Incremental Tree-Based Inference with Dependent Normalized Random Measures

Juho Lee and Seungjin Choi
Department of Computer Science and Engineering
Pohang University of Science and Technology
77 Cheongam-ro, Nam-gu, Pohang 790-784, Korea
{stonecold,seungjin}@postech.ac.kr

Abstract

Normalized random measures (NRMs) form a broad class of discrete random measures that are used as priors for Bayesian nonparametric models. Dependent normalized random measures (DNRMs) introduce dependencies in a set of NRMs, to facilitate the handling of data where the assumption of exchangeability is violated. Various methods have been developed to construct DNRMs; of particular interest is *mixed normalized random measures* (MNRMs), where DNRM is represented as a mixture of underlying shared normalized random measures. Emphasis in existing works is placed on the construction methods of DNRMs, but there is a little work on efficient inference for DNRMs. In this paper, we present a *tree-based inference* method for MNRM mixture models, extending Bayesian hierarchical clustering (BHC) which was originally developed as a deterministic approximate inference for Dirichlet process mixture (DPM) models. We also present an *incremental inference* for MNRM mixture models, building a tree incrementally in the sense that the tree structure is partially updated whenever a new data point comes in. The tree, when constructed in such a way, allows us to efficiently perform *tree-consistent MAP inference* in MRNM mixture models, determining a most probable tree-consistent partition, as well as to compute a marginal likelihood approximately. Numerical experiments on both synthetic and real-world datasets demonstrate the usefulness of our algorithm, compared to MCMC methods.

Appearing in Proceedings of the 17^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2014, Reykjavik, Iceland. JMLR: W&CP volume 33. Copyright 2014 by the authors.

1 Introduction

Normalized random measures (NRMs) [1] have recently been highlighted due to their flexibility and generality. As a broad class of discrete random measures, NRMs are used as priors in hierarchical Bayesian mixture modelling. NRMs include Dirichlet process (DP) [2] and normalized generalized Gamma process (NGGP) [3], the latter of which is an flexible prior with power-law behavior. Like Dirichlet process mixture models (DPMs) [4], NRMs apply well to NRM mixture models. Moreover, NRMs yield an exchangeable predictive distribution as the DP yields the exchangeable predictive distribution (usually referred to as the Chinese restaurant process (CRP) [5]); this makes the inference of NRM mixtures tractable.

The exchangeability of NRMs often fails with real-world datasets, such as time-varying data that are not exchangeable between different time steps, or spatial data that are not exchangeable between different locations. Those datasets shows dependency to certain covariates; for time varying data and spatial data, time and spatial locations, respectively, are those covariates. Hence, instead of a single random measure, models that consider a set of dependent random measures have been studied. In early time, dependent DPs (DDPs) [6] - which considers sets of dependent DPs were proposed, and many variations of DDP have been proposed [7, 8, 9, 10] by varying the way of giving dependency to multiple DPs. Such works has recently been extended to NRMs [11, 12, 13], forming a class of models called dependent NRMs (DNRMs).

Despite the development of flexible models, the inference of those models has not been extensively studied. For DPMs, various posterior inference algorithms were proposed, including MCMC sampling [14, 15], variational inference [16] and Bayesian hierarchical clustering [17]. However, for DDPs, NRMs and DNRMs, almost every model uses MCMC sampling for inference. Although MCMC sampling is flexible and convenient to derive, it is generally expensive. Moreover, MCMC is not eligible for incremental (online) inference when datasets are not given all at once, or when the size of a dataset is too large to fit in memory; thus, MCMC is not eligible for use in clustering with respect to large-scale or streaming datasets.

In this paper, we propose a novel posterior inference algorithm for DNRM mixtures. Our approach is based on Bayesian hierarchical clustering (BHC). BHC is a probabilistic hierarchical clustering algorithm, and at the same time, is an approximate posterior inference algorithm for DPMs. BHC builds a binary tree that represents the clustering of given data. This tree embeds the clusterings of datasets that can be obtained by cutting the tree, and each of those clusterings corresponds to a sample from the posterior of clusterings under DPM. L ikewise, our approach builds a binary tree that embeds the samples from the posterior of clusterings under DNRM mixtures. Moreover, unlike the original BHC, our approach incrementally builds trees by sequentially adding nodes. As a result, our new approach is much more efficient than the original BHC, and is well-suited to online inference.

2 Background

We briefly review NRMs [18, 1], and DNRMs with particular emphasis on recently-developed mixed normalized random measures (MNRMs) [12]. We also provide a brief overview of BHC [17], on which we base our incremental tree-based inference.

2.1 Normalized Random Measures

A completely random measure (CRM) [19] μ on a measure space (Θ, Ω) is a random measure such that for any disjoint $A_1, \ldots, A_n \subset \Theta$, the random variables $\mu(A_1), \ldots, \mu(A_n)$ are independent. A CRM can be represented as a linear functional of a Poisson process Π with intensity $\lambda(dw, d\theta)$ on the product space $\mathbb{R}^+ \times \Theta$:

$$\mu = \int_{\mathbb{R}^+} w\Pi(dw, d\theta) = \sum_k w_k \delta_{\theta_k}. \tag{1}$$

Here, λ is called the *Lévy intensity* of μ. μ is said to be *homogeneous* if its Lévy intensity decomposes into a product of two intensities, $\lambda(dw, d\theta) = \rho(dw)H(d\theta)$. Throughout this paper, we consider only homogeneous CRMs. We also assume that ρ satisfies:

$$\int_{\mathbb{R}^+} \rho(dw) = \infty, \quad \int_{\mathbb{R}^+} (1 - e^{-w})\rho(dw) < \infty, \tag{2}$$

which ensure that μ has infinite atoms and the total mass $\mu(\Theta) = \sum_{k=1}^{\infty} w_k$ is finite [1].

An NRM $\widetilde{\mu}$ is constructed by normalizing a CRM μ by its total mass $\mu(\Theta)$:

$$\mu \;=\; \sum_{k=1}^{\infty} w_k \delta_{\theta_k} \sim \mathrm{CRM}(\rho, H), \tag{3}$$

$$\widetilde{\mu} \;=\; \mu/\mu(\Theta). \tag{4}$$

A well-known example of NRM is a DP, which is obtained by normalizing a Gamma process that is a CRM with Lévy intensity $\rho(dw)H(d\theta) = \alpha w^{-1}e^{-w}dwH(d\theta)$. A generalized Gamma process (NGGP) is a CRM with Lévy intensity $\rho(dw)H(d\theta) = \frac{\alpha}{\Gamma(1-\sigma)}w^{-1-\sigma}e^{-\tau w}H(d\theta)$, where $\alpha > 0$, $\sigma \in (0, 1)$, and $\tau \geq 0$. Normalizing a GGP by its total mass yields a normalized GGP (NGGP) [3].

