Particle Gibbs Split-Merge Sampling for Bayesian Inference in Mixture Models

Alexandre Bouchard-Côté   BOUCHARD@STAT.UBC.CA
Department of Statistics
University of British Columbia
Corresponding address: 3182 Earth Sciences Building, 2207 Main Mall, Vancouver, BC, Canada
V6T 1Z4

Arnaud Doucet   DOUCET@STATS.OX.AC.UK
Department of Statistics
University of Oxford, United Kingdom

Andrew Roth   ANDREW.ROTH@LUDWIG.OX.AC.UK
Department of Statistics and Ludwig Institute for Cancer Research
University of Oxford, United Kingdom

Abstract

This paper presents an original Markov chain Monte Carlo method to sample from the posterior distribution of conjugate mixture models. This algorithm relies on a flexible split-merge procedure built using the particle Gibbs sampler introduced in Andrieu et al. (2009, 2010). The resulting so-called Particle Gibbs Split-Merge sampler does not require the computation of a complex acceptance ratio and can be implemented using existing sequential Monte Carlo libraries. We investigate its performance experimentally on synthetic problems as well as on geolocation data. Our results show that for a given computational budget, the Particle Gibbs Split-Merge sampler empirically outperforms existing split merge methods. The code and instructions allowing to reproduce the experiments is available at https://github.com/aroth85/pgsm.

Keywords: Dirichlet process mixture models; Gibbs sampler; Particle Gibbs sampler; Sequential Monte Carlo.

1. Introduction

Mixture models are very commonly used to perform clustering and density estimation, and they have consequently found numerous applications in a wide range of scientific fields. Since the introduction of Markov chain Monte Carlo (MCMC) methods in statistics over twenty five years ago, the Bayesian approach to mixture models has become very popular (Marin et al., 2005; Richardson and Green, 1997). However, sampling from the posterior distribution of mixture models remains a challenging computational problem.

When conjugate priors are used, it is possible to analytically integrate out the mixing proportions and the parameters of the components. This is the scenario we will focus on in this article. In this case, we aim to sample from the posterior distribution of the latent indicator variables associated with the observations, each latent variable indicating which

component of the mixture generates a given data point. A simple Gibbs sampler can be used which updates the latent indicator variables one-at-a-time (MacEachern, 1994; Escobar and West, 1995) but this algorithm is inefficient when the number of observed data points T is large. First, the simulated Markov chain would have to visit a long chain of lower probability configurations in order to split and merge large clusters. As a result, it is prone to getting trapped in severe local modes. Second, it is non-trivial to parallelize due to the inherently sequential nature of the updates.

The limitations of the simple Gibbs sampler has motivated a rich literature on MCMC algorithms for Bayesian mixture models which partially address these issues; see, e.g., Ishwaran and James (2001); Liang et al. (2007); Walker (2007); Kalli et al. (2011). In particular, procedures proposing to split and merge existing clusters in one single step have become prominent as they generally perform better than the simple Gibbs sampler (Richardson and Green, 1997; Neal, 2000; Dahl, 2003; Jain and Neal, 2004).

While designing an efficient merge proposal is simple, designing an efficient split proposal is a more complicated task. When the mixing proportions and parameters are not integrated out, split-merge moves were first proposed in Richardson and Green (1997). The proposals were built to ensure the conservation of some moments and accepted/rejected using Metropolis-Hastings steps. However, it is difficult to design efficient proposals in this context.

When the mixing proportions and parameters are integrated out, split-merge moves on the latent indicator variables were first proposed in Jain and Neal (2004). Assume one is interested in splitting a block/cluster of points $b \subset \{1, \ldots, T\}$ into two blocks. We select two points in b, which will be in distinct blocks after the split. There are $2^{|b|-2}$ possible ways to split the original block b, hence any efficient proposal needs to be informed by the observations corresponding to the indices in b. In Jain and Neal (2004), one selects two points at random which are used as anchors. When the two anchors are in separate clusters, a merging of the two clusters is proposed. When the two anchors are in the same cluster, a split is proposed as follows: first, the two anchor points seed a pair of new clusters, and second, several restricted Gibbs scans are performed to reallocate the remaining points originally clustered with the anchors to the two new clusters. All clusters which do not contain the anchors are not altered, leading to a restricted Gibbs move. After either a split or a merge is proposed, the Metropolis-Hastings ratio is computed to accept or reject the move. The number of Gibbs scans in the split move is a free tuning parameter for this sampler. In Dahl (2003), an alternative approach is proposed for split moves. The restricted Gibbs scans are replaced by a sequential allocation step whereby the anchors define two new clusters and all points which were originally clustered with these points are sequentially allocated to one of the anchor clusters.

These split-merge algorithms have become popular as they provide state-of-the-art performance but they are relatively difficult to implement due to their complex Metropolis-Hastings acceptance ratios.

In the present work, we propose a novel split-merge sampler based on the conditional Sequential Monte Carlo (SMC) algorithm appearing in the Particle Gibbs (PG) sampler (Andrieu et al., 2009, 2010), which we call the Particle Gibbs Split Merge (PGSM) sampler. Most of the complexity inherent to split-merge operators is encapsulated into the well-understood PG sampling procedure (Chopin and Singh, 2015), and no acceptance ratio

needs to be computed. Moreover, as the PGSM sampler relies on SMC methods, it benefits from advanced simulation methods from the SMC literature, such as adaptation schemes (Lee, 2011) and methods for parallel and distributed inference (Lee et al., 2010; Jun et al., 2012; Lee and Whiteley, 2016), as well as from efficient SMC software libraries (Johansen, 2009; Murray, 2015). The PGSM sampler does not make any topological assumption on the observation space in contrast to the posterior simulation techniques described in Dahl (2009) and Liang et al. (2007). This methodology complements the maximum *a posteriori* inference techniques developed in Daumé III (2007); Wang and Dunson (2011).

There has been previous work on applying sequential importance sampling and SMC methods for posterior simulation of Dirichlet processes and related mixture models. However, to the best of our knowledge, SMC methods have never been previously used to design split-merge moves. Indeed, the methods proposed in MacEachern et al. (1999); Fearnhead (2004); Fearnhead and Meligkotsidou (2007); Mansinghka et al. (2007); Caron and Doucet (2009); Carvalho et al. (2010) directly apply a single pass SMC algorithm to the entire clustering problem. Empirical results in Kantas et al. (2015) suggest that such methods may require a number of particle which scales at least quadratically with respect to the number of datapoints. The work of Ülker et al. (2010) uses SMC within the context of the SMC Samplers methodology (Del Moral et al., 2006), which makes it closer in spirit to existing MCMC methods. Our contribution is to provide a principled approach for breaking down the clustering problem into smaller sub-problems more amenable to the use of SMC techniques.

Finally, other lines of work are devoted to parallelization and distribution of MCMC methods for mixture models (Chang and Fisher III, 2013; Williamson et al., 2013; Gal and Ghahramani, 2014; Ge et al., 2015). As alluded to earlier, our method can potentially be parallelized and distributed using existing approaches from the SMC literature (Lee et al., 2010; Jun et al., 2012; Lee and Whiteley, 2016). Like the other available split-merge procedures, it is also possible to consider different split-merge moves simultaneously when the prior clustering distribution restricted to the clusters being updated does not depend on the number of clusters for the whole dataset. However, we do not focus on these aspects here.

The rest of this article is organized as follows. Section 2 introduces our notation for the types of Bayesian mixture models that we consider. Section 3 details the PGSM sampler. Section 4 applies the method to synthetic datasets, as well as real data from a geolocation application. We conclude with some directions for future work and discussion in Section 5.

2. Mixture models and Bayesian inference

In this section we first layout notation and then describe Bayesian mixture models. We focus on the case where the component base measure is conjugate to the data likelihood, so that the posterior distribution of any clustering can be evaluated analytically up to a normalizing constant.

2.1 Notation

We use bold letters for (random) vectors, and normal fonts for (random) scalars, sets, and matrices. For quantities such as an individual observation y_i, or a parameter θ, which can

be either scalars or vectors without affecting our methodology, we consider them as scalars without loss of generality. Given a vector $\mathbf{x} = (x_1, x_2, \ldots, x_n)$, and $i \leq j$, we use $\mathbf{x}_{i:j}$ to denote the sub-vector $\mathbf{x}_{i:j} = (x_i, x_{i+1}, \ldots, x_j)$. To simplify notation, we do not distinguish random variables from their realization. We define discrete probability distributions with their probability mass functions, and continuous probability distributions with their density functions with respect to the Lebesgue measure. A list of symbols is available in the Appendix.

2.2 Bayesian mixture model

Consider T observations $\mathbf{y} := (y_1, \ldots, y_T)$. A mixture model assumes that the observations indices $[T] := \{1, \ldots, T\}$ are partitioned into subsets. This partition is called a clustering, $c := \{b_1, \ldots, b_{|c|} : b_k \subseteq [T]\}$ where $|c|$ denotes the cardinality of the set c and each block b in the partition is referred to as a cluster. Given the clustering c, we define the following likelihood for the data

$$\pi(\mathbf{y}|c) := \prod_{b \in c} L(\mathbf{y}_b), \tag{1}$$

where $L(\mathbf{y}_b)$ is the likelihood of the observations in cluster b

$$L(\mathbf{y}_b) := \int \left(\prod_{i \in b} L(y_i|\theta) \right) H(\mathrm{d}\theta). \tag{2}$$

In this expression, $L(y_i|\theta)$ is a probability density function parametrized by θ and $H(\mathrm{d}\theta)$ a prior measure over this parameter.

The clustering c is unknown and is viewed as a random variable. Let $\tau(c)$ denote its prior probability, defined over the space of partitions of $[T]$ and assumed to factorize as

$$\tau(c) \propto \tau_1(|c|) \prod_{b \in c} \tau_2(|b|), \tag{3}$$

where $\tau_1 : \mathbb{N} \to \mathbb{R}^+$ and $\tau_2 : \mathbb{N} \to \mathbb{R}^+$ are arbitrary functions.

This assumption on the prior clustering distribution is not restrictive and includes several popular priors, such as:

Dirichlet process prior (Ferguson, 1973) with parameter $\alpha_0 > 0$: $\tau_1(j) \propto \alpha_0{}^j$ and $\tau_2(j) \propto (j-1)!$.

Pitman-Yor process prior (Pitman and Yor, 1997; Ishwaran and James, 2003) with parameters α_0, d ($\alpha_0 > -d$, $0 \leq d < 1$): $\tau_1(j) \propto \prod_{j'=1}^{j} \{\alpha_0 + d(j'-1)\}$ and $\tau_2(j) \propto \Gamma(j-d)$, where $\Gamma(\cdot)$ is the Gamma function.

Finite Dirichlet mixture with parameter $\alpha_{\mathrm{Dir}} > 0$ and k_0 components and symmetric concentration $(\alpha_{\mathrm{Dir}}, \ldots, \alpha_{\mathrm{Dir}})$: $\tau_1(j) \propto \mathbf{1}[j \leq k_0]$ and $\tau_2(j) \propto \Gamma(j + \alpha_{\mathrm{Dir}})$.

The likelihood (1)-(2) and prior (3) define the following target posterior distribution

$$\pi(c) := \pi(c \mid \mathbf{y}) \propto \tau(c) \prod_{b \in c} L(\mathbf{y}_b). \tag{4}$$

4

Since we view the observations as fixed, we drop the dependency on **y** from the notation throughout the paper. We detail in the following sections an original approach to sample from this posterior distribution.

3. Methodology

We organize the description of our method into two main parts. First, we define a generic construction for decomposing the problem of sampling from the posterior (Equation (4)) with arbitrary numbers of clusters into *split-merge* sub-problems. Second, we show how the PG methodology can be used to address these sub-problems.

3.1 Decomposing the clustering problem into split-merge subproblems

Algorithm 1 allows us to break down the problem of sampling from the posterior into split-merge subproblems. We refer the reader to Figure 1 for an illustrative example of the notation used throughout this description.

Algorithm 1

1: **function** SPLITMERGE($c, h(s), K_{c,s}(\bar{c}'|\bar{c})$)
2:      $s \sim h(\cdot)$ $\triangleright \ s = \{i_1, i_2\}$
3:      $\bar{c} \leftarrow \{b \in c : b \cap \bar{s} \neq \emptyset\}$ $\triangleright$ Clustering restricted to the anchors
4:      $\bar{s} \leftarrow \bigcup_{b \in \bar{c}} b$ $\triangleright$ Closure of the anchors with respect to the clustering $\bar{c}$
5:      $\bar{c}' \sim K_{c,s}(\cdot|\bar{c})$
6:      $c' \leftarrow \bar{c}' \cup (c \backslash \bar{c})$
7:      **return** c'
8: **end function**

The algorithm requires three inputs:

1. c: the current clustering,

2. $h(s)$: a distribution for proposing an unordered pair of *anchors* $s = \{i_1, i_2\} \subset [T]$,

3. $K_{c,s}(\bar{c}'|\bar{c})$: a Markov transition kernel over the space of partitions of $\bar{s}$. This kernel is assumed to be invariant with respect to the following target distribution:

$$\bar{\pi}_{c,s}(\bar{c}') \quad \propto \quad \bar{\tau}_1(|\bar{c}'|) \left(\prod_{b \in \bar{c}'} \tau_2(|b|) L(\mathbf{y}_b) \mathbf{1} \left[b \cap s \neq \emptyset \right] \right), \quad (5)$$

$$\bar{\tau}_1(j) \quad := \quad \tau_1(j + |c| - |\bar{c}|).$$

In the following, we drop the subscripts from the kernel K and target $\bar{\pi}$ for simplicity.

The distribution $\bar{\pi}$ has a form similar to the posterior distribution defined in Equation (4) with two modifications. First, $\tau_1(|c|)$ is replaced by $\bar{\tau}_1(|\bar{c}'|)$. Second, the support of the distribution is restricted so that each block in $\bar{c}'$ must contain at least one anchor point. This also implicitly enforces the constraint that $|\bar{c}'| \leq |s| = 2$.

