. In International Conference on Machine Learning, pages 1656–1664.

Moler C (1986). Matrix computation on distributed memory multiprocessors. Hypercube Multiprocessors, 86(181–195):31.

Moro S, Cortez P, and Rita P (2014). A data-driven approach to predict the success of bank telemarketing. Decision Support Systems, 62:22–31.

Müller P, Quintana F, and Rosner GL (2011). A product partition model with regression on covariates. Journal of Computational and Graphical Statistics, 20(1):260–278. [PubMed: 21566678]

Neal RM (2000). Markov chain sampling methods for Dirichlet process mixture models. Journal of Computational and Graphical Statistics, 9(2):249–265.

Neiswanger W, Wang C, and Xing E (2013). Asymptotically exact, embarrassingly parallel MCMC. arXiv preprint arXiv:1311.4780.

Ni Y, Müller P, Zhu Y, and Ji Y (2018). Heterogeneous reciprocal graphical models. Biometrics, just accepted.

Payne RD and Mallick BK (2018). Two-stage Metropolis-Hastings for tall data. Journal of Classification, just accepted.

Pennell ML and Dunson DB (2007). Fitting semiparametric random effects models to large data sets. Biostatistics, 8(4):821–834. [PubMed: 17429104]

Pitman J and Yor M (1997). The two-parameter Poisson-Dirichlet distribution derived from a stable subordinator. The Annals of Probability, pages 855–900.

Quiroz M, Kohn R, Villani M, and Tran M-N (2018). Speeding up MCMC by efficient data subsampling. Journal of the American Statistical Association, (just-accepted):1–35. [PubMed: 30034060]

R Core Team (2018). R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria.

Rastelli R and Friel N (2018). Optimal Bayesian estimators for latent variable cluster models. Statistics and Computing, 28:1169–1186. [PubMed: 30220822]

Rebentrost P, Mohseni M, and Lloyd S (2014). Quantum support vector machine for big data classification. Physical Review Letters, 113(13):130503. [PubMed: 25302877]

Richardson S and Green PJ (1997). On Bayesian analysis of mixtures with an unknown number of components (with discussion). Journal of the Royal Statistical Society: series B, 59(4):731–792.

Rodriguez A, Lenkoski A, and Dobra A (2011). Sparse covariance estimation in heterogeneous samples. Electronic Journal of Statistics, 5:981. [PubMed: 26925189]

Scott SL, Blocker AW, Bonassi FV, Chipman HA, George EI, and McCulloch RE (2016). Bayes and big data: The consensus Monte Carlo algorithm. International Journal of Management Science and Engineering Management, 11(2):78–88.

Singh A, Thakur N, and Sharma A (2016). A review of supervised machine learning algorithms In Computing for Sustainable Global Development (INDIACom), 2016 3rd International Conference on, pages 1310–1315. IEEE.

Tank A, Foti N, and Fox E (2015). Streaming variational inference for Bayesian nonparametric mixture models. In Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics, pages 968–976.

Wade S and Ghahramani Z (2018). Bayesian cluster analysis: Point estimation and credible balls (with discussion). Bayesian Analysis, 13(2):559–626.

Walker SG (2007). Sampling the Dirichlet mixture model with slices. Communications in Statistics, 36(1):45–54.

Wang L and Dunson DB (2011). Fast Bayesian inference in Dirichlet process mixture models. Journal of Computational and Graphical Statistics, 20(1):196–216.

Wang X and Dunson DB (2013). Parallelizing MCMC via Weierstrass sampler. arXiv preprint arXiv:1312.4605.

Welling M and Teh YW (2011). Bayesian learning via stochastic gradient Langevin dynamics. In Proceedings of the 28th International Conference on Machine Learning, pages 681–688.

White S, Kypraios T, and Preston S (2015). Piecewise Approximate Bayesian Computation: fast inference for discretely observed Markov models using a factorised posterior distribution. Statistics and Computing, 25(2):289. [PubMed: 26097293]

Williamson SA, Dubey A, and Xing EP (2013). Parallel Markov chain Monte Carlo for nonparametric mixture models. In Proceedings of the 30th International Conference on International Conference on Machine Learning, pages 98–106.

Zhang Y, Wainwright MJ, and Duchi JC (2012). Communication-efficient algorithms for statistical optimization. In Advances in Neural Information Processing Systems, pages 1502–1510.

Zhao W, Ma H, and He Q (2009). Parallel k-means clustering based on MapReduce In IEEE International Conference on Cloud Computing, pages 674–679. Springer.

Zuanetti DA, Müller P, Zhu Y, Yang S, and Ji Y (2018). Bayesian nonparametric clustering for large data sets. Statistics and Computing, just accepted.