The calculation of the posterior distribution of NRMs is crucial in Bayesian nonparametric models. We present a short summary of the posterior analysis of NRMs, the details of which can be found in [1]. Denote by $I_n = \{1, \ldots, n\}$ a set of indices. A *partition* π of I_n is a set of disjoint nonempty subsets of I_n whose union is I_n. The set of all possible partitions of I_n is denoted by Π_n For instance, in the case of $I_5 = \{1, 2, 3, 4, 5\}$, an exemplary random partition that consists of three clusters is $\pi = \big\{\{1\}, \{2, 4\}, \{3, 5\}\big\}$; its members are indexed by $c \in \pi$. Suppose that we are given n observations $\{\theta_i\}$ drawn i.i.d. from a normalized random measure $\widetilde{\mu}$, i.e., $\theta_i \sim \widetilde{\mu}$ for $i = 1, \ldots, n$. Since $\widetilde{\mu}$ is discrete, different observations can take the same values, and thus induce a partition $\pi \in \Pi_n$ when observations with the same values are grouped as clusters. For each cluster $c \in \pi$, we denote by θ^*_c the unique value taken by c and let $w_c = \mu(\theta^*_c)$. The theorem below summarizes the posterior characterization of $\widetilde{\mu}$, given $(\pi, \{\theta^*_c\}_{c\in\pi})$.

Theorem 1 ([1, 20])**.** *Let* $\mu \sim \mathrm{CRM}(\rho, H)$ *on* (Θ, Ω) *and let* $(\pi, \{\theta^*_c\}_{c\in\pi}) \sim \widetilde{\mu}$*. Introducing an auxiliary variable* $\xi \sim \mathrm{Gamma}(n, \mu(\Theta))$*, the posterior of* μ *is given by*

$$\mu | \xi, \pi, \{\theta^*_c\} = \bar{\mu} + \sum_{c\in\pi} w_c \delta_{\theta^*_c}, \tag{5}$$

where

$$\bar{\mu} \sim \mathrm{CRM}(\bar{\rho}, H), \quad \bar{\rho}(dw) \;\overset{\text{def}}{=}\; e^{-\xi w}\rho(dw),$$

and the posterior density over random weights of fixed atoms is given by

$$p(w_c | \xi, \pi, \{\theta^*_c\}) \;=\; \frac{w_c^{|c|} e^{-\xi w_c} \rho(w_c)}{\kappa_\rho(|c|, \xi)},$$

$$\kappa_\rho(m, \xi) \;\overset{\text{def}}{=}\; \int_{\mathbb{R}^+} w^m e^{-\xi w} \rho(dw).$$

The marginal distribution is given by

$$p(\pi, \{\theta^*_c\}, \xi) \;=\; \frac{\xi^{n-1} e^{-\psi_\rho(\xi)}}{\Gamma(n)} \prod_{c\in\pi} \kappa_\rho(|c|, \xi) H(\theta^*_c),$$

$$\psi_\rho(\xi) \;\overset{\text{def}}{=}\; \int_{\mathbb{R}^+} (1 - e^{-\xi w})\rho(dw).$$

The predictive distribution is given by

$$\theta|\xi, \pi, \{\theta_c^*\} \propto \kappa_\rho(1, \xi)H + \sum_{c\in\pi} \frac{\kappa_\rho(|c|+1, \xi)}{\kappa_\rho(|c|, \xi)} \delta_{\theta_c^*}.$$

In the case of DP, the marginal distribution is written as

$$p(\pi, \{\theta_c^*\}, \xi) = \frac{\xi^{n-1}}{\Gamma(n)(\xi+1)^{n+\alpha}} \prod_{c\in\pi} \alpha\Gamma(|c|)H(\theta_c^*).$$

The auxiliary variable ξ can be easily marginalized out in this case, leading to

$$p(\pi, \{\theta_c^*\}) = \frac{\Gamma(\alpha)}{\Gamma(n+\alpha)} \prod_{c\in\pi} \alpha\Gamma(|c|)H(\theta_c^*).$$

The predictive distribution, which is also known as CRP, is then written as

$$p(\theta|\pi, \{\theta_c^*\}) \propto \alpha H + \sum_{c\in\pi} |c|\delta_{\theta_c^*}.$$

In the case of NGGP, however, the auxiliary variable ξ cannot be easily marginalized out. Instead, we get the following joint distribution of ξ and π:

$$p(\pi, \{\theta_c^*\}, \xi) = \frac{\xi^{n-1}\exp\{-\frac{\alpha}{\sigma}((\tau+\xi)^\sigma - \tau^\sigma)\}}{\Gamma(n)(\xi+\tau)^n} \times \prod_{c\in\pi} \frac{\alpha\Gamma(|c|-\sigma)}{(\xi+\tau)^{-\sigma}\Gamma(1-\sigma)} H(\theta_c^*).$$

2.2 Normalized Random Measure Mixture Models

Suppose that we are given a set of n observations $\boldsymbol{X} = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_n\}$. NRM mixture models assume that observations $\{\boldsymbol{x}_i\}$ are drawn from a distribution $L(\cdot|\theta_i)$, treating θ_i drawn from a NRM $\widetilde{\mu}$, as the parameter of the distribution of observation $\boldsymbol{x}_i$. The generation process in a NRM mixture model is described as follows.

$$\begin{aligned} \mu &\sim \mathrm{CRM}(\rho, H), &(6)\\ \theta_i|\widetilde{\mu} &\sim \widetilde{\mu}, \quad \widetilde{\mu} = \mu/\mu(\Theta), &(7)\\ \boldsymbol{x}_i|\theta_i &\sim L(\cdot|\theta_i). &(8) \end{aligned}$$

For each cluster $c \in \pi$, we denote by $\boldsymbol{X}^{(c)} \overset{\text{def}}{=} \{\boldsymbol{x}_i \,|\, i \in c\}$ a collection of $\boldsymbol{x}_i$'s where index i is associated with c. Choosing the base distribution H that is conjugate to the likelihood $L(\cdot|\theta_i)$ yields the joint distribution in the NRM mixture model that is of the form

$$p(\boldsymbol{X}, \pi, \xi) \propto \prod_{c\in\pi} \kappa_\rho(|c|, \xi) p(\boldsymbol{X}^{(c)}|\mathcal{H}_c), \qquad (9)$$

where

$$p(\boldsymbol{X}^{(c)}|\mathcal{H}_c) \overset{\text{def}}{=} \int_\theta \left\{ \prod_{i\in c} L(\boldsymbol{x}_i|\theta) \right\} H(d\theta). \qquad (10)$$