Algorithm 1 returns an updated clustering where only the allocation of points in $\bar{s}$ have changed; i.e. the updated clustering c' only potentially differs from c at points which were
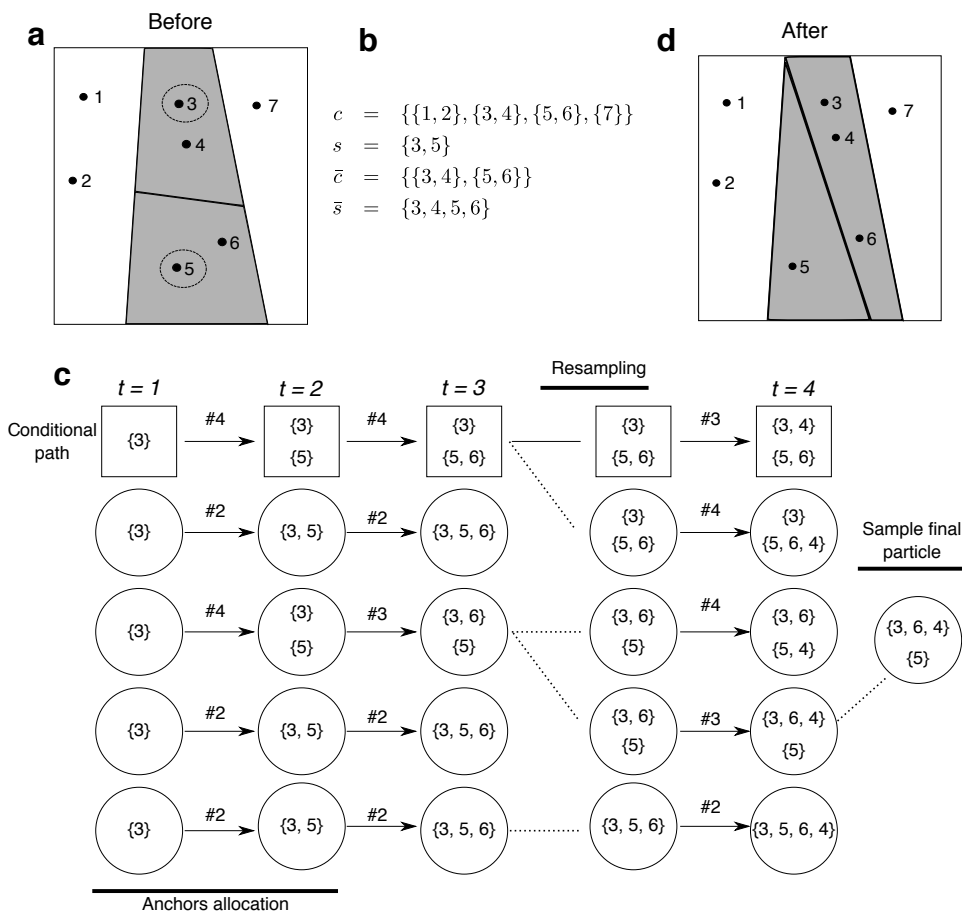
Figure 1: (**a**) Example illustrating the setup of a split merge and (**b**) the notation used in Section 3.1. The seven points denote the observation indices. While we show points in a two-dimensional space for visualization, the methodology does not rely on any topological properties of the observation space. The dashed circles denote the anchors s, and the shaded region, the closure of the anchors. (**c**) Example of a single PGSM step with $\boldsymbol{\sigma} = (3, 5, 6, 4)$. Square boxes represent particles corresponding to the fixed conditional path. Circles represent regular particles. Each particle has an associated clustering denoted by the set or pair of sets written inside each particle. Arrows indicate proposal draws labelled by the proposed state (see Figure 2). Dashed lines indicate when resampling occurs. (**d**) Configuration after the PGSM update illustrated in **c**.

initially clustered with the anchor points. The anchor proposal distribution h obviously impacts the performance of this procedure. We empirically compare the performance of three anchor proposal distributions h in Section 4.3.

This scheme has the following property:

Proposition 1 *If $c \sim \pi$, where π is given by Equation (4), then the output of Algorithm 1, c', satisfies $c' \sim \pi$. That is, the Markov kernel $K(c'|c)$ induced by Algorithm 1 is π-invariant.*

3.2 Overview of the particle Gibbs algorithm

Ideally, we would like to sample independently from $\bar{\pi}$ in Algorithm 1, that is we would like to have $K(\bar{c}'|\bar{c}) = \bar{\pi}(\bar{c}')$, but this is too computationally expensive if $|\bar{s}|$ is large. Our primary contribution is an original way to address this issue using SMC-based methods.

In principle, it would be possible to use SMC methods to obtain a sample approximately distributed according to $\bar{\pi}$ (Fearnhead, 2004). However, if we were to use this sample within Algorithm 1, the resulting invariant distribution would not be π. For this reason, we consider Particle MCMC (PMCMC) methods (Andrieu et al., 2009, 2010).

PMCMC methods allow us to use SMC ideas in a principled way within MCMC schemes. We will focus here on the PG sampler and show how one can use this methodology to obtain an efficient MCMC kernel K targeting the distribution $\bar{\pi}$ given in Equation (5). The outcome of the PG sampling steps will be either to cluster all the points in $\bar{s}$ into one block or to break $\bar{s}$ into two clusters, with the restriction that each of the two blocks should contain one anchor. Interestingly, the form of the PG algorithm is the same no matter if the two anchors were previously together or apart before its execution. This contrasts with previous split-merge algorithms such as Jain and Neal (2004), which require a different treatment for split and merge moves.

To sample from $\bar{\pi}$, PG breaks the sampling of $\bar{c}'$ into a sequence of $n \coloneqq |\bar{s}|$ simpler sampling problems. In this scenario, contrary to most applications of PG, there is no intrinsic time ordering of the observations. We randomize the order in which the points are included by introducing, conditionally on s and $\bar{s}$, a random permutation $\boldsymbol{\sigma} \coloneqq (\sigma_1, \ldots, \sigma_n)$. This permutation is sampled using Algorithm 2.

Algorithm 2

1: **function** SAMPLEPERMUTATION$(s, \bar{s})$
2:     $\sigma_1 \sim \text{Uniform}(s)$
3:     $\sigma_2 \leftarrow s \backslash \{\sigma_1\}$
4:     $(\sigma_3, \sigma_4, \ldots, \sigma_n) \leftarrow \text{UniformPermutation}(\bar{s} \backslash s)$
5:     **return** $\boldsymbol{\sigma}$
6: **end function**

In other words, $\boldsymbol{\sigma}$ is uniform over the permutations of the observation indices in $\bar{s}$ such that the members of s appear in the first two entries. The variable σ_t specifies the index of the observation y_{σ_t} introduced into the PG algorithm at SMC iteration ("algorithmic" time) t and x_t is the corresponding allocation decision. A particle $\mathbf{x}_t$ is defined as a sequence of allocation decisions, $\mathbf{x}_t \coloneqq (x_1, \ldots, x_t)$, where $x_t \in \mathcal{X}$, $t \in \{1, \ldots, n\}$.

Given $\boldsymbol{\sigma}$, we denote the SMC proposals used within PG by $q_t^{\boldsymbol{\sigma}}(x_t|\mathbf{x}_{t-1})$, and the intermediate unnormalized target distributions, by $\gamma_t^{\boldsymbol{\sigma}}(\mathbf{x}_t)$. We remind the reader that both $\gamma_t^{\boldsymbol{\sigma}}$ and $q_t^{\boldsymbol{\sigma}}$ are allowed to depend on arbitrary subsets of the observations $\mathbf{y}$; see, e.g., (Del Moral et al., 2006). However, we omit this dependency for notational simplicity. Our methodology is flexible with respect to the choice of the proposals and the choice of the intermediate

unnormalized target distributions. For our methodology to provide consistent estimates, only the following weak assumptions have to be satisfied.

Assumption 1 *For all $t \in \{1, \ldots, n\}$, we assume* $\mathrm{support}(\gamma_t^{\boldsymbol{\sigma}}) \subseteq \mathrm{support}(q_t^{\boldsymbol{\sigma}})$ *where* $q_t^{\boldsymbol{\sigma}}(\mathbf{x}_t) \coloneqq q_1^{\boldsymbol{\sigma}}(x_1) \prod_{k=2}^{t} q_k^{\boldsymbol{\sigma}}(x_k | \mathbf{x}_{k-1})$ *for $t \geq 2$.*

Assumption 2 *We assume that there exists a bijection $\phi^{\boldsymbol{\sigma}}$ taking a particle as input, and outputting a clustering of $\overline{s}$. More precisely, $\phi^{\boldsymbol{\sigma}}$ is a bijection between the support of the proposal, and the support of the split-merge target distribution, $\phi^{\boldsymbol{\sigma}} : \mathrm{support}(q_n^{\boldsymbol{\sigma}}) \to \mathrm{support}(\overline{\pi})$.*

Assumption 3 *We assume that $\gamma_n^{\boldsymbol{\sigma}}(\mathbf{x}_n) \propto \overline{\pi}(\phi^{\boldsymbol{\sigma}}(\mathbf{x}_n))$.*

Assumption 1 ensures that all the importance weights appearing in the SMC method are well-defined. Assumption 2 is a simple condition ensuring that we can consistently relabel the particles. Assumption 3 ensures that we target the desired distribution at algorithmic time n. Note that Assumption 3 only restricts the choice of γ_t for the final SMC iteration, $t = n$. We use this flexibility in Section 3.4. We show in the next section how to design $q^{\boldsymbol{\sigma}}$, $\gamma^{\boldsymbol{\sigma}}$, $\phi^{\boldsymbol{\sigma}}$ and $\mathcal{X}$ that satisfy these assumptions.

PG proceeds in a way similar to standard SMC algorithms, with the important difference that one of the N particle paths is fixed. In our setup, this path is obtained using the inverse of the bijection described in Assumption 2, applied to the state of the restricted clustering $\overline{c}$ prior to the current PG step. As discussed in Chopin and Singh (2015), we can without loss of generality set the genealogy of the conditioning path $\overline{c}$ to $(1, ..., 1)$, i.e. we use the particle index $p = 1$ for this conditioning path: $\mathbf{x}_n^1 \coloneqq (\phi^{\boldsymbol{\sigma}})^{-1}(\overline{c})$. This defines a path by taking a prefix of length t of the vector $\mathbf{x}_n^1$ for $\mathbf{x}_t^1$, i.e. $\mathbf{x}_t^1 = (\mathbf{x}_n^1)_{1:t}$.

The final ingredient required to describe the PG algorithm is a *conditional resampling* distribution $r(\boldsymbol{a} \mid \boldsymbol{w})$, where $\boldsymbol{a} \coloneqq (a_2, \ldots, a_N)$ denotes the resampling ancestors, $a_p \in \{1, \ldots, N\}$, and $\boldsymbol{w} \coloneqq (w^1, \ldots, w^N)$ denotes a vector of probabilities. We limit ourselves to multinomial resampling:

$$r(\boldsymbol{a} \mid \boldsymbol{w}) = \prod_{p=2}^{N} r(a_p \mid \boldsymbol{w}) = \prod_{p=2}^{N} w^{a_p}. \tag{6}$$

More elaborate schemes can be used, see Andrieu et al. (2009, 2010). Instead of resampling at each time step as in vanilla SMC algorithms, we only resample when the relative Effective Sampling Size (ESS) criterion, which takes values between 0 and 1, is below a pre-specified threshold β, $\beta \in [0, 1]$. The adaptive resampling procedure was proposed by Liu and Chen (1995) for standard particle methods and the correctness of this procedure for PG has been established in Lee (2011). The resulting procedure is described in Algorithm 3.

Most of Algorithm 3 is concerned with the creation of temporary auxiliary variables (lines 1–34). These auxiliary variables can all be discarded after the algorithm returns $\overline{c}'$, as they can be resampled from scratch every time Algorithm 3 is run. The part of the algorithm that performs the actual split or merge is in lines 35 and 36. At this point of the execution of the algorithm, the particle population at SMC generation n can be interpreted (via $\phi^{\boldsymbol{\sigma}}$) as a distribution over clusterings of $\overline{s}$, with some particles corresponding to merging

---

Algorithm 3

1: **function** PARTICLEGIBBSSPLITMERGE$(s, \overline{s}, \overline{c}, \overline{\pi})$ ▷ Inputs coming from Algorithm 1
2:     $\boldsymbol{\sigma} \leftarrow$ SAMPLEPERMUTATION$(s, \overline{s})$
3:     $\mathbf{x}_n^1 \leftarrow (\phi^{\boldsymbol{\sigma}})^{-1}(\overline{c})$ ▷ Compute the conditional path
4:     **for** $t \in \{1, \ldots, n-1\}$ **do**
5:         $\mathbf{x}_t^1 \leftarrow \left(\mathbf{x}_n^1\right)_{1:t}$ ▷ First particle of each generation matches the conditional path
6:     **end for**
7:     **for** $p \in \{2, \ldots, N\}$ **do**   ▷ Initialize particles
8:         $x_1^p \sim q_1^{\boldsymbol{\sigma}}(\cdot)$
9:         $\mathbf{x}_1^p \leftarrow (x_1^p)$
10:     **end for**
11:     **for** $p \in \{1, \ldots, N\}$ **do**   ▷ Initialize incremental importance weights
12:         $\tilde{w}_1^p \leftarrow \frac{\gamma_1^{\boldsymbol{\sigma}}(\mathbf{x}_1^p)}{q_1^{\boldsymbol{\sigma}}(\mathbf{x}_1^p)}$
13:     **end for**
14:     **for** $p \in \{1, \ldots, N\}$ **do**
15:         $w_1^p \leftarrow \frac{\tilde{w}_1^p}{\left(\sum_{p'=1}^N \tilde{w}_1^{p'}\right)}$ ▷ Compute normalized weights
16:     **end for**
17:     **for** $t \in \{2, \ldots, n\}$ **do**
18:         **if** $\left(N \sum_{p=1}^N (w_{t-1}^p)^2\right)^{-1} < \beta$ **then**   ▷ Resample only if relative ESS is too low
19:             $\boldsymbol{a} \sim r(\cdot \mid \boldsymbol{w}_{t-1})$ ▷ Perform the conditional resampling step
20:             $\tilde{\boldsymbol{w}}_{t-1} \leftarrow (1, 1, 1, \ldots, 1)$ ▷ Reset the weights
21:         **else**
22:             $\boldsymbol{a} \leftarrow (1, 2, 3, \ldots, N)$ ▷ Resampling is skipped: set $\boldsymbol{a}$ to the identity map
23:         **end if**
24:         **for** $p \in \{2, \ldots, N\}$ **do**
25:             $x_t^p \sim q_t^{\boldsymbol{\sigma}}\left(\cdot \mid \mathbf{x}_{t-1}^{a_p}\right)$ ▷ Propose new block allocation for y_{σ_t}
26:             $\mathbf{x}_t^p \leftarrow (\mathbf{x}_{t-1}^{a_p}, x_t^p)$ ▷ Concatenate new block allocation to path
27:         **end for**
28:         **for** $p \in \{1, \ldots, N\}$ **do**
29:             $\tilde{w}_t^p \leftarrow \tilde{w}_{t-1}^p \cdot w(\mathbf{x}_{t-1}^{a_p}, x_t^p)$ ▷ Update weights (see Equation (9))
30:         **end for**
31:         **for** $p \in \{1, \ldots, N\}$ **do**
32:             $w_t^p \leftarrow \frac{\tilde{w}_t^p}{\left(\sum_{p'=1}^N \tilde{w}_t^{p'}\right)}$ ▷ Compute normalized weights
33:         **end for**
34:     **end for**
35:     $\mathbf{x}_n' \sim \sum_{p=1}^N w_n^p \delta_{\mathbf{x}_n^p}(\cdot)$ ▷ Sample particle representing new state
36:     $\overline{c}' \leftarrow \phi^{\boldsymbol{\sigma}}(\mathbf{x}_n')$ ▷ Compute updated partition
37:     **return** $\overline{c}'$
38: **end function**

---

all points in $\overline{s}$ into one block (i.e. when $|\phi^{\boldsymbol{\sigma}}(\mathbf{x}_n')| = 1$), and others, to various way of splitting $\overline{s}$ into two blocks (when $|\phi^{\boldsymbol{\sigma}}(\mathbf{x}_n')| = 2$).

$$
\begin{aligned}
\mathcal{S}(\#1) &:= \{\#2, \#4\} \\
\mathcal{S}(\#2) &:= \{\#2\} \\
\mathcal{S}(\#3) &:= \{\#3, \#4\} \\
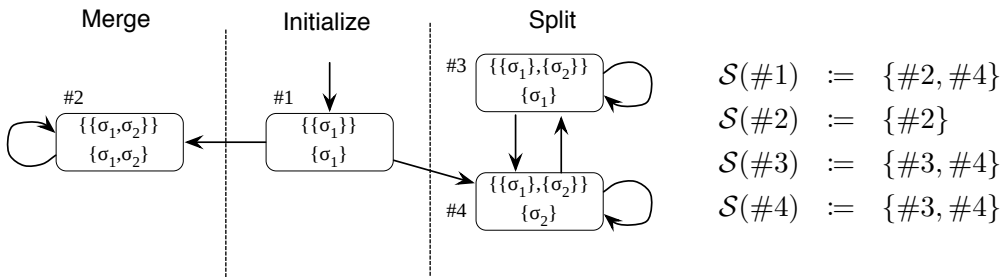\mathcal{S}(\#4) &:= \{\#3, \#4\}
\end{aligned}
$$

Figure 2: Left: State space $\mathcal{X}$ and allowed transitions $\mathcal{S}(\cdot)$ for the local allocation decisions. Right: allowed transitions between the states.