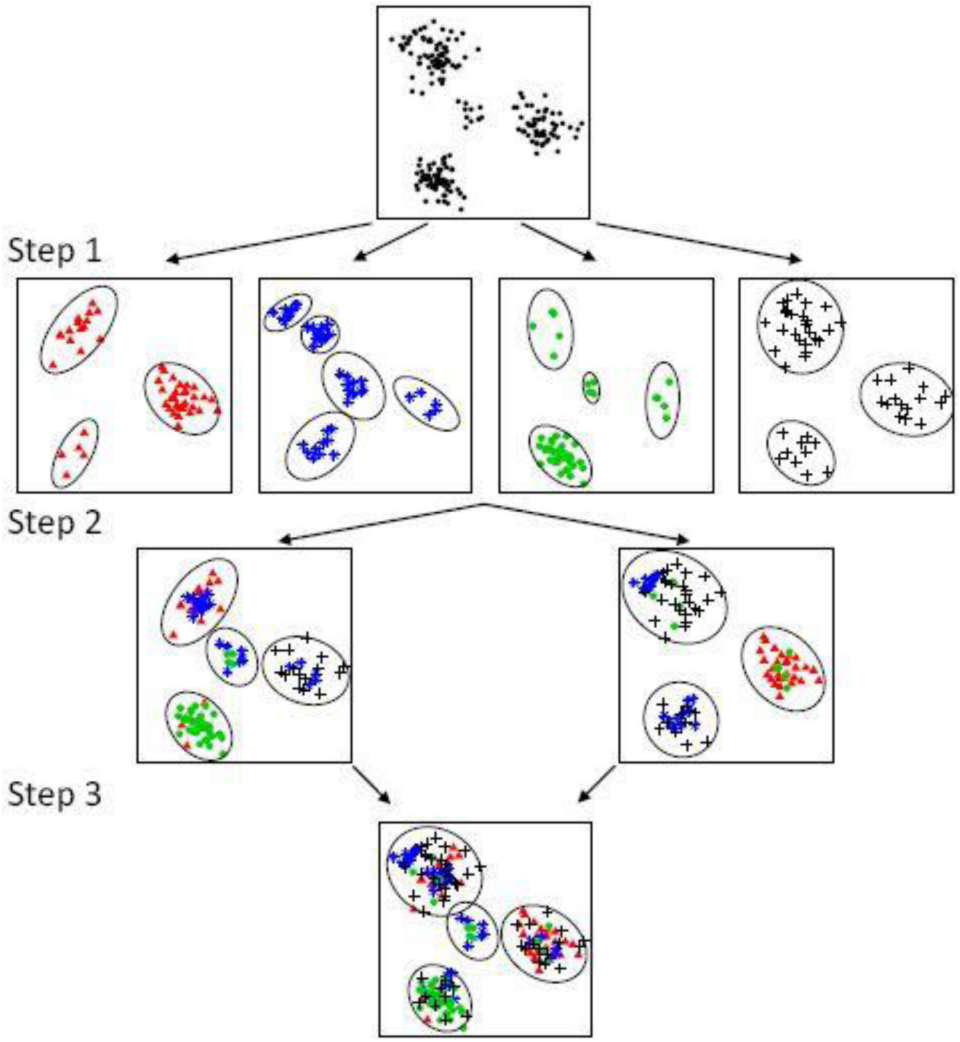


Fig. 1.

Example workflow of a 3-step SIGN algorithm. Step 1: the dataset is randomly distributed into 4 shards, each denoted by a unique type (color) of marker and observations are partitioned into local clusters (represented by the ellipses) within each shard in parallel. Step 2: local clusters are collected, randomly distributed into $M_2 = 2$ shards, and then partitioned into regional clusters within each shard. Step 3: regional clusters are aggregated and partitioned into global clusters.