Here, $\mathcal{H}_c$ is a hypothesis that the data in c were generated from a single cluster, and $p(\boldsymbol{X}^{(c)}|\mathcal{H}_c)$ is the marginal probability when $\mathcal{H}_c$ holds. Posterior inference in an NRM mixture evaluates the posterior distribution $p(\pi|\boldsymbol{X}, \xi)$, and so that the most probable partition π given $\boldsymbol{X}$ can be calculated. However, this computation is intractable since the evaluation of the marginal likelihood $p(\boldsymbol{X}, \xi)$,

$$p(\boldsymbol{X}, \xi) \propto \sum_{\pi\in\Pi_n} \prod_{c\in\pi} \kappa_\rho(|c|, \xi) p(\boldsymbol{X}^{(c)}|\mathcal{H}_c), \qquad (11)$$

requires a summation over Π_n. The most popular approach to approximate inference is the use of an MCMC sampler [21], where $p(\boldsymbol{X}, \xi)$ is approximately computed by drawing many samples from $p(\pi|\boldsymbol{X}, \xi)$. Sequential Monte Carlo (SMC) approximates $p(\pi|\boldsymbol{X}, \xi)$ by generating fixed number of sequentially growing particles [22].

2.3 Dependent Normalized Random Measure Mixtures

As mentioned, there are several ways to construct dependent NRMs, and we focus on a specific model called the mixed normalized random measures (MNRMs) proposed in [12]. Assume that we want to model T datasets indexed by a covariate t. Let $\boldsymbol{X}_t = \{\boldsymbol{x}_{t,1}, \dots, \boldsymbol{x}_{t,n_t}\}$ and $\boldsymbol{X}_{1:T} = \bigcup_{t=1}^T \boldsymbol{X}_t$. Let $I_{t,n_t} = \{(t,1), \dots, (t, n_t)\}$ be a set of indices for $\boldsymbol{X}_t$ where each index is represented as a tuple of covariate t and data index i. We write $I_{1:T} = \bigcup_{t=1}^T I_{t,n_t}$.

In MNRM, a set of independent *basis CRMs* on (Θ, Ω) is defined with covariate $r = 1, \dots, R$, written as

$$\nu_r = \sum_{k=1}^\infty w_{r,k}\delta_{\theta_{r,k}} \sim \mathrm{CRM}(\rho_r, H_r), \quad r = 1, \dots, R. \quad (12)$$

Then each dependent CRM μ_t is represented as a mixture of basis CRMs with mixing coefficients $q_{t,r}$.

$$\mu_t = \sum_{r=1}^R q_{t,r}\nu_r. \qquad (13)$$

For example, suppose that we are modelling a time-varying dataset that changes over time. Let $R \subset \{1, \dots, T\}$, and $q_{t,r} = \exp(-\gamma|t-r|)$[1]. If t and t' are close, the corresponding CRMs μ_t and $\mu_{t'}$ are similar since their mixing coefficients $q_{t,r}$ and $q_{t',r}$ are similar. MNRMs are then obtained by normalizing μ_ts.

Like in NRMs, Drawing $\{\{\theta_{t,i}\}_{i=1}^{n_t}\}_{t=1}^T$ from $\{\widetilde{\mu}_t\}_{t=1}^T$ induces a partition π of $I_{1:T}$ and the unique samples $\{\theta_c^*\}_{c\in\pi}$.

Theorem 2. *([12]) Let $(\pi, \{\theta_c^*\})$ be the samples drawn from DNRMs described as above. For each cluster $c \in \pi$, introduce an auxiliary variable z_c where $z_c = r$ means*

[1]These mixing coefficients were originally used in the earlier work on DDPs. [9]

that θ^*_c has come from the rth basis CRM $\nu_r(\nu_r(\theta^*_c) > 0)$. Additionally, introduce a set of auxiliary variables $\{\xi_t\}_{t=1}^T$ where $\xi_t \sim \mathrm{Gamma}(n_t, \mu_t(\Theta))$. Then the marginal and predictive distribution is written as

$$p(\pi, \{z_c, \theta^*_c\}, \{\xi_t\}) = \prod_{t=1}^{T} \frac{\xi_t^{n_t-1}}{\Gamma(n_t)} \prod_r e^{-\psi_{\rho_r}(\bar{\xi}_r)} \times \prod_{c\in\pi} \bar{q}_{c,z_c} \kappa_{\rho_r}(|c|, \bar{\xi}_{z_c}) H_{z_c}(\theta^*_c), \quad (14)$$

$$p(\theta^*|\pi, \{z_c, \theta^*_c\}) \propto \sum_{r=1}^{R} \kappa_{\rho_r}(1, \bar{\xi}_r) H_r + \sum_{c\in\pi} q_{t,z_c} \frac{\kappa_{\rho_{z_c}}(|c|+1, \bar{\xi}_{z_c})}{\kappa_{\rho_{z_c}}(|c|, \bar{\xi}_{z_c})} \delta_{\theta^*_c}, \quad (15)$$

where

$$\bar{q}_{c,z_c} \overset{\mathrm{def}}{=} \prod_{t=1}^{T} q_{t,z_c}^{|c\cap I_{t,n_t}|}, \quad \bar{\xi}_r \overset{\mathrm{def}}{=} \sum_{t=1}^{T} q_{t,r}\xi_r. \quad (16)$$

The marginal and predictive distribution of the MNRM has a form similar to that of NRM, except that it includes z_c indicating the basis CRMs from which each sample was drawn. Hence, the predictive distribution includes R possibilities of drawing new samples from each basis CRM ν_r.

The generative process of an MNRM mixture model is then straightforward,

$$\theta_{t,i} \sim \widetilde{\mu}_t, \quad \boldsymbol{x}_{t,i}|\theta_{t,i} \sim L(\cdot|\theta_{t,i}), \quad (17)$$

and the joint likelihood is written as

$$p(\boldsymbol{X}_{1:T}, \pi, \{\xi_t\}) \propto \prod_{c\in\pi} \bar{q}_{c,z_c} \kappa_{\rho_{z_c}}(|c|, \bar{\xi}_{z_c}) p(\boldsymbol{X}^{(c)}|z_c, \mathcal{H}_c), \quad (18)$$

where $p(\boldsymbol{X}^{(c)}|z_c, \mathcal{H}_c)$ is the same as (10) except that it is integrated with H_{z_c} instead of H.