Correctness of this procedure follows straightforwardly from the original PG argument (see Appendix B for details):

Proposition 2 *Under Assumptions 1, 2, and 3, the output of Algorithm 3, $\bar{c}'$, satisfies $\bar{c}' \sim \bar{\pi}$ if $\bar{c} \sim \bar{\pi}$, for any $N \geq 2$, i.e. the Markov kernel $\overline{K}(\bar{c}'|\bar{c})$ induced by Algorithm 3 is $\bar{\pi}$-invariant.*

3.3 Intermediate target distributions and proposals construction

We detail here the construction of a set of proposal distributions $q^{\boldsymbol{\sigma}}$, unnormalized target distributions $\gamma^{\boldsymbol{\sigma}}$, and mappings $\phi^{\boldsymbol{\sigma}}$ satisfying Assumptions 1 to 3. We denote the space of possible allocation decisions at a given PG iteration by $\mathcal{X}$. Our construction is based on an encoding where the space $\mathcal{X}$ consists in the rectangles shown in Figure 2. We call the rectangles *states* for short. These states are used to build particles: recall that a particle $\mathbf{x}_t$ is defined as a list of local decisions, $\mathbf{x}_t := (x_1, \ldots, x_t)$, $x_{t'} \in \mathcal{X}$.

The state appended to a particle at time t represents (a) the clustering restricted to the anchors (shown in the first line of each rectangle in Figure 2), and (b), the cluster joined by y_{σ_t} (encoded by the anchor(s) contained in the joined cluster, second line in the same figure). As shown in Figure 2, the "merge state" (left) is an absorbing state, encoding the fact that following this local decision, all children particles are forced to join the unique block in the restricted clustering. The two "split states" (right), on the other hand, both have two outgoing transitions, encoding the fact that for each index in $\bar{s} \backslash s$, the corresponding observation needs to be allocated to one of the two blocks.

There is a bijection between the support of $\bar{\pi}$, and particles respecting the transition constraints defined by the arrows in Figure 2. More precisely, for each state $x \in \mathcal{X}$, we let $\mathcal{S}(x)$ denote the set of allowed transitions from x. We write $\mathbf{x}_t \in \mathcal{S}_t$ if (a) $x_1 = \#1$, and (b) for all $t' \in \{2, \ldots, t\}$, $x_{t'} \in \mathcal{S}(x_{t'-1})$. From this definition, we obtain the following result whose proof is given in Appendix C.

Proposition 3 *For any permutation $\boldsymbol{\sigma}$ satisfying $\{\sigma_1, \sigma_2\} = s$, there is a bijective map $\phi^{\boldsymbol{\sigma}}$ from the space of particles respecting the transition constraints, $\mathcal{S}_n$, to the support of the restricted target,* $\mathrm{support}(\bar{\pi})$.

We use this bijection to define a sequence of intermediate target and proposal distributions. The intermediate target at time t of support $\mathcal{S}_t$ is given by:

$$\gamma_t^{\boldsymbol{\sigma}}(\mathbf{x}_t) := \bar{\tau}_1(\bar{c}_t) \left(\prod_{b \in \bar{c}_t} \tau_2(|b|) L(\mathbf{y}_b) \right), \tag{7}$$

where $\bar{c}_t = \phi^{\boldsymbol{\sigma}_{1:t}}(\mathbf{x}_t)$. By construction, we have that for $t = n$, $\gamma_n^{\boldsymbol{\sigma}}(\mathbf{x}_n) \propto \bar{\pi}(\phi^{\boldsymbol{\sigma}}(\mathbf{x}_n))$ so Assumption 3 is satisfied.

We define as proposals:

$$
\begin{aligned}
q_1^{\boldsymbol{\sigma}}(x_1) &:= \delta_{\#1}(x_1), & (8) \\
q_t^{\boldsymbol{\sigma}}(x_t \mid \mathbf{x}_{t-1}) &:= \frac{\gamma_t^{\boldsymbol{\sigma}}(\mathbf{x}_t)}{\sum_{x_t' \in \mathcal{S}(x_{t-1})} \gamma_t^{\boldsymbol{\sigma}}(\mathbf{x}_{t-1}, x_t')},
\end{aligned}
$$

where $(\mathbf{x}_{t-1}, x_t')$ denotes the concatenation of x_t' to the vector $\mathbf{x}_{t-1}$, and $\mathbf{x}_t = (\mathbf{x}_{t-1}, x_t)$. These definitions satisfy Assumption 1, and yield the following weight updates:

$$
\begin{aligned}
w_t(\mathbf{x}_{t-1}, x_t) &:= \frac{\gamma_t^{\boldsymbol{\sigma}}(\mathbf{x}_t)}{\gamma_{t-1}^{\boldsymbol{\sigma}}(\mathbf{x}_{t-1})} \frac{1}{q_t^{\boldsymbol{\sigma}}(x_t \mid \mathbf{x}_{t-1})} & (9) \\
&= \frac{\sum_{x_t' \in \mathcal{S}(x_{t-1})} \gamma_t^{\boldsymbol{\sigma}}(\mathbf{x}_{t-1}, x_t')}{\gamma_{t-1}^{\boldsymbol{\sigma}}(\mathbf{x}_{t-1})} \\
&= \sum_{x_t' \in \mathcal{S}(x_{t-1})} \frac{\gamma_t^{\boldsymbol{\sigma}}(\mathbf{x}_{t-1}, x_t')}{\gamma_{t-1}^{\boldsymbol{\sigma}}(\mathbf{x}_{t-1})}.
\end{aligned}
$$

If $t > |s|$ then Equation (10) simplifies as follows

$$\frac{\gamma_t^{\boldsymbol{\sigma}}(\mathbf{x}_t)}{\gamma_{t-1}^{\boldsymbol{\sigma}}(\mathbf{x}_{t-1})} = \frac{\tau_2(|b_t^+|)}{\tau_2(|b_t^-|)} L\left(\mathbf{y}_{b_t^+} \mid \mathbf{y}_{b_t^-}\right), \tag{10}$$

where

$$L\left(\mathbf{y}_{b_t^+} \mid \mathbf{y}_{b_t^-}\right) := \frac{L\left(\mathbf{y}_{b_t^+}\right)}{L\left(\mathbf{y}_{b_t^-}\right)}. \tag{11}$$

Here b_t^- and b_t^+ encode the block in which a point is added to when transitioning from $\mathbf{x}_{t-1}$ to $\mathbf{x}_t$, the first being the block before the addition, and the second, the same block after the addition:

$$b_t^- := \bar{c}_{t-1} \backslash \bar{c}_t, \quad b_t^+ := b^- \cup \{\sigma_t\}. \tag{12}$$

Depending on the form of the partition prior and likelihood it may be possible to simplify these quantities into more computationally efficient forms.

3.4 An improved sequence of intermediate target distributions

We now describe an improvement over the basic intermediate and proposal distributions presented in the previous section. This improvement addresses a "greediness" problem of the (conditional) SMC procedure. Consider a case where the ratio $\bar{\tau}_1(1)/\bar{\tau}_1(2)$ between a merge and a split is large. This can occur for example when the Dirichlet process concentration parameter α_0 is small. In this case, the proposal in the first non-trivial step, $q_2^{\boldsymbol{\sigma}}$, will assign most of its mass to the transition from state #1 to state #2 (see Figure 2). However, the likelihood might overcome this prior when $|\bar{s}|$ is large. But proposing such split has low probability under the definitions given in the previous section, as #2 is an absorbing state.

To overcome this issue, we build a new sequence of intermediate distributions, which delay the incorporation of the prior:

$$\widehat{\gamma_t^{\boldsymbol{\sigma}}}(\mathbf{x}_t) := \begin{cases} \mathbf{1}[\mathbf{x}_t \in \mathcal{S}_t], & \text{if } t \in \{1,2\}, \\ (\gamma_2^{\boldsymbol{\sigma}}(\mathbf{x}_{1:2}))^{\zeta_t} \frac{\gamma_t^{\boldsymbol{\sigma}}(\mathbf{x}_t)}{\gamma_2^{\boldsymbol{\sigma}}(\mathbf{x}_{1:2})}, & \text{otherwise.} \end{cases} \tag{13}$$

where ζ_t is a positive increasing annealing schedule such that $\zeta_n = 1$. We use the following proposal based on these new intermediate distributions:

$$\widehat{q_1^{\boldsymbol{\sigma}}}(x_1) \quad := \quad \delta_{\#1}(x_1), \tag{14}$$

$$\widehat{q_t^{\boldsymbol{\sigma}}}(x_t \mid \mathbf{x}_{t-1}) \quad := \quad \frac{\widehat{\gamma_t^{\boldsymbol{\sigma}}}(\mathbf{x}_t)}{\sum_{x_t' \in \mathcal{S}(x_{t-1})} \widehat{\gamma_t^{\boldsymbol{\sigma}}}(\mathbf{x}_{t-1}, x_{t-1}')}.$$

This yields the weight updates:

$$\widehat{w_t}(\mathbf{x}_{t-1}, x_t) \quad := \quad \sum_{x_t' \in \mathcal{S}(x_{t-1})} \frac{\widehat{\gamma_t^{\boldsymbol{\sigma}}}(\mathbf{x}_{t-1}, x_t')}{\widehat{\gamma_{t-1}^{\boldsymbol{\sigma}}}(\mathbf{x}_{t-1})}. \tag{15}$$

For simplicity, we pick $\zeta_t = \frac{t-2}{n-2}$. This choice simplifies ratios of intermediate distributions to:

$$\frac{\widehat{\gamma_t^{\boldsymbol{\sigma}}}(\mathbf{x}_t)}{\widehat{\gamma_{t-1}^{\boldsymbol{\sigma}}}(\mathbf{x}_{t-1})} \quad = \quad \begin{cases} 1 & \text{if } t = 2, \\ (\gamma_2^{\boldsymbol{\sigma}}(\mathbf{x}_{1:2}))^{\Delta\zeta} \frac{\gamma_t^{\boldsymbol{\sigma}}(\mathbf{x}_t)}{\gamma_{t-1}^{\boldsymbol{\sigma}}(\mathbf{x}_{t-1})} & \text{if } t > 2, \end{cases} \tag{16}$$

where $\Delta\zeta := (n-2)^{-1}$.

3.5 Runtime analysis

To simplify the analysis of the running time, we make a few assumptions.

Assumption 4 *The parametric likelihood model has the following properties:*

1. *let $\psi := \psi(\mathbf{y}_b)$ denote a sufficient statistic, and define, with a slight abuse of notation, $L(\psi) := L(\mathbf{y}_b)$. For a given sufficient statistic value ψ, the likelihood $L(\psi)$ can be computed in time $O(l)$,*

2. *the sufficient statistic for b^+, $\psi^+ := \psi(\mathbf{y}_{b^+})$, can be updated in time $O(u)$ from the sufficient statistic for b^-, $\psi^- := \psi(\mathbf{y}_{b^-})$.*

The next assumption holds for all the clustering priors reviewed in Section 2.

Assumption 5 *The ratio $\frac{\tau_2(j+1)}{\tau_2(j)}$ can be computed in constant time.*

For example, with a Dirichlet process, this ratio is equal to $j!/(j-1)! = j$. Since $|b^+| = |b^-| + 1$, Assumption 5 implies that the ratio $\frac{\tau_2(|b^+|)}{\tau_2(|b^-|)}$ in Equation (10) can be computed in constant time.

Proposition 4 *Under Assumptions 4 and 5, one weight computation, Equation (15), takes time $O(u+l)$. The storage cost per particle is $O(1)$. Moreover, the running time per weight computation is independent of the number of clusters.*

The running time result follows directly from the fact that $|\mathcal{S}(\cdot)| \leq 2$, and hence the sum in Equation (15) has a constant number of terms. The constant storage cost follows from the finite dimensionality of the sufficient statistics (see Assumption 4), and from $|\mathcal{X}| = 4$.

We also remind the reader that for most resampling schemes, including the one in Equation (6), the computational cost as a function of the number of particles and SMC iterations is $O(Nn) = O(N|\bar{s}|)$ (Doucet and Johansen, 2011).

3.6 Generalization

For simplicity, we have assumed so far that $|s| = 2$, and hence, according to the auxiliary variable analysis of Appendix B, $|\bar{c}| \leq 2$. In fact, the same auxiliary variables with more than two anchor points can be used to construct novel sampling algorithms. Details are given in Appendix D.

This generalization loses some interpretability compared to the split-merge case ($|s| = 2$), but can be useful in finite clustering models. In this case, it may only be possible to split a cluster if a merge is performed simultaneously. For this reason, we use $|s| = 3$ in the finite Dirichlet mixture model examples in Section 4.3. For the Dirichlet Process, we did not observe notable improvements by going from $|s| = 2$ to $|s| = 3$, so we use the former setting for the non-parametric models.

4. Applications

In this section, we demonstrate the performance of our methodology and compare it to standard alternatives. We use a series of synthetic datasets covering a large spectrum of cluster separateness, as well as real data coming from a geolocation application.

4.1 Implementation and evaluation

We have implemented the following three Dirichlet Process (DP) clustering samplers in the same Python codebase: the PGSM method described in this work, the efficient Sequentially-Allocated Merge Split (SAMS) method of Dahl (2003), as well as the standard Gibbs sampler. The code and instructions allowing to reproduce the experiments is available at `https://github.com/aroth85/pgsm`. The implementation of the likelihood computations are the same for all samplers, thus the running times are comparable. We have tested

the correctness of our computer implementations by computing the true posterior distribution on small examples via combinatorial enumeration, and verified that the Monte Carlo estimates converged to this distribution for all three methods.