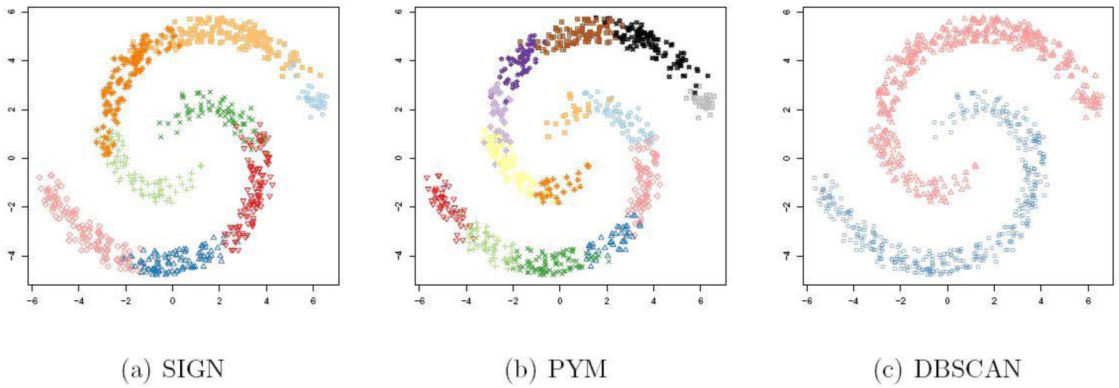
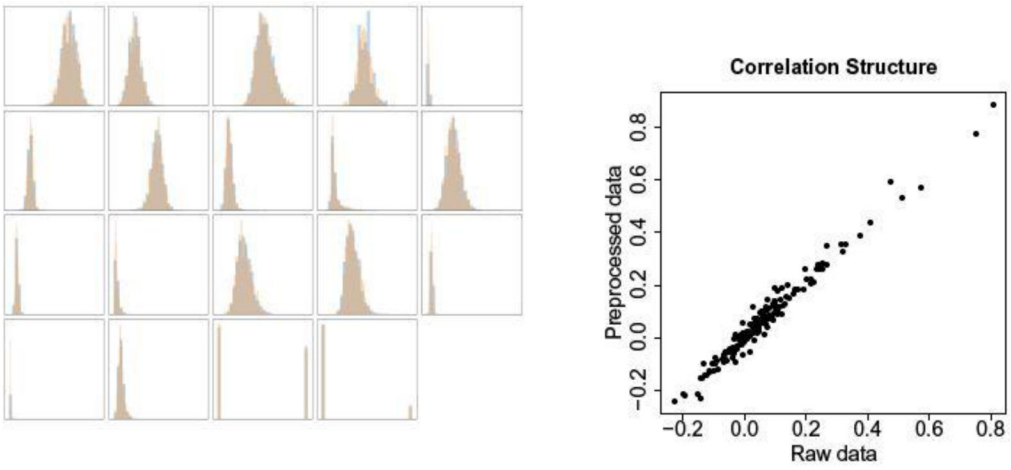


Fig. 2.

Two-spiral data. One randomly selected simulation result for inference under (a) SIGN, (b) standard implementation of PYM, and (c) DBSCAN. Marker types indicate clusters.



(a) Marginal distributions (b) Pairwise correlations

Fig. 3.

GAN-preprocessed EHR data versus raw EHR data. (a) Marginal distribution of each variable. For each variable, the two overlaid histograms show the agreement between the preprocessed and the raw data. (Variable names and ranges are deliberately not shown.) (b) Correlation of each pair of variables. Each dot represents the Pearson correlation coefficients of one pair of variables in the raw EHR data (x-axis) versus the same in the GAN-preprocessed EHR data (y-axis). In total, we have $\binom{19}{2}$ pairs/dots.

Table 1

Clustering performance of the methods used for Simulation I. The table reports the average estimated number of clusters (C), misclustering rate (MISC) and mean squared error in estimating cluster-specific means μb (MSE) for inference under SIGN, SIGN with variation of information loss (SIGN-VI), standard implementation of Pitman-Yor process mixture (PYM) and DPSCAN. Numerical errors (as standard deviations over repeat simulation) are given within the parentheses.

	SIGN	**SIGN-VI**	**PYM**	**DBSCAN**
C	4.94 (0.31)	4.90 (0.30)	5.08 (0.27)	3.64 (1.63)
MISC	0.08 (0.03)	0.09 (0.03)	0.04 (0.01)	0.41 (0.11)
MSE	0.01 (0.01)	0.01 (0.01)	0.01 (0.00)	0.49 (0.18)

Table 2

Performance of the methods used for Simulations III and IV, and the two case studies. The table reports AUC for inference under the SIGN approximation, (standard implementation of) PPMx, BART, RF, LR and SVM. Numerical errors (as standard deviations over repeat simulation) are given within the parentheses.

	Simulation III	**Simulation IV**	**EHR**	**Bank**
SIGN	0.808 (0.067)	0.838 (0.067)	0.880	0.825
PPMx	0.824 (0.060)	0.841 (0.063)	-	-
BART	0.755 (0.062)	0.866 (0.050)	0.867	0.792
RF	0.793 (0.059)	0.838 (0.067)	0.869	0.786
LR	0.600 (0.091)	0.524 (0.073)	0.856	0.781
SVM	0.622 (0.077)	0.585 (0.077)	0.856	0.761

Table 3

20 Covariates in the long-term deposit data. For categorical covariates, the number within the parentheses indicates the number of categories.

Covariate name	**Type**
Type of job	Categorical (12)
Marital status	Categorical (4)
Education	Categorical (8)
Default or not	Categorical (3)
Housing loan or not	Categorical (3)
Contact communication type	Categorical (2)
Last contact month of year	Categorical (12)
Last contact day of the week	Categorical (5)
Outcome of the previous campaign	Categorical (3)
Age	Continuous
Last contact duration	Continuous
Number of contacts	Continuous
Number of days from a previous campaign	Continuous
Number of contacts before this campaign	Continuous
Employment variation rate	Continuous
Consumer price index	Continuous
Consumer confidence index	Continuous
Euribor 3 month rate	Continuous
Number of employees	Continuous