2.4 Bayesian Hierarchical Clustering

Bayesian hierarchical clustering (BHC) is a agglomerative clustering, but unlike traditional methods, it uses marginal likelihoods of a probabilistic model to decide which clusters to merge, providing a few advantages over traditional distance-based agglomerative clustering algorithms [17]. BHC was also shown to provide a fast bottom-up approximate inference for Dirichlet process mixture models (DPMs). We briefly review BHC here since we base our tree-based inference on it.

The marginal likelihood of data $\boldsymbol{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ is written as a sum over all possible partitions:

$$p(\boldsymbol{X}) = \sum_{\pi\in\Pi_n} \left[\left(\prod_{c\in\pi} p(\boldsymbol{X}^{(c)}|\mathcal{H}_c) \right) p(\pi) \right], \quad (19)$$

where $p(\boldsymbol{X}^{(c)}|\mathcal{H}_c)$ is the marginal probability of $\boldsymbol{X}^{(c)}$ when the hypothesis $\mathcal{H}_c$ holds, where $\mathcal{H}_c$ assumes that all elements in $\boldsymbol{X}^{(c)}$ were generated from a single cluster c. In the case of mixture models with DP prior drawn from $\mathrm{DP}(\alpha, H)$, the marginal likelihood is of the form [17]:

$$p(\boldsymbol{X}) = \frac{\Gamma(\alpha)}{\Gamma(n+\alpha)} \sum_{\pi\in\Pi_n} \prod_{c\in\pi} \alpha\Gamma(|c|) p(\boldsymbol{X}^{(c)}|\mathcal{H}_c), \quad (20)$$

where $|c|$ is the size of cluster c and $\Gamma(\cdot)$ is gamma function. The evaluation of the marginal likelihood requires summing over *all possible partitions* of the data, which is not tractable in practice.

BHC approximately computes the marginal likelihood (20), summing over all tree-consistent partitions for a particular tree built greedily bottom-up, instead of summing over all possible partitions of the data. Such approximation was shown to yield a tight lower-bound on the marginal likelihood [23]. We denote by $\mathcal{T}_c$ a *binary tree* whose leaves are associated with $\boldsymbol{X}^{(c)}$. When binary trees are used to explain the generation of $\boldsymbol{X}^{(c)}$, only two cases are considered: (1) generation from a single cluster, reflected by $\mathcal{H}_c$; (2) two subclusters $\boldsymbol{X}^{(c_l)}$ and $\boldsymbol{X}^{(c_r)}$, each of which is associated with subtrees $\mathcal{T}_{c_l}$ and $\mathcal{T}_{c_r}$, respectively. Thus, the probability of $\boldsymbol{X}^{(c)}$ in tree $\mathcal{T}_c$ is written as:

$$p(\boldsymbol{X}^{(c)}|\mathcal{T}_c) = p(\mathcal{H}_c)p(\boldsymbol{X}^{(c)}|\mathcal{H}_c) + \big(1 - p(\mathcal{H}_c)\big) p(\boldsymbol{X}^{(c_l)}|\mathcal{T}_{c_l}) p(\boldsymbol{X}^{(c_r)}|\mathcal{T}_{c_r}). \quad (21)$$

The prior $p(\mathcal{H}_c)$ is recursively defined as follows:

$$\gamma_{\{i\}} \overset{\mathrm{def}}{=} \alpha, \ \ \gamma_c \overset{\mathrm{def}}{=} \alpha\Gamma(|c|) + \gamma_{c_l}\gamma_{c_r}, \quad p(\mathcal{H}_c) \overset{\mathrm{def}}{=} \alpha\Gamma(|c|)/\gamma_c, \quad (22)$$

The marginal likelihood in tree $\mathcal{T}_{I_n}$, computed by BHC, is given below, where one can see that BHC provides an approximate inference for DPMs.

Theorem 3 ([17]).

$$p(\boldsymbol{X}|\mathcal{T}_{I_n}) = \frac{1}{\gamma_{I_n}} \sum_{\pi\in\Pi_{\mathcal{T}_{I_n}}} \prod_{c\in\pi} \alpha\Gamma(|c|) p(\boldsymbol{X}^{(c)}|\mathcal{H}_c), \quad (23)$$

where $\Pi_{\mathcal{T}_{I_n}}$ is the set of all tree-consistent partitions of $\boldsymbol{X}$.

A notable distinction of the tree-consistent marginal likelihood (23), compared to the true marginal likelihood of DPM (20) is in summing over only tree-consistent partitions, rather than all possible partitions. In fact, it was shown in [17] that the tree-consistent marginal likelihood is a lower-bound on the marginal likelihood of DPM:

$$\frac{\Gamma(\alpha)\gamma_{I_n}}{\Gamma(n+\alpha)} p(\boldsymbol{X}|\mathcal{T}_{I_n}) \leq p(\boldsymbol{X}). \quad (24)$$

We explain how BHC builds a particular tree $\mathcal{T}_{I_n}$, given data $\boldsymbol{X}$. Starting from n leaves, BHC calculates the posterior probability $p(\mathcal{H}_c|\boldsymbol{X}^{(c)}, \mathcal{T}_c)$ for each pair (c_l, c_r) to decide which clusters to merge. Subsequent descriptions are slightly different from the original paper of BHC. To this end, we introduce the following *potential functions*

$$\phi(\boldsymbol{X}^{(c)}|\mathcal{T}_c) \overset{\text{def}}{=} \gamma_c\, p(\boldsymbol{X}^{(c)}|\mathcal{T}_c), \quad (25)$$

$$\phi(\boldsymbol{X}^{(c)}|\mathcal{H}_c) \overset{\text{def}}{=} \alpha\Gamma(|c|)\, p(\boldsymbol{X}^{(c)}|\mathcal{H}_c), \quad (26)$$

leading to a simpler form of recursion which combines (21) and (22)

$$\phi(\boldsymbol{X}^{(c)}|\mathcal{T}_c) = \phi(\boldsymbol{X}^{(c)}|\mathcal{H}_c) + \phi(\boldsymbol{X}^{(c_l)}|\mathcal{T}_{c_l})\phi(\boldsymbol{X}^{(c_r)}|\mathcal{T}_{c_r}). \quad (27)$$