Unless we state otherwise, we initialized the samplers with the single-cluster configuration. In datasets much smaller than those studied in this work, initializing the Gibbs sampler to the fully disconnected clustering is advantageous (Sudderth, 2006), but in larger datasets, the quadratic burn-in cost involved with this initialization is not scalable. However, we verified that after a long burn-in period the Gibbs method initialized to the fully disconnected clustering eventually reaches the same likelihood values in the synthetic examples. We also investigate the high cost of the fully disconnected initialization in the results shown in Figure 9.

To evaluate the performance of the samplers, we held-out a random but fixed 10% of each dataset. We collected samples and computed the predictive likelihood and V-measure (Rosenberg and Hirschberg, 2007) every 100 iterations. All experiments are replicated 10 times, and smoothed using a moving average with a window size of 20 for plotting.

4.2 Likelihoods and priors

In six of the synthetic experiments and the geolocation experiments discussed further in Section 4.3 and Section 4.4, we used a Normal-Inverse-Wishart conjugate likelihood model. In two of the synthetic experiments in Section 4.3 we used Bernoulli mixture models with 50 dimensions. Each dimension is an independent draw from a Bernoulli random variable with cluster specific parameters. We set a proportion of the dimensions to be uninformative as follows. Values for uninformative dimensions were drawn from Bernoulli variables with parameter 0.5 regardless of the cluster membership. Values for the remaining dimensions were drawn from cluster specific Bernoulli variables with parameters sampled from the Uniform distribution. For the cancer data discussed in Section 4.5, we use the application-specific PyClone likelihood model (Roth et al., 2014). The PyClone model uses genomic sequence data from tumours to identify mutations which co-occur in cells and estimates the proportion of cells harbouring the mutations. The model is not conjugate, so we apply a discretization that allows us to treat the model as conjugate. Complete details for each model are provided in Appendix E.

We use a DP prior with base measure given by the conjugate prior of the corresponding likelihood model in all experiments. We use a Gamma$(1, 0.1)$ prior and resample the value of the concentration parameters α_0 using a standard auxiliary variable method (Escobar and West, 1995). The value of α_0 is initialized to 1.0.

4.3 Artificial datasets

We used four sources of synthetic data. First, the four datasets from Franti and Virmajoki (2006) denoted S1–4. Each of the four datasets consists in 5000 points generated from 15 bivariate Normal distributions with increasing amount of overlap between the clusters. Second, we created another synthetic dataset, which we call C1, shown in Figure 5, right. Third, we simulated two datasets with 5000 points from a Bernoulli mixture model with 50 dimensions and 16 clusters, where we set 25% (Ber-0.25) and 50% (Ber-0.5) of dimensions
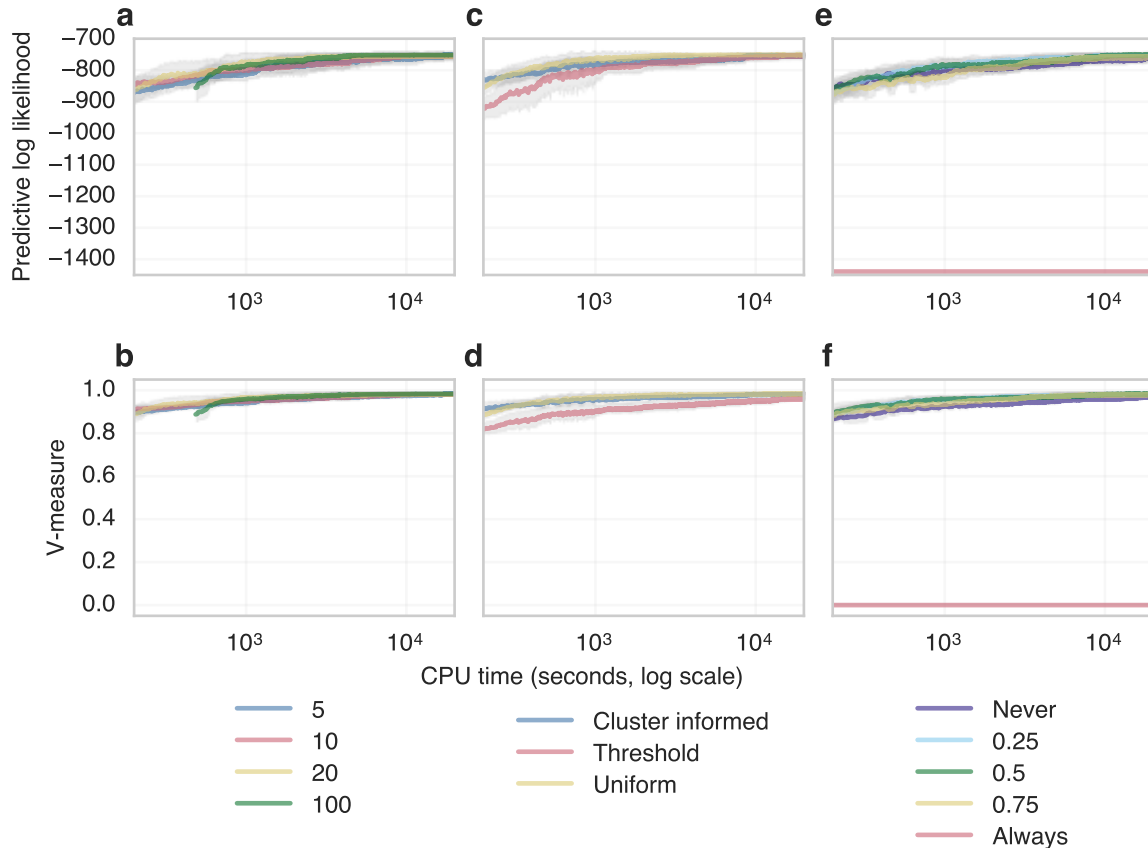
Figure 3: Effect on the predictive performance and clustering accuracy as a function of CPU time in log scale. **a**) and **b**) Varying the number of particles with the cluster informed proposal and a relative ESS resampling threshold of 0.5. **c**) and **d**) Varying the distribution for proposing pairs of anchor points, h, with 20 particles and an ESS resampling threshold of 0.5. **e**) and **f**) Varying the relative ESS resampling threshold with the cluster informed anchor proposal and 20 particles.

to be uninformative. Finally, we used 64 (DIM064) and 128 (DIM128) dimensional Normal datasets from Fränti et al. (2006) with 1024 data points and 16 clusters.

We started with a series of pilot experiments on S1 only, designed to assess the effect of various tuning parameters on the performance of PGSM. For all pilot experiments we use the unmixed PGSM sampler to isolate the effect of each tuning parameter. In practice it is usually better to alternate between one iteration of the PGSM sampler and one iteration of the Gibbs sampler. The effect of this mixing is explored later in this section.

We first explore the performance as we vary the number of particles used for each PGSM iteration (Figures 3 **a** and **b**). The curves with more particles take more time per iteration to run, however seem to achieve slightly better V-measure and predictive likelihood after
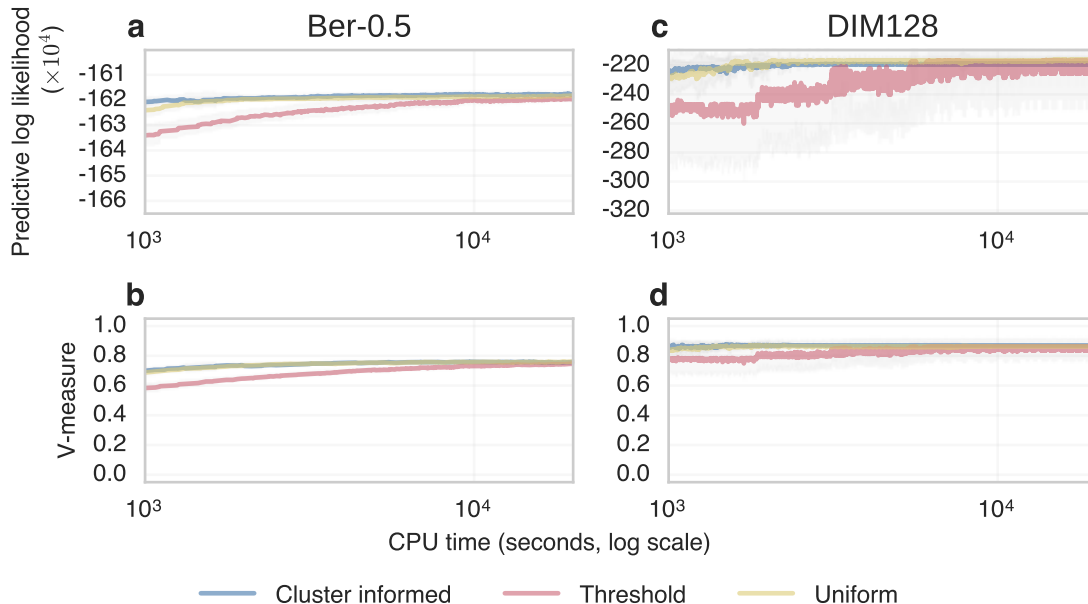
15

Figure 4: Effect on the predictive performance and clustering accuracy as a function of CPU time in log scale with different distribution h for proposing pairs of anchor points. **a**) and **b**) Comparison using a 50 dimensional Bernoulli dataset with 50% of the dimensions being uninformative. **c**) and **d**) Comparison using a 128 dimensional Normal dataset.

the initial iterations. The performance difference are negligible and the PGSM sampler generally seems insensitive to the number of particles for this dataset. For subsequent experiments we used 20 particles.

Next we compare the performance of different proposal distributions for the anchor auxiliary variables (Figures 3 **c** and **d**). For this experiment we kept the number of particles fixed at 20 and the resampling threshold at 0.5. We consider three proposal distributions.

Uniform: Sample the anchors uniformly at random from the $\binom{T}{2}$ possibilities.

Cluster informed: Sample the first anchor uniformly at random. Sample a cluster to draw the second anchor from with probability $\frac{1}{|c-1|}$ for the cluster containing the first anchor; otherwise proportional to

$$\frac{L(y_{\bar{b} \cup b})}{L(y_{\bar{b}})L(y_b)}$$

where $\bar{b}$ is the cluster containing the first anchor and b is the candidate cluster. Sample the second anchor uniformly from the chosen cluster.

Threshold informed: Sample the first anchor uniformly at random and the second anchor from clusters that have Chinese restaurant attachment probabilities greater than a threshold of 0.01.

16

To ensure the adaptation of the informed proposals stops, and does not perturb the invariant distribution of the sampler, we only update the proposal distributions when the number of clusters instantiated breaks the previous record. Adaptation is guaranteed to terminate in finite time given there are only a finite number of points. This would usually take a long time, so in practice it is advantageous to stop adaptation after a fixed period of time. Detailed implementations of the cluster informed and threshold informed proposals are given in Appendix F. Our results suggest that performance is not strongly affected by the anchor proposal distribution h. We only saw a small advantage when using the informed proposal distributions for the auxiliary anchor variables in s. We also explored the effect of h in higher dimensional datasets (Figure 4). Again we found the results are not sensitive to the choice of h. With the exception of the circle dataset, where we used the uniform proposal, we used the cluster informed prior for both the PGSM and SAMS samplers in subsequent experiments.

The frequency of the resampling step had a larger effect. A critical implementation point in order for the PGSM method to work is that resampling should be done adaptively by monitoring the ESS of the particle approximations in Algorithm 3 (Liu and Chen, 1995; Lee, 2011). We varied the relative ESS resampling threshold β from 0 (never resample) to 1 (always resample) (Figures 3 **e** and **f**). We observed that the performance is markedly degraded if resampling is performed after each SMC iteration, but similar for all other resampling thresholds. We used a threshold of 0.5 in all other experiments.

Next, we used the dataset C1 to investigate the effectiveness of our method with the finite clustering model introduced in Section 2, with the number of clusters fixed to $k_0 = 5$. In this case, standard split-merge methods such as SAMS are less helpful since only merging can be performed when the maximum number of clusters has been allocated. The PGSM sampler does not have this restriction and naturally allows simultaneously splitting and merging while preserving the total number of clusters. Furthermore, the PGSM sampler can use more than two anchors, potentially allowing for large changes in configuration without altering the number of clusters. We compared the PGSM with two ($|s| = 2$) and three ($|s| = 3$) anchors to the Gibbs sampler. The PGSM method outperformed the Gibbs sampler, though increasing the number of anchors did not improve the performance (Figure 5 **a**). We plot the predictive densities (Figure 5 **b**, **d**, **f**) and cluster allocations (Figure 5 **c**, **e**, **g**) after running each sampler for 1000 seconds. At this point the PGSM sampler used a single cluster to model the points in the middle, while the Gibbs samplers used two clusters to model the central cluster.

In Figures 6 and 7 we show a series of experiments on the four datasets S1–4 describe in the previous section. We compare the PGSM to standard Gibbs and the SAMS method of Dahl (2003). We first compared pure kernels, where the split-merge samplers are not mixed with standard Gibbs moves (Figure 6). The pure PGSM kernel outperformed both Gibbs and SAMS on datasets S1 and S2. The Gibbs kernel and PGSM perform similarly for datasets S3 and S4, and both outperformed SAMS. When the split-merge moves are mixed with standard Gibbs moves, the split-merge methods outperformed Gibbs on datasets S1 and S2, with all methods showing similar performance on datasets S3 and S4 (Figure 7).

Finally, we explored the performance of the methods on four high dimensional datasets. The mixed PGSM and Gibbs samplers performed the best on the Bernoulli datasets, while the unmixed PGSM sampler is slower to reach the same predictive likelihood and V-measure
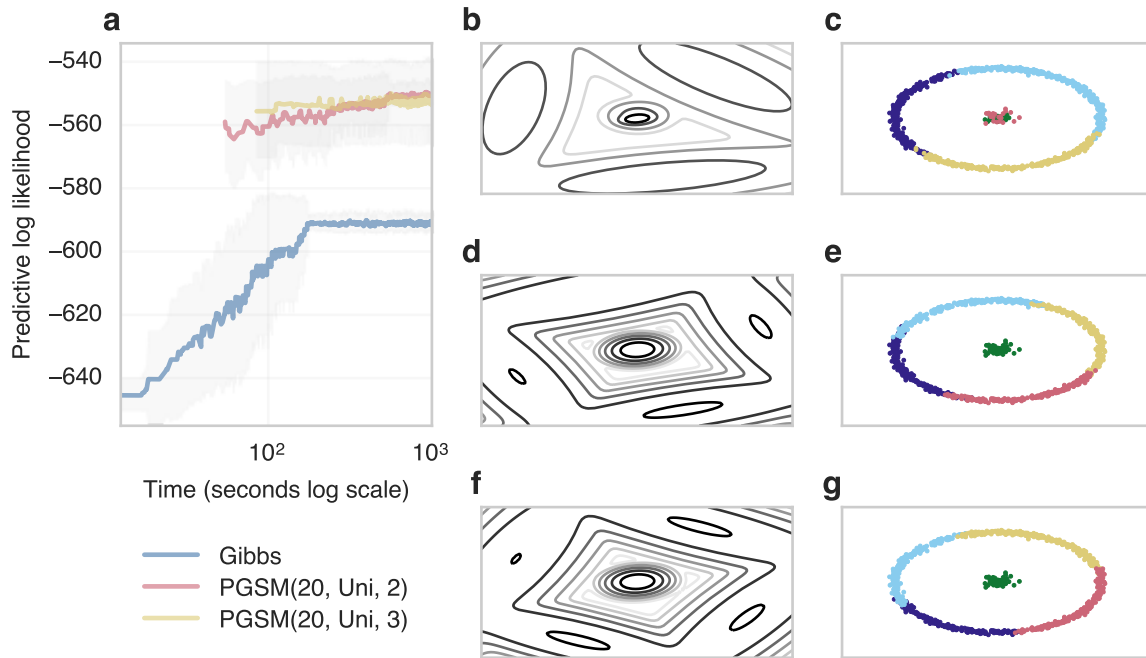
Figure 5: Comparison of Gibbs and PGSM for finite Dirichlet prior with $k_0 = 5$. **a**) Predictive log likelihood comparison of Gibbs and PGSM using two (PGSM(20, Uni, 2)) or three (PGSM(20, Uni, 3)) anchors. Predictive density after 1000 seconds for **b**) Gibbs; **d**) PGSM(20, Uni, 2); **f**) PGSM(20, Uni, 3). Cluster assignment after 1000 seconds for **c**) Gibbs; **e**) PGSM(20, Uni, 2); **g**) PGSM(20, Uni, 3).
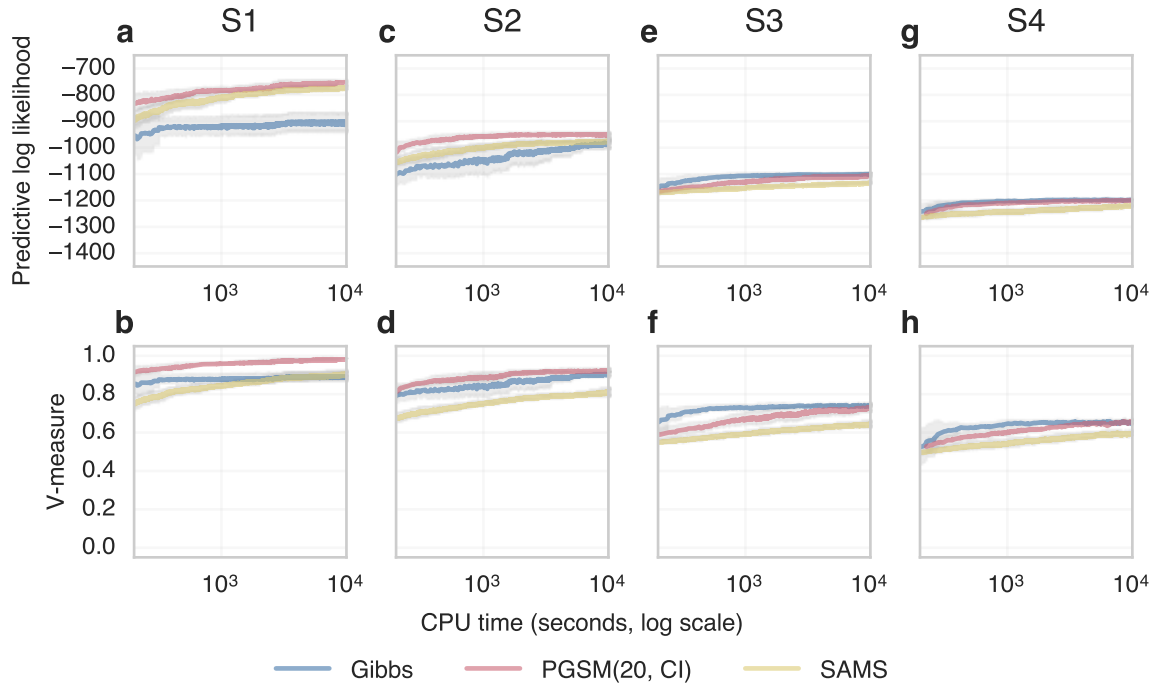
18

Figure 6: Comparison of MCMC algorithm using only a single kernel at a time (pure kernels) on 2D Normal datasets. Predictive log likelihood for datasets **a**) S1; **c**) S2; **e**) S3; **g**) S4. V-measure for datasets **b**) S1; **d**) S2; **f**) S3; **h**) S4.

(Figures 8 **a-d**). The mixed SAMS sampler failed to reach the same predictive likelihood as the PGSM and Gibbs methods, oscillating around lower values. The unmixed SAMS sampler appears to be trapped in a local mode, corresponding to poor predictive likelihood and V-measure. For the Normal datasets, the Gibbs sampler was trapped in a local mode and had markedly worse performance than other methods (Figures 8 **e-h**). The unmixed samplers outperformed the mixed equivalents on the 64 dimensional data. Furthermore, the unmixed PGSM method had a large performance advantage over all other methods on the 128 dimensional dataset.

4.4 Geolocation data

We compared the performance of the three sampling methods on a geolocation dataset. The dataset, described in more detail in Fränti et al. (2010), consists of a subset of data collected by MOPSI, a Finnish mobile application where users can post their current geographic location via their mobile device. The subset we used consists of a list of 13,467 locations (latitude-longitude pairs) from users located in Finland until 2012. The data is freely accessible from `http://cs.joensuu.fi/sipu/datasets/`.
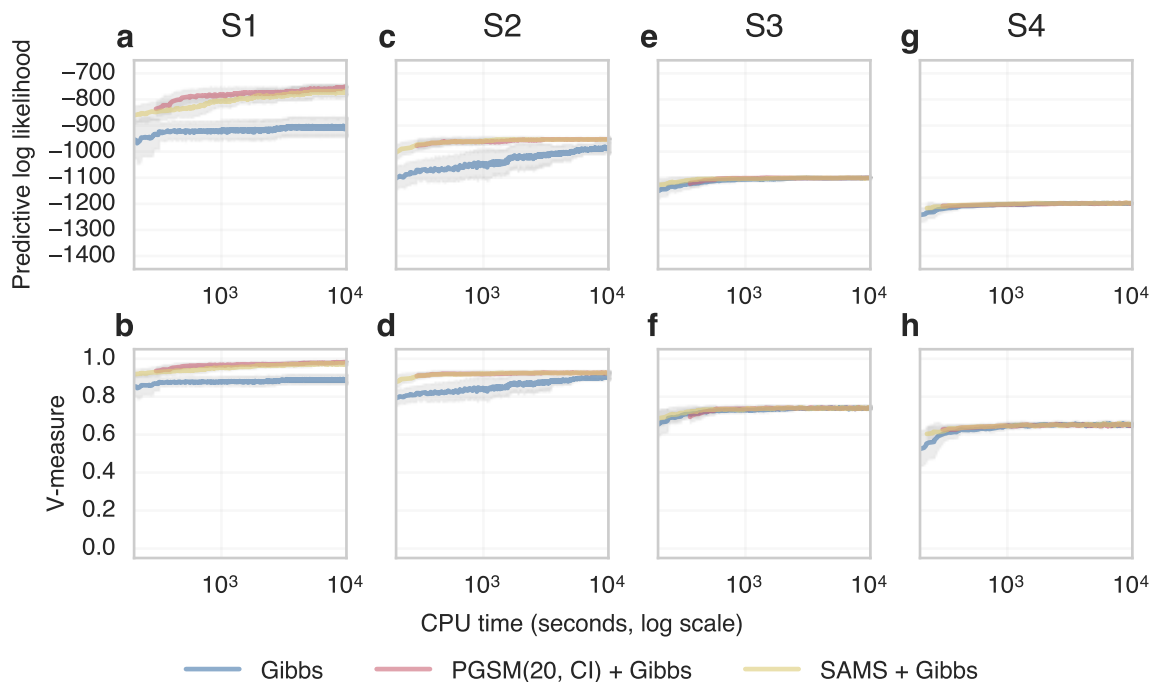
Figure 7: Comparison of MCMC algorithm using split-merge moves combined with standard Gibbs moves (mixed kernels) on 2D Normal datasets. Predictive log likelihood for datasets **a**) S1; **c**) S2; **e**) S3; **g**) S4. V-measure for datasets **b**) S1; **d**) S2; **f**) S3; **h**) S4.

We use this data as a proxy for the estimation of mobile device user density. The DP mixture of Normal-Inverse-Wishart distributions provides a natural way to obtain a parsimonious estimate of population density, where the flexibility on the shape and number of clusters can accommodate a broad range of density variability factors ranging from densely populated cities to vast low-density rural areas.

We summarize the results in Figure 9. In Figures 9 **a-c**, we display quantitative results as measured by held-out predictive likelihood performance. In Figure 9 **a**, we show that mixed PGSM, mixed SAMS and Gibbs samplers perform similarly. In Figure 9 **b**, we show that the performance of SAMS is considerably degraded if SAMS is not mixed with a Gibbs kernel. In Figure 9 **c**, we show that the performance of PGSM is less degraded if not mixed with a Gibbs kernel.

In Figures 9 **e-j**, we visualize the posterior predictive density approximated using MCMC samples. We also show the raw data in Figure 9 **d** for reference. The following three pairs of density plots are included to illustrate the high computational cost of initializing a standard Gibbs sampler at a fully disconnected configuration. From left to right: the first pair shows the predictive density after one round of Gibbs sampling initialized at the fully disconnected configuration (Figures **e** and **f**); the second pair, after sampling with PGSM
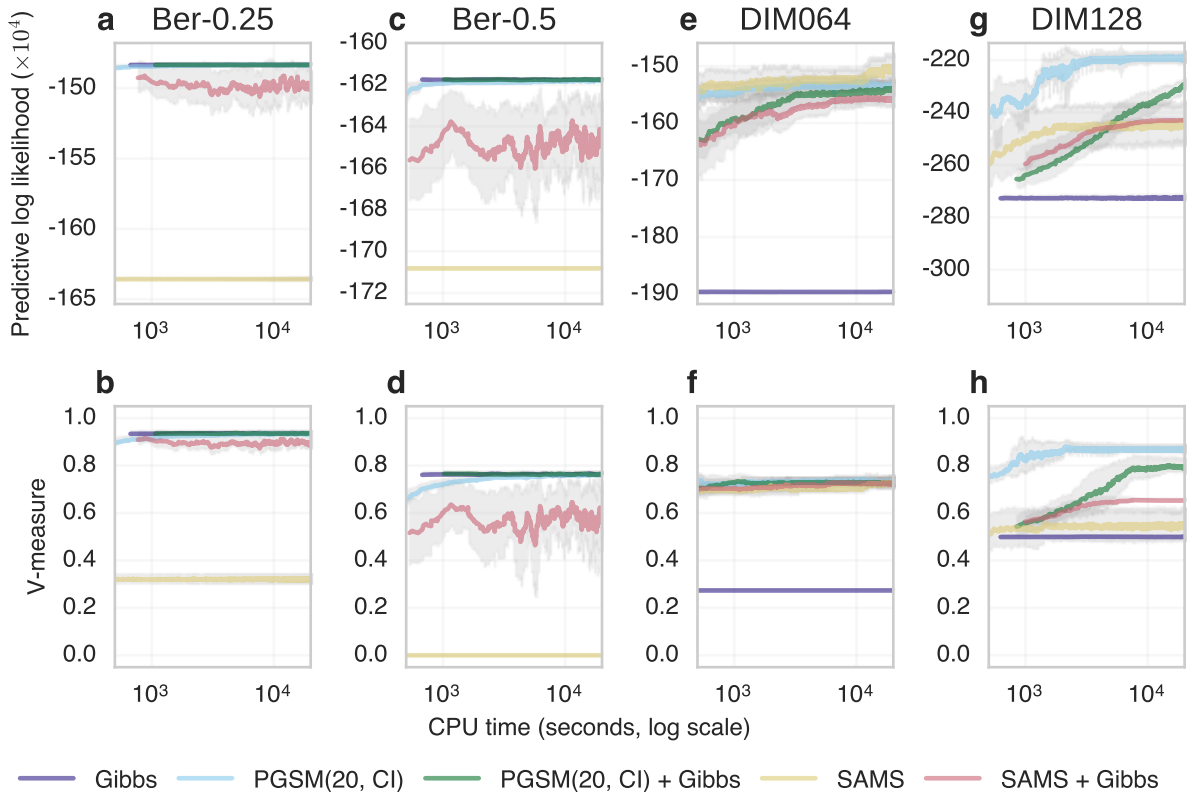
Figure 8: Comparison of MCMC algorithm using split-merge moves with combined with standard Gibbs moves (mixed kernels) on 50 dimensional Bernoulli data with 25% (Ber-0.25) and 50% (Ber-0.5) and Normal data 64 (DIM064) and 128 (DIM128) dimensions. Predictive log likelihood for datasets **a**) Ber-0.25; **c**) Ber-0.5; **e**) DIM064; **g**) DIM128. V-measure for datasets **b**) Ber-0.25; **d**) Ber-0.5; **f**) DIM064; **h**) DIM128.

initialized at the fully connected configuration for the same time (Figures **g** and **h**); the third, after running the Gibbs sampler for 10^5 seconds (Figures **i** and **j**). This demonstrates that our method can produce accurate and compact density estimates without relying on an expensive initialization phase.