With the potential functions defined in (25) and (26), the posterior probability $p(\mathcal{H}_c|\boldsymbol{X}^{(c)}, \mathcal{T}_c)$ for a pair (c_l, c_r) can be written as

$$p(\mathcal{H}_c|\boldsymbol{X}^{(c)}, \mathcal{T}_c) = \frac{p(\mathcal{H}_c)p(\boldsymbol{X}^{(c)}|\mathcal{H}_c)}{p(\boldsymbol{X}^{(c)}|\mathcal{T}_c)} = \frac{1}{1 + d(c_l \cup c_r)}, \quad (28)$$

where

$$d(c_l \cup c_r) \overset{\text{def}}{=} \frac{\phi(\boldsymbol{X}^{(c)}|\mathcal{T}_{c_l})\phi(\boldsymbol{X}^{(c)}|\mathcal{T}_{c_r})}{\phi(\boldsymbol{X}^{(c)}|\mathcal{H}_c)}. \quad (29)$$

In fact, $d(c_l \cup c_r)$ serves as a distance between clusters, since $p(\mathcal{H}_c|\boldsymbol{X}^{(c)}, \mathcal{T}_c)$ becomes larger as $d(c_l \cup c_r)$ becomes smaller. Thus, at each iteration, a pair of clusters with smallest d value is merged. The advantage of BHC over the traditional hierarchical clustering is that it can automatically choose the level at which to cut a tree (i.e., choose the proper number of clusters) using the probability $p(\mathcal{H}_c|\boldsymbol{X}^{(c)}, \mathcal{T}_c)$. If $p(\mathcal{H}_c|\boldsymbol{X}^{(c)}, \mathcal{T}_c) < 0.5$ (or $d(c_l \cup c_r) > 1$), c_l and c_r are concluded to be generated from different clusters and the merging stops. The BHC algorithm is summarized in **Algorithm 1**. The time complexity of BHC is $\mathcal{O}(n^2)$, as in the standard agglomerative clustering.

Algorithm 1 BHC

```
Input: $\boldsymbol{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$, $\alpha$
  Assign leaf nodes to each $\{i\}$ for $i = 1, \ldots, n$.
  Compute $d(c_l \cup c_r)$ for all pair $(i, j)$.
  while Minimum $d(c_l \cup c_r) < 1$ do
    Merge $c = c_l \cup c_r$ with smallest $d(c_l \cup c_r)$.
    Compute $d(c \cup c')$ for remaining nodes $c'$.
  end while
```

3 Tree-Based Inferences for MNRM Mixtures

In this section, we describe the main contribution of our paper. We first extend BHC, defining $\phi(\boldsymbol{X}^{(c)}|\mathcal{H}_c)$ appropriately for MNRM mixture models. Then we present an incremental tree-based inference algorithm, where we build a binary tree *incrementally* in the sense that the binary tree built is partially updated as a new node comes in.

3.1 BHC for MNRM Mixture Models

It follows from (18) that the marginal likelihood of a MNRM mixture model is given by

$$p(\boldsymbol{X}_{1:T}, \{\xi_t\}) \propto \sum_{\pi\in\Pi_{1:T}} \prod_{c\in\pi} \sum_{z_c} \bar{q}_{c,z_c}\kappa_{\rho_{z_c}}(|c|, \bar{\xi}_{z_c}) p(\boldsymbol{X}^{(c)}|z_c, \mathcal{H}_c), (30)$$

where $\Pi_{1:T}$ is a set of all possible partitions of $I_{1:T}$.

The marginal likelihood (30) is similar to the one of a DPM (20). Thus, it leads to the same recursive equation (27) for the potential function $\phi(\boldsymbol{X}^{(c)}|\mathcal{T}_c)$, while we define the potential function $\phi(\boldsymbol{X}^{(c)}|\mathcal{H}_c)$ as

$$\phi(\boldsymbol{X}^{(c)}|\mathcal{H}_c) \overset{\text{def}}{=} \sum_{z_c=1}^{R} \bar{q}_{c,z_c}\kappa_{\rho_{z_c}}(|c|, \bar{\xi}_{z_c}) p(\boldsymbol{X}^{(c)}|z_c, \mathcal{H}_c), \quad (31)$$

which is different from (26) defined for a DPM. As in DPMs, we obtain a similar approximate lower bound on (30), which is given below.

Theorem 4.

$$\phi(\boldsymbol{X}_{1:T}|\mathcal{T}_{I_{1:T}}) = \sum_{\pi\in\Pi_{1:T}|\mathcal{T}_{I_{1:T}}} \prod_{c\in\pi} \kappa_{\rho_{z_c}}(|c|, \bar{\xi}_{z_c}) p(\boldsymbol{X}^{(c)}|z_c, \mathcal{H}_c), (32)$$

which leads to

$$\prod_t \frac{\xi_t^{n_t - 1}}{\Gamma(n_t)} \prod_r e^{-\psi_{\rho_r}(\bar{\xi}_r)} \phi(\boldsymbol{X}_{1:T}|\mathcal{T}_{I_{1:T}}) \le p(\boldsymbol{X}_{1:T}, \{\xi_t\}). \quad (33)$$

This can be easily proved by a similar manner to **Theorem 3**.

3.2 Incremental Bayesian Hierarchical Clustering

The posterior inference may be done using original BHC (**Algorithm 1**), but there are two problems, in that case. First, we need to infer $\{\xi_t\}$ and hyperparameters. Actually, even in simple DPM, there is a hyperparameter α, so the authors of the original paper [17] proposed an EM-like algorithm that alternates between building the tree and optimizing α. This may be very expensive when n is large. The second problem is that BHC is designed to operate in a batch fashion; it is not applicable to incremental inference.

To resolve those problems, we propose incremental BHC (IBHC). Unlike the original BHC that builds the tree starting from the all the data points, IBHC *sequentially* inserts data points into the tree and the tree is completed after all the data points are inserted. Hence, it is well-suited for the incremental inferences needed for MNRM mixtures. Moreover, since IBHC does not need $\mathcal{O}(n^2)$ computation of distance measures for all the pair of nodes, it is much faster than BHC and thus alternating between building the

Algorithm 2 Seq-Insert

```
Input: $c = c_l \cup c_r, i$
  Compute $d(c_l \cup c_r)$, $d(c_l \cup \{i\})$ and $d(c_r \cup \{i\})$.
  if $d(c_l \cup c_r)$ is the smallest then
    insert $\{i\}$ next to $c$ and stop.
  else if $d(c_l \cup \{i\})$ is the smallest then
    Seq-Insert($c_l, \{i\}$).
  else
    Seq-Insert($c_r, \{i\}$).
  end if
```

tree and inferring auxiliary variables and hyperparameters is quite doable. Note that IBHC outcomes differ by the order of inserted data points; hence we try a number of random permutations and choose the best one.