4.5 Inferring population structure in heterogeneous tumours

The PyClone model (Roth et al., 2014) is designed to infer the proportion of cancer cells in a tumour sample which contain a mutation, which we refer to as the cellular prevalence of the mutation. The input data consists of a set of digital measurements of allelic abundance which is assumed to be proportional to the true abundance of the allele in the sample. The key factors which need to be deconvolved to convert this measurement to an estimate of
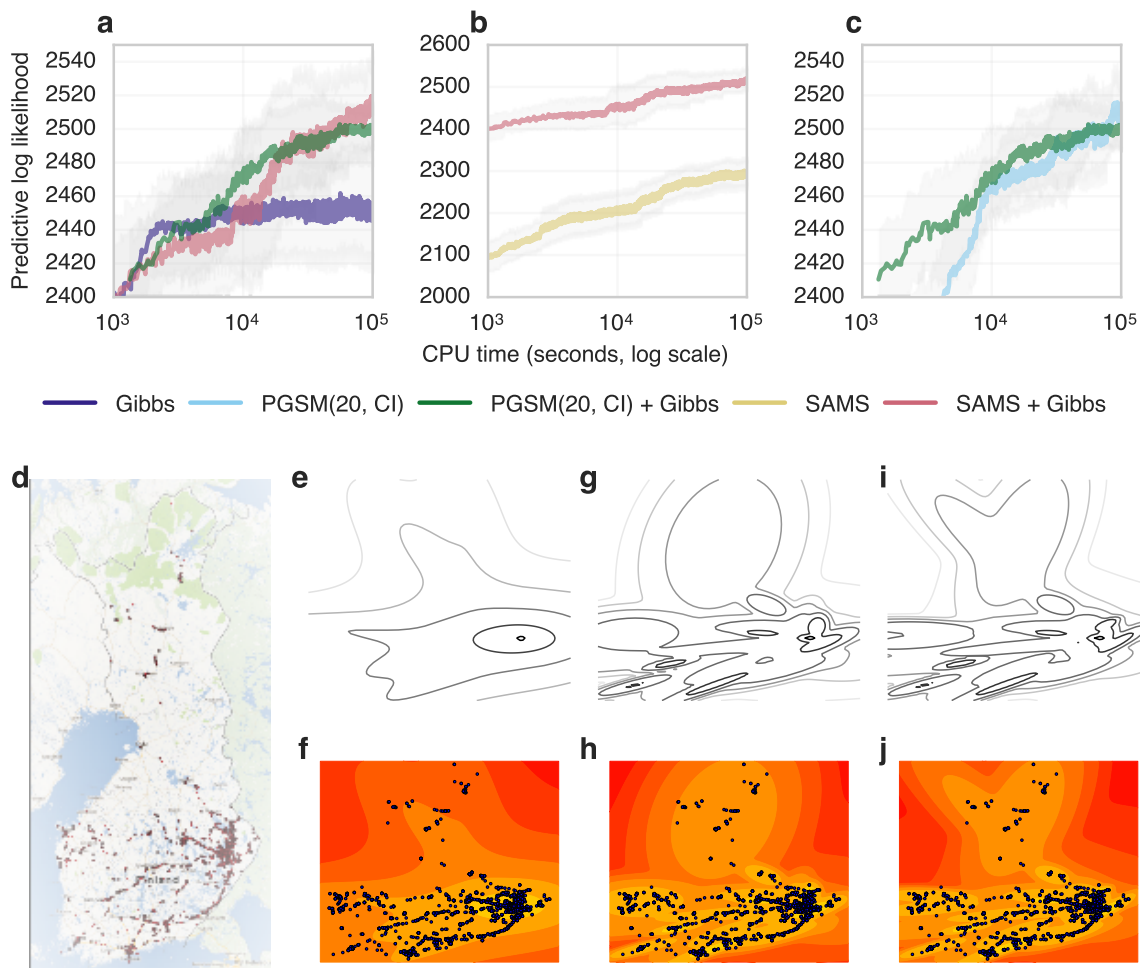
Figure 9: Geolocation dataset. Comparison of predictive likelihoods of **a**) mixed kernels; **b**) PGSM pure and mixed kernels. **c**) SAMS pure and mixed kernels. **d**) MOPSI geolocation dataset. **e**) Predictive density of Gibbs initialized from the disconnected configurations after one iteration; **d**) with point data points overlayed. **g**) and **h**) Predictive density of mixed PGSM kernel after an equivalent amount of time. **i**) and **j**) Predictive density of disconnected Gibbs after 10^5 seconds.

cellular prevalence are that some cells derive from healthy (normal) tissue and the genomes of cancer cells contain multiple copies of a locus. The model assumes that mutations will group by cellular prevalence due to the expansion of populations of genetically identical cells. The number of populations is unknown, thus the PyClone model uses a DP prior with a Uniform($[0, 1]$) base measure. The component parameters are interpreted as the cellular prevalence of the mutations associated with the component.
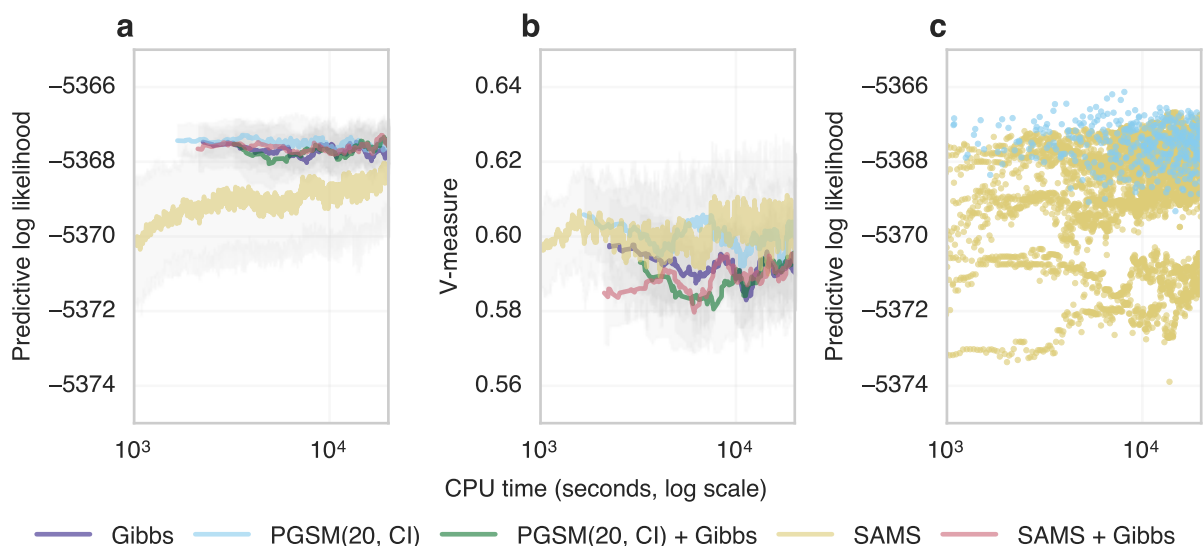
Figure 10: Clustering of cancer mutations from synthetic next-generation sequencing data. **a**) Comparison of predictive likelihood. **b**) Trace of log predictive likelihood for PGSM and SAMS pure kernels from 10 random restarts.

We show results on a dataset with 10,000 synthetic mutations in Figure 4.5. All methods except the pure SAMS kernel performed similarly in terms of predictive likelihood, while the pure SAMS kernel performed significantly worse (Figure 4.5 **a**). The pure PGSM and SAMS kernels outperformed the other methods in terms of V-measure, though the difference were small (Figures 4.5 **b**).

As observed in the other domains, the performance of SAMS critically depends on mixing the kernel with GIBBS moves. We show the data points for each replication of the pure split-merge kernels further supporting this point (Figure 4.5 **c**).

5. Discussion

We have proposed a new methodology to design efficient split-merge moves for Bayesian mixture models. The method also generalizes to new types of moves useful for finite clustering models when $|s| > 2$. We have shown empirically that the proposed method is competitive in a range of clustering and likelihood models, including synthetic and real datasets from geolocation and genomics applications.

Our method, being based on the established PMCMC framework, opens up many directions for future improvements. This includes applying recent advances in parallel implementations of SMC, for example via graphical processing units (Lee et al., 2010), or modifications of the SMC algorithm itself (Jun et al., 2012; Murray et al., 2016; Lee and Whiteley, 2016).

Another area of improvement comes from the development of resampling schemes tailored to discrete latent variables. In Algorithm 3, the number of possible distinct successors for each given particle is a small finite number (at most two if $|s| = 2$ for example). The complexity of the problem comes from the fact that a potentially long sequence of such decisions need to be made in order to split a cluster. In these specific scenarios, custom PMCMC methods based on the early work of Fearnhead and Clifford (2003) have been developed in Whiteley et al. (2010) and would provide futher improvement.

The fact that the state transitions $\mathcal{S}(\cdot)$ have an absorbing state has both advantages and disadvantages. On the one hand it may cause Algorithm 3 to be greedy, as explained in Section 3.4. We have described in the same section a choice of intermediate and proposal distributions tailored to alleviate this issue. A potential alternative consists in designing a resampling distribution r, which conditions on the survival of at least one representative of both a merge and a split. None of the existing resampling schemes have this property. On the other hand, having an absorbing state has the advantage that if all particles simulated by Algorithm 3 at some iteration t are equal to the merge absorbing state (i.e. $x_t^p = \#2$ for all particle index $p \in \{1, \ldots, N\}$), then there is no need to continue the computation of the particle filter for $t' > t$.

In standard applications of the PG algorithm, coalescence of the particle genealogy may cause slow mixing as noted in Andrieu et al. (2010). The issue is that the particles $\mathbf{x}_n^1, \mathbf{x}_n^2, \ldots, \mathbf{x}_n^N$ appearing in Algorithm 3 have components at time t for $t \ll n$ which coincide with high probability with the components of the conditioning path. This can be resolved using more sophisticated MCMC moves on the PG auxiliary variables (Whiteley, 2010; Whiteley et al., 2010; Lindsten et al., 2014). In our non-standard setup, this issue is partially mitigated by the fact that the order $\boldsymbol{\sigma}$ at which the particles are introduced is itself random. Nonetheless, it would be interesting to implement these more advanced schemes to the problem at hand.

We have shown in Section 4.5 a simple and effective method for handling models where each cluster component is governed by a non-conjugate model with a low-dimensional parameterization. We leave for future work the extension of our method to higher dimensional non-conjugate likelihood models. This problem can be approached, for example, by combining our method with the auxiliary variables described in Neal (2000).

Acknowledgements

Appendix A. Correctness of the decomposition into split-merge subproblems

We present in this section the proof of correctness of the decomposition of the clustering into split-merge sub-problems. The main tool used to prove this result is an auxiliary variable construction. The auxiliary variable consists of a pair (s, c_-), where s is the set of anchors,
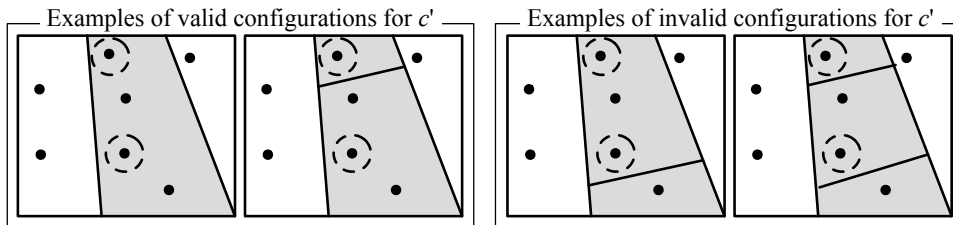
Figure 11: Assuming that the values of c, c_- and s are as in Figure 1, this illustrates some examples of configurations c' that are part of (left), or excluded from (right), the support of $c'|s, c_-$.

and c_- consists of the blocks of the partition c that do not contain anchor points:

$$c_- := \{b \in c : b \cap s = \emptyset\}, \tag{17}$$

These intuitively correspond to the blocks of the partition that are forced to stay unchanged in this split-merge step. We will view the split-merge step as a Gibbs step conditioning on c_-, s.

A slight subtlety is that conditioning on the auxiliary variables not only forces the blocks in c_- to stay constant; it also forces the other blocks to each contain at least one of the anchors. See Figure 11 for an example. This leads to condition 2 in Lemma 5.

Lemma 5 *Let c, c' denote two partitions of $[T]$. Let $s \subseteq [T]$. Define $c_- := \{b \in c : s \cap b = \emptyset\}$ and $c_-' := \{b \in c' : s \cap b = \emptyset\}$. Then $c_- = c_-'$ if and only if the following two conditions hold:*

1. $c \cap c_- = c' \cap c_-$, *and,*

2. $b \in c' \backslash c_- \implies b \cap s \neq \emptyset$.

Proof ($\implies$) Condition 1 holds trivially. For condition 2, suppose (a) $c_- = c_-'$, (b), $b \in c' \backslash c_-$, but (c) $b \cap s = \emptyset$. By (b), $b \in c'$ and $b \notin c_-$. This and (c) implies that $b \in c_-'$. But this contradicts (a), so condition 2 holds as well.

($\impliedby$) First, suppose $b \in c_-$. By condition 1, $b \in c_-'$. Therefore, $c_- \subseteq c_-'$.

Second, suppose $b \in c_-'$. By the contrapositive of condition 2, $b \notin c' \backslash c_-$. This point and $c_-' \subseteq c'$ implies that $b \in c_-$. ∎

We can now turn to the proof of Proposition 1. We copy its statement here for convenience:

Proposition 6 *If $c \sim \pi$, then the output of Algorithm 1, c', satisfies $c' \sim \pi$; i.e. the Markov kernel $K(c'|c)$ induced by Algorithm 1 is π-invariant.*

Proof Consider the model augmented with the auxiliary variables s and c_- (see Figure 12(a)), defined formally using the following auxiliary distribution:

$$\tilde{\pi}(s, c_-, c) := \pi(c)h(s)\mathbf{1}[c_- = c_-(s,c)], \tag{18}$$

where $c_-(s,c) := \{b \in c : b \cap s = \emptyset\}$. Note that this auxiliary distribution admits the target distribution as a marginal:

$$\sum_s \sum_{c_-} \tilde{\pi}(s, c_-, c) = \pi(c) \sum_s h(s) \sum_{c_-} \mathbf{1}[c_- = c_-(s,c)] \tag{19}$$

$$= \pi(c) \sum_s h(s) = \pi(c),$$

where the sum over c_- is over all sets of subsets of $[T]$, and the sum over s is over all subsets of $[T]$. We used the fact that only one c_- satisfies $c_-(s,c) = c_-$, and that h is a probability mass function.

Next, we introduce three kernels with inputs and outputs denoted by:

$$c \xoverset{K_1}{\longmapsto} (s, c_-, c) \xoverset{K_2}{\longmapsto} (s, c_-, c') \xoverset{K_3}{\longmapsto} c'. \tag{20}$$

These kernels play the following roles:

- K_1 samples the auxiliary variables according to $\tilde{\pi}(s, c_- \mid c)$, while keeping c fixed,

- K_2 performs a Metropolis-within-Gibbs step on c targeting the auxiliary distribution $\tilde{\pi}$,

- K_3 deterministically projects the triplet back to the original space, retaining only the clustering c.

Formally:

$$K_1(s', c_-', c' \mid c) := h(s')\mathbf{1}[c = c']\mathbf{1}[c_-' = c_-(s,c)], \tag{21}$$
$$K_2(s', c_-', c' \mid s, c_-, c) := \tilde{\pi}(c'|s, c_-)\mathbf{1}[s' = s]\mathbf{1}[c_-' = c_-],$$
$$K_3(c'|s, c_-, c) := \mathbf{1}[c' = c].$$

Since $\tilde{\pi}$ admits π as a marginal, the composition of $K_1, K_2,$ and K_3 is clearly π-invariant. It is therefore enough to show that when $c_- = c_-(s,c)$ where c is a valid partition of $[T]$, sampling from K_2 is equivalent to sampling from the Markov kernel $K(c'|c)$ induced by Algorithm 1:

$$\tilde{\pi}(c' \mid s, c_-) \propto \pi(c')\mathbf{1}[c_- = c_-(s, c')] \tag{22}$$
$$= \pi(c')\mathbf{1}[c_-(s, c) = c_-(s, c')]$$
$$= \tau_1(|c'|) \left(\prod_{b \in c'} \tau_2(|b|)L(\mathbf{y}_b) \right) \mathbf{1}[c_-(s, c) = c_-(s, c')].$$

Using Lemma 5, we now rewrite the support as follows:

$$\mathbf{1}[c_-(s, c') = c_-(s, c)] = \mathbf{1}[c \cap c_- = c' \cap c_-]\mathbf{1}[b \in c'\backslash c_- \implies b \cap s \neq \emptyset]. \tag{23}$$

Figure 12: Graphical models of the auxiliary variables used in the correctness proofs. The structure of the dependencies give an intuitive justification that the original model can be recovered as a marginal in both cases, as there are not directed path from the auxiliary variables to the original variables. (a) In Appendix A, the auxiliary variables are s and c_-, and the original variable is c. (b) In Appendix B, the auxiliary variables are $\boldsymbol{\sigma}$ and g, and the original variable is $\bar{c}$.