Let us first consider a single dataset $\boldsymbol{X} = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_n\}$. Starting from a tree with a single data point, we sequentially insert the data points into the tree in the random order. Without loss of generality, assume that the data points are inserted in the order of $\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_n$. Suppose that a tree has been built with $\boldsymbol{x}_1, \dots, \boldsymbol{x}_{i-1}$, and we are about to insert $\boldsymbol{x}_i$ into this tree. The tree was cut at the level where the minimum d value was larger than 1, forming a partition π of I_{i-1}. For each top-level node $c \in \pi$, we compute $d(c \cup \{i\})$ and find the closest c. If the minimum d is larger than 1, we conclude that $\boldsymbol{x}_i$ belongs to a novel cluster and create a new leaf node with i; otherwise, $\boldsymbol{x}_i$ is inserted into cluster c using the *Seq-Insert* algorithm (Figure 1, left).

Now we describe the Seq-Insert algorithm. What the Seq-Insert essentially do is, when inserting a node into a cluster, to find the best position of the node inside the cluster, in terms of the potential function. Suppose that i is inserted into $c = c_l \cup c_r$. i is attached to one of the three-possible cases; as a sibling of c, as a sibling of c_l, or as a sibling of c_r (Figure 1, right). $\phi(\boldsymbol{X}^{(c\cup\{i\})}|\mathcal{T}_{c\cup\{i\}})$ for each case is computed as follows.

- **Case 1**: Inserting i as a sibling of $c = c_l \cup c_r$

$$\begin{aligned}\phi(\boldsymbol{X}^{(c\cup\{i\})}|\mathcal{T}_{c\cup\{i\}}) &= \phi(\boldsymbol{X}^{(c\cup\{i\})}|\mathcal{H}_{c\cup\{i\}}) \\ &+ \underline{\phi(\boldsymbol{X}^{(c)}|\mathcal{H}_c)\phi(\boldsymbol{x}_i|\mathcal{T}_{\{i\}})} \\ &+ \phi(\boldsymbol{X}^{(c_l)}|\mathcal{T}_{c_l})\phi(\boldsymbol{X}^{(c_r)}|\mathcal{T}_{c_r})\phi(\boldsymbol{x}_i|\mathcal{T}_{\{i\}})\end{aligned}$$

- **Case 2**: Inserting i as a sibling of c_l

$$\begin{aligned}\phi(\boldsymbol{X}^{(c\cup\{i\})}|\mathcal{T}_{c\cup\{i\}}) &= \phi(\boldsymbol{X}_{c\cup\{i\}}|\mathcal{H}_{c\cup\{i\}}) \\ &+ \underline{\phi(\boldsymbol{X}^{(c_l\cup\{i\})}|\mathcal{H}_{c_l\cup\{i\}})\phi(\boldsymbol{X}^{(c_r)}|\mathcal{T}_{c_r})} \\ &+ \phi(\boldsymbol{X}^{(c_l)}|\mathcal{T}_{c_l})\phi(\boldsymbol{X}^{(c_r)}|\mathcal{T}_{c_r})\phi(\boldsymbol{x}_i|\mathcal{T}_{\{i\}})\end{aligned}$$

- **Case 3**: Inserting i as a sibling of c_r

$$\begin{aligned}\phi(\boldsymbol{X}^{(c\cup\{i\})}|\mathcal{T}_{c\cup\{i\}}) &= \phi(\boldsymbol{X}_{c\cup\{i\}}|\mathcal{H}_{c\cup\{i\}}) \\ &+ \underline{\phi(\boldsymbol{X}^{(c_r\cup\{i\})}|\mathcal{H}_{c_r\cup\{i\}})\phi(\boldsymbol{X}^{(c_l)}|\mathcal{T}_{c_l})} \\ &+ \phi(\boldsymbol{X}^{(c_l)}|\mathcal{T}_{c_l})\phi(\boldsymbol{X}^{(c_r)}|\mathcal{T}_{c_r})\phi(\boldsymbol{x}_i|\mathcal{T}_{\{i\}})\end{aligned}$$

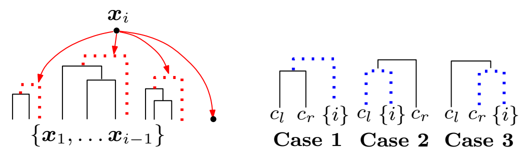

Figure 1: (Left) A new data point $\boldsymbol{x}_i$ may be inserted into the existing clusters, or create a novel one. (Right) Three possible locations of inserting i into $c = c_l \cup c_r$.

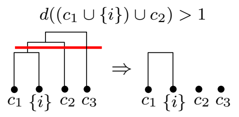
$d((c_1 \cup \{i\}) \cup c_2) > 1$
$\Rightarrow$
c_1 $\{i\}$ c_2 c_3
c_1 $\{i\}$ c_2 c_3

Figure 2: When splitting occurs after the Seq-Insert.

As a result, comparing the three cases reduces to comparing the three underlined quantities:

$$\begin{aligned}\underline{\phi(\boldsymbol{X}^{(c)}|\mathcal{H}_c)\phi(\boldsymbol{x}_i|\mathcal{T}_{\{i\}})} &= C \cdot d(c_l \cup c_r)^{-1} \\ \underline{\phi(\boldsymbol{X}^{(c_l\cup\{i\})}|\mathcal{H}_{c_l\cup\{i\}})\phi(\boldsymbol{X}^{(c_r)}|\mathcal{T}_{c_r})} &= C \cdot d(c_l \cup \{i\})^{-1} \\ \underline{\phi(\boldsymbol{X}^{(c_r\cup\{i\})}|\mathcal{H}_{c_r\cup\{i\}})\phi(\boldsymbol{X}^{(c_l)}|\mathcal{T}_{c_l})} &= C \cdot d(c_r \cup \{i\})^{-1},\end{aligned}$$

where $C = \phi(\boldsymbol{X}^{(c_l)}|\mathcal{T}_{c_l})\phi(\boldsymbol{X}^{(c_r)}|\mathcal{T}_{c_r})\phi(\boldsymbol{x}_i|\mathcal{T}_{\{i\}})$. Hence, we can choose the optimal case by comparing the d values among the nodes. If $d(c_l \cup c_r)$ is the smallest, i is attached next to c, at which point the Seq-Insert algorithm terminates. Otherwise, if $d(c_l \cup \{i\})$ is the smallest, i is attached next to c_l, and the Seq-Insert algorithm is recursively repeated with $c \leftarrow c_l$ (**Algorithm 2**).