Let now $\bar{c}' = c' \backslash c_-$. Plugging in the last line of Equation (22), we obtain:

$$\tilde{\pi}(c' \mid s, c_-) = \bar{\tau}_1(|\bar{c}'|) \left(\prod_{b \in \bar{c}'} \tau_2(|b|) L(\mathbf{y}_b) \right) \mathbf{1}[c \cap c_- = c' \cap c_-] \mathbf{1}[b \cap s \neq \emptyset] \quad (24)$$

$$= \bar{\pi}(\bar{c}') \mathbf{1}[c \cap c_- = c' \cap c_-],$$

where $\bar{\pi}(\bar{c}')$ is defined in Equation (5). Since Algorithm 1 does not change the clustering of points outside of $\bar{s}$ (line 6 of Algorithm 1), it follows that the indicator function in the last line of Equation (24) is equal to one. ∎

Appendix B. Correctness of particle Gibbs for split merge

We provide here the proof of Proposition 2. The main steps in the proof follow a structure similar to the proof of Proposition 1.

Proposition 7 *Under Assumption 1, 2, and 3, and if $\bar{c} \sim \pi$, then the output of Algorithm 3, $\bar{c}'$, satisfies $\bar{c}' \sim \pi$ for any $N \geq 2$; i.e. the Markov kernel $\overline{K}(\bar{c}'|\bar{c})$ induced by Algorithm 3 is π-invariant.*

Proof We augment the model $\bar{c}$ with the auxiliary variables $\boldsymbol{\sigma}$ and g (see Figure 12(b)), defined as:

1. $\boldsymbol{\sigma}$ is distributed according to the output of Algorithm 2, defined in Section 3.2.

2. Given $\boldsymbol{\sigma}$ and $\bar{c}$, the variables $g = (\boldsymbol{a}_{2:n}, \mathbf{x}_{1:n}^{1:N}, k)$ are distributed according to the specification of Algorithm 3, with the exception that all particle indices are shuffled according to an independent permutation of $\{1, \ldots, N\}$ at each generation. Here k is the index of the particle sampled at iteration n (on line (35)).

Next, we introduce three kernels with inputs and outputs denoted by:

$$\bar{c} \xmapsto{\overline{K}_1} (\boldsymbol{\sigma}, \bar{c}) \xmapsto{\overline{K}_2} (\boldsymbol{\sigma}, g, \bar{c}') \xmapsto{\overline{K}_3} \bar{c}'. \quad (25)$$

These kernels play the following roles:

- $\overline{K}_1$ samples the permutation $\boldsymbol{\sigma}$ while keeping the auxiliary variables $\bar{c}$ fixed,

- $\overline{K}_2$ samples g using the PG step then sets $\bar{c}'$ to $\phi^{\boldsymbol{\sigma}}(\mathbf{x}_n^k)$,

- $\overline{K}_3$ deterministically projects the triplet back to the original space, retaining only the restricted clustering $\bar{c}'$.

The kernel $\overline{K}_2$ is equivalent to a standard PG algorithm. Assumption 1, 3, and Theorem 5(a) of Andrieu et al. (2010) imply that $\overline{K}_2$ is $\overline{\pi}$-invariant (and in fact, irreducible). Assumption 2 ensures that the computation of the conditioned path is well-defined. ∎

Appendix C. Construction of the bijection

We provide here the proof of Proposition 3:

Proposition 8 *For any permutation $\boldsymbol{\sigma}$ satisfying $\{\sigma_1, \sigma_2\} = s$, there is a bijective map $\phi^{\boldsymbol{\sigma}}$ from the space of particles respecting the transition constraints, $\mathcal{S}_n$, to the support of the restricted target, $\mathrm{support}(\overline{\pi})$.*

Proof Consider the following mapping:

$$\phi^{\boldsymbol{\sigma}}(\mathbf{x}_t) := \begin{cases} \{\{\sigma_1, \ldots, \sigma_t\}\} & \text{if } x_t \in \{\#1, \#2\}, \\ \{\overline{\sigma}_1(\mathbf{x}_t), \overline{\sigma}_2(\mathbf{x}_t)\} & \text{otherwise,} \end{cases} \tag{26}$$

where $\overline{\sigma}_i(\mathbf{x}_t) := \{\sigma_{t'} : x_{t'} = \#(2+i), 1 \leq t' \leq t\}$. It is easy to check that it has an inverse given by:

$$\left(\left(\phi^{\boldsymbol{\sigma}}\right)^{-1}(\bar{c})\right)_t := \begin{cases} \#1 & \text{if } t = 1, \\ \#2 & \text{if } t > 1, |\bar{c}| = 1, \\ \#3 & \text{if } t > 1, |\bar{c}| > 1, \sigma_1 \sim_{\bar{c}} \sigma_t, \\ \#4 & \text{if } t > 1, |\bar{c}| > 1, \sigma_2 \sim_{\bar{c}} \sigma_t, \end{cases} \tag{27}$$

where $\sigma_i \sim_{\bar{c}} \sigma_j$ means that y_{σ_i} is in the same block as y_{σ_j} for the clustering $\bar{c}$. By the construction of the support of $\overline{\pi}$, exactly one of the four cases above holds when $\bar{c} \in \mathrm{support}(\overline{\pi})$.

∎

Appendix D. Generalization to $|s| > 2$

We describe here the algorithmic implications of increasing the number of anchor points, $|s|$, to some constant greater than two. This constant should be selected so that the number of partitions of $|s|$ points is much lower than the number of particles.

The algorithm is generally unchanged, with the following exceptions:

1. Algorithm 2 is modified to sample $(\sigma_1, \sigma_2, \ldots, \sigma_{|s|})$ uniformly over the permutations of s, and $(\sigma_{|s|+1}, \ldots, \sigma_n)$, over the permutations of $\overline{s} \backslash s$,

2. as before, the local allocation state space $\mathcal{X}$ can be viewed as a pair each containing a partition and a block in this partition (see Figure 2). In the case where $|s| = 2$, the partitions are taken from the union of the set of partitions of a set of size one with the set of partitions of a set of size two. When $|s| > 2$, we add more states, corresponding to partitions of a set of size three, etc. until we add states corresponding to partitions of a set of size $|s|$. The support of the transition $\mathcal{S}$ consists in (a) edges $x \to x'$ linking a state x' such that removing one element from one of its blocks yields x, and (b) edges $x \to x'$ where x and x' correspond to the same partition of a set of size $|s|$. This is a generalization of the case $|s| = 2$ shown in Figure 2. The mapping ϕ^{σ} is generalized in the obvious way,

3. in Section 3.4 the following equations are substituted,

   (a) $t \in \{1, 2\} \to t \in \{1, 2, \dots, |s|\}$,
   (b) $t = 2 \to t \in \{2, \dots, |s|\}$,
   (c) $t > 2 \to t > |s|$,
   (d) $\Delta\zeta := (n-2)^{-1} \to \Delta\zeta := (n-|s|)^{-1}$.

Appendix E. Models

E.1 Multivariate normal

The first likelihood we use is the multivariate normal (MVN) with density denoted $\mathcal{N}(y|\mu, \Sigma)$. We specify a normal inverse Wishart (NIW) prior for the mean and covariance parameters with density denoted $\mathcal{NIW}(\mu, \Sigma|\nu, r, u, S)$. The densities are given by

$$\mathcal{NIW}(\mu, \Sigma|\nu, r, u, S) = \mathcal{N}\left(\mu|u, \frac{1}{r}\Sigma\right)\mathcal{IW}(\Sigma|\nu, S), \tag{28}$$

$$\mathcal{N}(y|\mu, \Sigma) = \frac{1}{(2\pi)^{\frac{D}{2}}|\Sigma|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(y - \mu)^T \Sigma^{-1}(y - \mu)\right),$$

$$\mathcal{IW}(\Sigma|\nu, S) = \frac{|S|^{\frac{\nu}{2}}}{2^{\frac{\nu p}{2}}\Gamma_D\left(\frac{\nu}{2}\right)}|\Sigma|^{-\frac{\nu+p+1}{2}}\exp\left(-\frac{1}{2}\mathrm{tr}(S\Sigma^{-1})\right),$$

where $\Gamma_D(x) = \pi^{\frac{D(D-1)}{4}}\prod_{d=1}^{D}\Gamma\left(x + \frac{d-1}{2}\right)$.

We use the following priors for all experiments $(\nu, r, u, S) = (\nu_0, r_0, u_0, S_0) = (2 + D, 1, \mathbf{0}, \mathbf{I})$, where $\mathbf{0}$ is the D dimensional vector of zeros, and $\mathbf{I}$ is the D dimensional identity matrix. The posterior distribution of μ, Σ given $\boldsymbol{y} = (y_1, \dots, y_m)$ is $\mathcal{NIW}(\mu, \Sigma|\nu_m, r_m, u_m, S_m)$ where

$$\begin{aligned}
\nu_m &= \nu_0 + m, \tag{29}\\
r_m &= r_0 + m,\\
u_m &= \frac{r_0 u_0 + \sum_{i=1}^{m} y_i}{r_m},\\
S_m &= S_0 + \sum_{i=1}^{m} y_i y_i^T + r_0 u_0 u_0^T - r_m u_m u_m^T.
\end{aligned}$$

For computational efficiency it is convenient to express these updates iteratively using the following equations:

$$
\begin{aligned}
\nu_m &= \nu_{m-1} + 1, \\
r_m &= r_{m-1} + 1, \\
u_m &= \frac{r_{m-1}u_{m-1} + y_m}{r_m}, \\
S_m &= S_{m-1} + \frac{r_m}{r_{m-1}}(y_m - u_m)(y_m - u_m)^T.
\end{aligned}
\tag{30}
$$

Using these equations the Cholesky decomposition of S_0 can be performed once using $O(D^3)$ operations and cached. This decomposition can then be updated using m rank one updates, each requiring $O(D^2)$ operations, to obtain S_m. This allows for efficient evaluation of the marginal and predictive likelihoods as $|S_m|$ can be evaluated using $O(D)$ operations using the Cholesky decomposition, instead of the standard $O(D^3)$ operations.

The marginal likelihood for the MVN-NIW congugate pair is

$$
\begin{aligned}
L(\boldsymbol{y}) &= \int \prod_{i=1}^{m} L(y_i|\theta)H(\mathrm{d}\theta) \\
&= \int \prod_{i=1}^{m} \mathcal{N}(y_i|\mu,\Sigma)\mathcal{NIW}(\mu,\Sigma|\nu,r,u,S)\,\mathrm{d}\mu\,\mathrm{d}\Sigma \\
&= \frac{1}{\pi^{\frac{mD}{2}}} \frac{r_0^{\frac{D}{2}}}{r_m^{\frac{D}{2}}} \frac{|S_0|^{\frac{\nu_0}{2}}}{|S_m|^{\frac{\nu_m}{2}}} \frac{\prod_{d=1}^{D}\Gamma\left(\frac{\nu_m+d-1}{2}\right)}{\prod_{d=1}^{D}\Gamma\left(\frac{\nu_0+d-1}{2}\right)}.
\end{aligned}
\tag{31}
$$

The predictive likelihood is given by

$$
\begin{aligned}
L(\boldsymbol{y}^+|\boldsymbol{y}^-) &= \frac{L(y_1,\ldots,y_m)}{L(y_1,\ldots,y_{m-1})} \\
&= \frac{1}{\pi^{\frac{D}{2}}} \frac{r_{m-1}^{\frac{D}{2}}}{r_m^{\frac{D}{2}}} \frac{|S_{m-1}|^{\frac{\nu_{m-1}}{2}}}{|S_m|^{\frac{\nu_m}{2}}} \frac{\prod_{d=1}^{D}\Gamma\left(\frac{\nu_m+d-1}{2}\right)}{\prod_{d=1}^{D}\Gamma\left(\frac{\nu_{m-1}+d-1}{2}\right)}.
\end{aligned}
\tag{32}
$$

E.2 Bernoulli

We use a Bernoulli likelihood, Bernoulli$(x|\theta)$, with a Beta prior distribution, Beta$(\theta|\alpha,\beta)$. We use the following priors $(\alpha,\beta) = (\alpha_0,\beta_0) = (1,1)$ for all experiments. The densities are

$$
\begin{aligned}
\text{Bernoulli}(x|\theta) &= \theta^x(1-\theta)^{1-x}, \\
\text{Beta}(\theta|\alpha,\beta) &= \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)}\theta^{\alpha-1}\theta^{\beta-1}.
\end{aligned}
\tag{33}
$$

The posterior density of θ given $\boldsymbol{y} = (y_1, \ldots, y_m)$ is $\text{Beta}(\alpha_m, \beta_m)$ where $\alpha_m = \alpha_0 + \sum_{i=1}^m y_i$ and $\beta_m = \beta_0 + \sum_{i=1}^m (1 - y_i)$. The marginal likelihood is

$$
\begin{aligned}
L(\boldsymbol{y}) &= \int \prod_{i=1}^m L(y_i | \theta) H(\mathrm{d}\theta) \qquad\qquad\qquad (34) \\
&= \int \prod_{i=1}^m \text{Bernoulli}(y_i | \theta) \text{Beta}(\theta | \alpha_0, \beta_0) \, \mathrm{d}\theta \\
&= \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha_m)\Gamma(\beta_m)} \frac{\Gamma(\alpha_m + \beta_m)}{\Gamma(\alpha_0 + \beta_0)},
\end{aligned}
$$

and the predictive log likelihood is

$$
\begin{aligned}
L(\boldsymbol{y}^+ | \boldsymbol{y}^-) &= \frac{L(y_1, \ldots, y_m)}{L(y_1, \ldots, y_{m-1})} \qquad\qquad\qquad (35) \\
&= \frac{\Gamma(\alpha_{m-1})\Gamma(\beta_{m-1})}{\Gamma(\alpha_m)\Gamma(\beta_m)} \frac{\Gamma(\alpha_m + \beta_m)}{\Gamma(\alpha_{m-1} + \beta_{m-1})}.
\end{aligned}
$$