After executing the Seq-Insert(c, i), the potential function $\phi(\boldsymbol{X}^{(c)\cup\{i\}}|\mathcal{T}_{c\cup\{i\}})$ is computed; starting from the level at which i was inserted, all the potential functions of clusters between the starting level and top level (c) are updated. As the potential functions change, the distance measures d also change, and some clusters may be split if their distances d get larger than 1. For example, in Figure 2, after i is inserted next to c_1, $d((c_1 \cup \{i\}) \cup c_2)$ becomes larger than 1, and thus c_2 and c_3 are split. The clusters split during the updates are kept as separate clusters. Suppose that k clusters remained after the insertion of the entire data. We run BHC (**Algorithm 1**) using those k clusters as bottom nodes to merge the clusters that were split during the insertion, and this can be done efficiently ($\mathcal{O}(k^2)$).

Once the tree is completed, we sample the auxiliary variables $\{\xi_t\}$ and hyperparameters given the partition get by cutting the tree. Then we fix those variables and build the tree again. We repeat building the tree and sampling the variables several times with different random orderings, and pick the tree with the highest log-likelihood.

Now we describe the algorithm that clusters $\boldsymbol{X}_{1:T} = \{\boldsymbol{X}_1, \dots, \boldsymbol{X}_T\}$ incrementally. Suppose that we have built a tree with $\boldsymbol{X}_{1:t} = \{\boldsymbol{X}_1, \dots, \boldsymbol{X}_t\}$, and the new dataset $\boldsymbol{X}_{t+1}$ enters. As explained earlier, we insert the new data points with the Seq-Insert, and resample the auxiliary variables $\{\xi_1, \dots, \xi_{t+1}\}$ and hyperparameters. Note that if

Algorithm 3 IBHC

```
Input: $\boldsymbol{X}_{1:T} = \bigcup_{t=1}^{T} \boldsymbol{X}_t, M$
  for $t = 1, \dots, T$ do
    for $m = 1, \dots, M$ do
      for $i \in$ random ordering of$\{1, \dots, n_t\}$ do
        for $c \in$ existing clusters do
          compute $d(c, \{i\})$.
        end for
        if Minimum $d(c, \{i\}) > 1$ then
          Set $\{i\}$ as a new cluster.
        else
          Seq-Insert$(c, \{i\})$ for $c$ with minimum $d$.
        end if
      end for
      Update the potential functions.
      Run BHC with remaining clusters.
      Sample $\{\xi_t\}$ and hyperparameters.
    end for
  end for
```

$\{\xi_1, \dots, \xi_t\}$ are updated, the potential functions of the tree built with $\boldsymbol{X}_{1:t}$ should also be updated. After that, we fix the auxiliary variables and rebuild the tree only for the $\boldsymbol{X}_{t+1}$ (the tree for $\boldsymbol{X}_{1:t}$ is fixed). We repeat this several times for $\boldsymbol{X}_{t+1}$, pick the best one and proceed to the next dataset $\boldsymbol{X}_{t+2}$ (**Algorithm 3**). Inserting a node into a tree requires $\mathcal{O}(k+h)$ operations, where k is the number of existing clusters before inserting the node, and h is the height of the cluster to insert. When the tree is balanced and the number of clusters is much smaller than the number of data points, IBHC is much faster than BHC.

4 EXPERIMENTS

4.1 Synthetic Data

We compared IBHC with other inference algorithms for MNRM mixtures using synthetic data. We generated 2D time-varying data from Gaussian mixtures as follows. At each time, we generated $\mathrm{Poisson}(1)$ number of new centers from $\mathcal{N}(0, 10I)$. Further, when $t \geq 2$, we chose a subset of centers generated in the past with probability 0.7. Then for each newly generated or selected center μ, we generated 100 data points from $\mathcal{N}(\mu, 1.5I)$. We repeated this process for 15 time steps to generate about 10,000 data points.

For MNRM, we set the r as even numbers between 1 and 15, and set $q_{t,r} = \exp(-0.2|t-r|)$. We used NGGP with the hyperparameters α, τ and σ. we placed a prior $\alpha \sim \mathrm{Gamma}(1,1)$; the posterior of α is again Gamma distributed so that it can be easily sampled. For σ, we placed a prior $\sigma \sim \mathrm{Beta}(1,2)$ and sampled σ via slice-sampling [24]. We fixed $\tau = 10^{-3}$. We used the same base measure for all ν_r, and used the Gaussian likelihood and the Gaussian-Wishart prior for L and H:

$$H(\mu, \Lambda) = \mathcal{N}(\mu | m, (s\Lambda)^{-1}) \mathcal{W}(\Lambda | \Psi, \nu), \quad (34)$$
$$L(\boldsymbol{x} | \mu, \Lambda) = \mathcal{N}(\boldsymbol{x} | \mu, \Lambda^{-1}), \quad (35)$$

	log-likelihood	time [s]
IBHC	-62567 $\pm$ 120	36 $\pm$ 2
Gibbs	-63970 $\pm$ 735	10611 $\pm$ 956
SMC	-64180 $\pm$ 250	2763 $\pm$ 187

Table 1: Comparisons of IBHC (25 iterations), Gibbs sampler (Gibbs, 3,000 iterations) and SMC (100 particles).

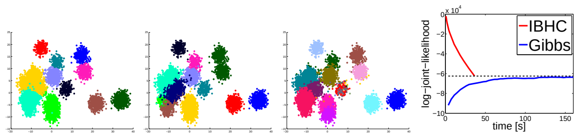

Figure 3: Clustering results of IBHC (first), Gibbs (second) and SMC (third, best viewed in color). Log-likelihood change over time of IBHC and Gibbs sampler (fourth).

where we fixed $s = 1, \nu = 6$ and set m and Λ as the empirical mean and covariance of the dataset, respectively.

We compared three algorithms; IBHC, Gibbs sampler [12] and SMC [22]. IBHC was iterated 25 times for each execution, and we sampled $\{\xi_t\}, \alpha, \sigma$ at every five iterations. Gibbs sampler was iterated 3,000 times, and SMC was run with 100 particles. IBHC was repeated 10 times, and the other two were repeated five times. We compared the maximum log-joint-likelihoods (18) and running times. All the experiments were done on an Intel Core i7 3.60GHz, 64GB RAM machine, and all the programs were written and compiled with Microsoft Visual C++ [2]. We found that IBHC achieves the best log-likelihoods in the shortest time (Table 1). Figure 3 shows an example of clustering results from using the three algorithms, and the progression of log-likelihoods over time for IBHC and Gibbs sampler. Note that the log-likelihood of IBHC decreases as new data are imported, and quickly converges to its solution.