E.3 PyClone

For the cancer genomics data we use the application-specific PyClone likelihood model over clonal prevalences, genotypes, and observed read counts. The key variables in the model are as follows (see Roth et al. (2014) for a more detailed description of the model):

$$
\begin{aligned}
\phi_i \quad &: \quad \text{proportion of cancer cells with mutation } i, \; \phi_i \in [0, 1], \\
t \quad &: \quad \text{proportion of cancer cells in a sample (treated as known)}, \; t \in [0, 1], \\
\psi_i \quad &: \quad \text{genotype of normal, non-mutated cancer and mutated cancer cells}, \; \psi_i \in (g_N, g_R, g_V), \\
g_x \quad &\in \quad \mathcal{G} = \{\text{A}, \text{B}, \text{AA}, \text{AB}, \ldots\}, \\
\pi_{i, \psi_i} \quad &: \quad \text{probability that mutation } i \text{ has genotype } \psi_i \text{ (elicited from auxillary data)}, \\
c(g_x) \quad &= \quad \#\text{A}(g_x) + \#\text{B}(g_x), \\
\mu(g_x) \quad &= \quad \frac{\#\text{A}(g_x)}{c(g_x)}, \\
\xi(\psi, \phi, t) \quad &: \quad \text{probability of sampling a B from the population of cells in the sample, i.e.:} \\
&= \quad \frac{(1-t)c(g_N)\mu(g_N) + t(1-\phi)c(g_R)\mu(g_R) + t\phi c(g_V)\mu(g_V)}{(1-t)c(g_N) + t(1-\phi)c(g_R) + t\phi c(g_V)}, \\
y_i \quad &: \quad \text{number of sequence reads with a B and total number of reads covering mutation } i, \text{ i.e.:} \\
&= \quad (y_{i,b}, y_{i,d}) \in \mathbb{N}^2.
\end{aligned}
$$

The generative model is specified as follow:

$$
\begin{aligned}
H_0 &= \text{Uniform}([0, 1]), \qquad\qquad\qquad (36) \\
\alpha_0 &\sim \text{Gamma}(\alpha_0 | a, b), \\
H | \alpha_0, H_0 &\sim \text{DP}(H | \alpha_0, H_0), \\
\phi_i | H &\sim H, \\
y_i | \psi_i, \phi_i, t &\sim \text{Binomial}(y_{i,b} | y_{i,d}, \xi(\psi_i, \phi, t)).
\end{aligned}
$$

This model is not conjugate. However, if we let $x \in \{x_0, \ldots, x_M\} = \left\{0, \frac{1}{M-1}, \ldots, \frac{M-2}{M-1}, 1\right\}$ be a discretization of the interval $[0, 1]$ and replace the continuous uniform base measure, $H_0 = \text{Uniform}([0, 1])$, with the discrete uniform measure, $H_0 = \text{Uniform}(\{x_0, \ldots, x_M\})$, then we can approximate the model. Using this approximation, we can now treat the model as if it were conjugate. The marginal likelihood for data $(y_1, ..., y_m)$ is given by

$$
\begin{aligned}
\int \prod_{i=1}^{m} L(y_i|\theta) H(\mathrm{d}\theta) &= \int \prod_{i=1}^{m} \sum_{\psi_i \in \mathcal{G}^3} \pi_{i,\psi_i} \text{Binomial}(y_{i,b}|y_{i,d}, \xi(\psi_i, \phi, t)) H(\mathrm{d}\phi) & (37) \\
&= \sum_{k=0}^{M} \prod_{i=1}^{m} \sum_{\psi_i \in \mathcal{G}^3} \pi_{i,\psi_i} \text{Binomial}(y_{i,b}|y_{i,d}, \xi(\psi_i, x_k, t)) \frac{1}{M} \\
&= \sum_{k=0}^{M} \prod_{i=1}^{m} \exp\left(\log \underbrace{\sum_{\psi_i \in \mathcal{G}^3} \pi_{i,\psi_i} \text{Binomial}(y_{i,b}|y_{i,d}, \xi(\psi_i, x_k, t))}_{\Xi_k(y_i)} \right) \frac{1}{M} \\
&= \sum_{k=0}^{M} \exp\left(\sum_{i=1}^{m} \Xi_k(y_i) \right) \frac{1}{M},
\end{aligned}
$$

where we have the sufficient statistics

$$
\mathbf{\Xi}(y_i) = (\Xi_0(y_i), \ldots, \Xi_M(y_i)). \tag{38}
$$

Remark 9 *The possibly infinite sum $\sum_{\psi_i \in \mathcal{G}^3}$ is truncated to a finite sum over biologically plausible states.*

Appendix F. Anchor proposal distribution

The anchor proposal distribution, h, is a free tuning parameter for the PGSM sampler. In principle, proposals which are informed by the current clustering state of the chain or by the topology of the space may improve the performance of the sampler.

We consider two informed proposal distributions. While bespoke proposals for each model may perform better, we restrict attention here to proposals which can be applied generically to any class of model for which the PGSM sampler is applicable. In particular, we do not assume a distance metric is available. Both proposals we discuss are only applicable when two anchor points are used.

Remark 10 *If the any of the sets that we sample uniformly from are empty, we return two anchors sampled uniformly at random.*

Algorithm 4 Cluster informed (CI) proposal

1: $i_1 \sim \text{Uniform}([T])$
2: $\bar{b} \leftarrow b \in c$ s.t. $i_1 \in b$
3: $c' \leftarrow c \setminus \{\bar{b}\}$
4: **for** $b \in c'$ **do**
5:      $s_b \leftarrow \frac{L(y_{\bar{b} \cup b})}{L(y_{\bar{b}})L(y_b)}$
6: **end for**
7: $s_{\bar{b}} \leftarrow \frac{\sum_{b \in c'} s_b}{|c|-1}$ ▷ Merge probability is set to $\frac{1}{|c|-1}$
8: **for** $b \in c$ **do**
9:      $p_b \leftarrow \frac{s_b}{\sum_{b \in c} s_b}$
10: **end for**
11: $b' \sim \text{Discrete}(c, p_b)$ ▷ Sample a block b' in c with probability p_b
12: $i_2 \sim \text{Uniform}(b' \setminus i_1)$
13: **return** i_1, i_2

Algorithm 5 Threshold informed (TI) proposal

1: $i_1 \sim \text{Uniform}([T])$
2: **for** $b \in c$ **do**
3:      **if** $i_1 \in b$ **then**
4:          $b \leftarrow b \setminus i_1$
5:      **end if**
6:      $s_b \leftarrow \tau_2(b)L(y_{i_1}|b)$ ▷ CRP attachment probability where $L(\cdot|b)$ is the predictive distribution
7: **end for**
8: **for** $b \in c$ **do**
9:      $p_b \leftarrow \frac{s_b}{\sum_{b \in c} s_b}$
10: **end for**
11: $b' \sim \text{Uniform}(\{b : p_b \geq t\})$ ▷ t is a pre-specified threshold, set to 0.01 in the experiments
12: $i_2 \sim \text{Uniform}(b' \setminus i_1)$
13: **return** i_1, i_2

References

C. Andrieu, A. Doucet, and R. Holenstein. Particle Markov chain Monte Carlo for efficient numerical simulation. In *Monte Carlo and Quasi Monte Carlo Methods Conference 2008, Lecture Notes in Statistics*, pages 45–60. Springer-Verlag, 2009.

C. Andrieu, A. Doucet, and R. Holenstein. Particle Markov chain Monte Carlo methods. *Journal of the Royal Statistical Society: Series B*, 72(3):269–342, 2010.

F. Caron and A. Doucet. Bayesian nonparametric models on decomposable graphs. In *Advances in Neural Information Processing Systems*, pages 225–233. 2009.

C. Carvalho, H. Lopes, N. Polson, and M. Taddy. Particle learning for general mixtures. *Bayesian Analysis*, 5(4):709–740, 12 2010.

J. Chang and J. W. Fisher III. Parallel sampling of Dirichlet process mixutre models using sub-clusters splits. In *Advances in Neural Information Processing Systems*, pages 620–628, 2013.

N. Chopin and S. S. Singh. On particle Gibbs sampling. *Bernoulli*, 21(3):1855–1883, 2015.

D. B. Dahl. An improved merge-split sampler for conjugate Dirichlet process mixture models. Technical Report 1086, University of Wisconsin–Madison, 2003.

D. B. Dahl. Modal clustering in a univariate class of product partition models. *Bayesian Analysis*, 4(2):243–264, 2009.

H. Daumé III. Fast search for Dirichlet process mixture models. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 2007.

P. Del Moral, A. Doucet, and A. Jasra. Sequential Monte Carlo samplers. *Journal of the Royal Statistical Society: Series B*, 68(3):411–436, 2006.

A. Doucet and A. M. Johansen. A tutorial on particle filtering and smoothing: Fifteen years later. In *Oxford Handbook of Nonlinear Filtering*. Oxford University Press, 2011.

M. D. Escobar and M. West. Bayesian density estimation and inference using mixtures. *Journal of the American Statistical Association*, 90(430):577–588, 1995.

P. Fearnhead. Particle filters for mixture models with an unknown number of components. *Statistics and Computing*, 14:11–21, 2004.

P. Fearnhead and P. Clifford. On-line inference for hidden Markov models via particle filters. *Journal of the Royal Statistical Society: Series B*, 65(4):887–899, 2003.

P. Fearnhead and L. Meligkotsidou. Filtering methods for mixture models. *Journal of Computational and Graphical Statistics*, 16(3):586–607, 2007.

T. S. Ferguson. A Bayesian analysis of some nonparametric problems. *The Annals of Statistics*, 1(2):209–230, 1973.

P. Franti and O. Virmajoki. Iterative shrinking method for clustering problems. *Pattern Recognition*, 39(5):761–765, 2006.

P. Fränti, O. Virmajoki, and V. Hautamäki. Fast agglomerative clustering using a k-nearest neighbor graph. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 28(11):1875–1881, 2006.

P. Fränti, J. Kuittinen, A. Tabarcea, and L. Sakala. MOPSI location-based search engine: concept, architecture and prototype. In *ACM Symposium on Applied Computing*, volume 10, pages 872–873, 2010.

Y. Gal and Z. Ghahramani. Pitfalls in the use of parallel inference for the Dirichlet process. In *Proceedings of the 31st International Conference on Machine Learning*, pages 208–216. JMLR Workshop and Conference Proceedings, 2014.

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

H. Ishwaran and L. F. James. Gibbs sampling methods for stick-breaking priors. *Journal of the American Statistical Association*, 96:161–173, 2001.

H. Ishwaran and L. F. James. Generalized weighted Chinese restaurant processes for species sampling mixture models. *Statistica Sinica*, 13:1211–1235, 2003.

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

A. M. Johansen. SMCTC: Sequential Monte Carlo in C++. *Journal of Statistical Software*, 30(6):1–41, 2009.

S. Jun, L. Wang, and A. Bouchard-Côté. Entangled Monte Carlo. In *Advances in Neural Information Processing Systems*, volume 25, pages 2735–2743, 2012.

M. Kalli, J. E. Griffin, and S. G. Walker. Slice sampling mixture models. *Statistics and Computing*, 21(1):93–105, 2011.

N. Kantas, A. Doucet, S. S. Singh, J. M. Maciejowski, and N. Chopin. On particle methods for parameter estimation in state-space models. *Statistical Science*, 30(3):328–351, 2015.

A. Lee. *On auxiliary variables and many-core architectures in computational statistics*. PhD thesis, Department of Statistics, Oxford University, 2011.

A. Lee and N. Whiteley. Forest resampling for distributed sequential Monte Carlo. *Statistical Analysis and Data Mining*, 9(4):230–248, 2016.

A. Lee, C. Yau, M. Giles, A. Doucet, and C. Holmes. On the utility of graphics cards to perform massively parallel simulation of advanced Monte Carlo methods. *Journal of Computational and Graphical Statistics*, 19(4):769–789, 2010.

P. Liang, M. I. Jordan, and B. Taskar. A permutation-augmented sampler for Dirichlet process mixture models. *Proceedings of the International Conference on Machine learning*, 24:545–552, 2007.

F. Lindsten, M. I. Jordan, and T. B. Schön. Particle Gibbs with ancestor sampling. *Journal of Machine Learning Research*, 15:2145–2184, 2014.

J.S. Liu and R. Chen. Blind deconvolution via sequential imputation. *Journal of the American Statistical Association*, 90:567–576, 1995.

S. N. MacEachern. Estimating normal means with a conjugate style Dirichlet process prior. *Communications in Statistics: Simulation and Computation*, 23:727–741, 1994.

S. N. MacEachern, M. Clyde, and J. S. Liu. Sequential importance sampling for non-parametric Bayes models: The next generation. *Canadian Journal of Statistics*, 27(2): 251–267, 1999.

V. K. Mansinghka, D. M. Roy, R. Rifkin, and J. Tenenbaum. AClass: An online algorithm for generative classification. In *Proceedings of the 11th International Conference on Artificial Intelligence and Statistics*, volume 2, pages 315–322, 2007.

J. Marin, K. L. Mengersen, and C. Robert. Bayesian modelling and inference on mixtures of distributions. In D. Dey and C.R. Rao, editors, *Handbook of Statistics: Volume 25*. Elsevier, 2005.

L. Murray, A. Lee, and P. Jacob. Parallel resampling in the particle filter. *Journal of Computational and Graphical Statistics*, 25(3):789–805, 2016.

L. M. Murray. Bayesian state-space modelling on high-performance hardware using LibBi. *Journal of Statistical Software*, 67(10):1–36, 2015.

R. M. Neal. Markov chain sampling methods for Dirichlet process mixture models. *Journal of Computational and Graphical Statistics*, 9:249–265, 2000.

J. Pitman and M. Yor. The two-parameter Poisson-Dirichlet distribution derived from a stable subordinator. *Annals of Probability*, 25(2):855–900, 1997.

S. Richardson and P. J. Green. On Bayesian analysis of mixtures with an unknown number of components. *Journal of the Royal Statistical Society: Series B*, 59:731–792, 1997.

Andrew Rosenberg and Julia Hirschberg. V-measure: A conditional entropy-based external cluster evaluation measure. In *EMNLP-CoNLL*, volume 7, pages 410–420, 2007.

A. Roth, J. Khattra, D. Yap, A. Wan, E. Laks, J. Biele, G. Ha, S. Aparicio, A. Bouchard-Côté, and S. P. Shah. PyClone: statistical inference of clonal population structure in cancer. *Nature methods*, 11(4):396–398, 2014.

E. B. Sudderth. *Graphical Models for Visual Object Recognition and Tracking*. PhD thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, 2006.

Y. Ülker, B. Günsel, and A. T. Cemgil. Sequential Monte Carlo samplers for Dirichlet process mixtures. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*, volume 9, pages 876–883, 2010.

S. G. Walker. Sampling the Dirichlet mixture model with slices. *Communications in Statistics - Simulation and Computation*, 36(1):45–54, 2007.

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

N. Whiteley. Discussion on Particle Markov chain Monte Carlo methods. *Journal of the Royal Statistical Society: Series B*, 72:306–307, 2010.

N. Whiteley, C. Andrieu, and A. Doucet. Efficient Bayesian inference for switching state space models using discrete particle Markov chain Monte Carlo methods. Technical Report 10:04, Bristol Statistics Research Report, 2010.

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