4.2 Online Video Segmentation

As a generalization of image segmentation, video segmentation is used to segment videos into coherent segments; this can be achieved through the clustering of pixels. Video segmentation is often resolved in the 3D spatio-temporal domain to incorporate the spatio-temporal smoothness, and this is referred to as *offline video segmentation.* However, offline segmentation is impossible when the video size is very large. *Online video segmentation* is an alternative way of dividing video sequences into a certain number of chunks, and it segments the chunks incrementally.

We assumed that videos were generated from MNRM mixtures, and excuted online video segmentation via IBHC. We divided video sequences into 10-frame chunks, whereupon each chunk was over-segmented using SLIC super-

[2] The source codes will be available at `http://mlg.postech.ac.kr/~stonecold`

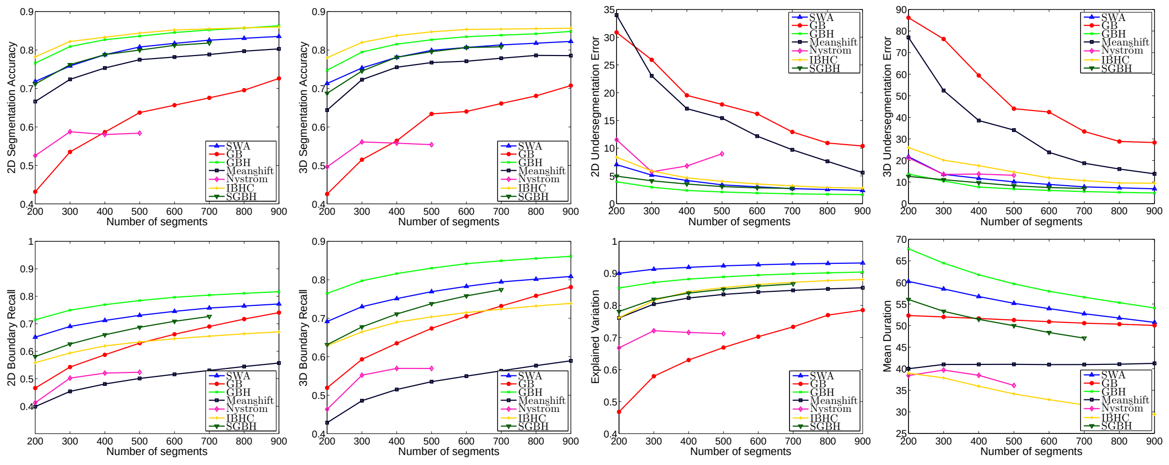

Figure 4: Comparison of video segmentation algorithms on Chen's benchmark dataset.

voxel [25][3]. Each supervoxel was represented with a 60-bin LAB color-histogram ($\boldsymbol{x}_{t,i}$), and the histograms formed the tth dataset to cluster ($\boldsymbol{X}_t$). We used the multinomial and Dirichlet prior for L and H:

$$H(\boldsymbol{\theta}) = \mathrm{Dir}(\boldsymbol{\theta}|\beta \mathbf{1}_{60}), \quad L(\boldsymbol{x}|\boldsymbol{\theta}) = \mathrm{Mult}(\boldsymbol{\theta}), \qquad (36)$$

where β is a hyperparameter and $\mathbf{1}_{60}$ is the 60 dimensional one-vector. We used the same NGGP as those outlined in section 4.1. For the spatial-smoothness, we assumed that only spatially adjacent pairs would be merged during IBHC. Instead of the dissimilarity d, we used

$$d'(c_l \cup c_r) = \begin{cases} (c_l \cup c_r) & \text{if } \mathrm{adj}(c_l, c_r) \\ \infty & \text{otherwise} \end{cases}, \qquad (37)$$

where $\mathrm{adj}(c_l, c_r) = 1$ if c_l and c_r are adjacent.

We tested IBHC on Chen's benchmark [26] using the libsvx package[4]. The benchmark contains eight sequences of $160 \times 240 \times (60 \sim 80)$ size. We evaluated the results using the performance measures proposed in [26], which contains the following eight measures: (2d/3d) boundary recall, measuring how well the boundaries of ground-truths are preserved;, (2d/3d) accuracy, measuring what fraction of ground-truths are correctly segmented;, (2d/3d) undersegmentation error, measuring what fraction of ground-truths are invaded by other segments;, explained variation, measuring the compression rate of segmentations ; and mean duration, measuring the average duration of segments. We compared IBHC with other offline and online methods. The offline methods include graph based (GB), graph-based hierarchical (GBH), segmentation by weighted aggregation (SWA), mean-shift and Nyström. The online method was streaming graph-based hierarchical (SGBH) (10 frames at once). Since these methods do not infer the number of segments, they are tested to generate multiple results with different numbers of segments. For IBHC, we adjusted the hyperparameter β from 1 to 30 to generate multiple results.[5] The results are shown in Figure 4.

IBHC showed the best (2d/3d) accuracy, even if it was an online method. It also outperformed most of the offline methods in other performance measures, but showed poor mean-duration and boundary recalls. Compared to the online method (SGBH), IBHC was better in (2d/3d) accuracy and explained variations. In summary, IBHC was comparable to the state-of-the-art algorithms without much application-specific tuning.

5 CONCLUSIONS

In this paper we have presented an incremental approximate inference, referred to as IBHC, for MNRM mixture models, which can be applied to the task of clustering of time-varying data where the assumption of exchangeability is violated. We have first described the extension of BHC (which was originally developed for DPMs) to MNRM mixture models, and then developed its incremental algorithm, IBHC, where the tree structure is partially updated when a new node comes in. Experiments on both synthetic and real-world datasets demonstrated the validity and efficiency of IBHC, compared to MCMC methods.

Acknowledgements: This work was supported by National Research Foundation (NRF) of Korea (NRF-2013R1A2A2A01067464), NIPA-MSRA Creative IT/SW Research Project, and POSTECH Rising Star Program.

[3]See `http://ivrg.epfl.ch/supplementary_material/RK_SLICSuperpixels`.

[4]See `http://www.cse.buffalo.edu/~jcorso/r/supervoxels/`

[5]See [26] for more detailed explanations and references for performance measures and comparing methods.

References

[1] L. F. James, A. Lijoi, and I. Prünster. Posterior analysis for normalized random measures with independent increments. *Scandinavian Journal of Statistics*, 36(1):76–97, 2009.

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

[3] A. Brix. Generalized Gamma measures and shot-noise Cox processes. *Advances in Applied Probability*, 31:929–953, 1999.

[4] C. E. Antoniak. Mixtures of Dirichlet processes with applications to Bayesian nonparametric problems. *The Annals of Statistics*, 2(6):1152–1174, 1974.

[5] D. Aldous. Exchangeability and related topics. *École d'Été de Probabilités de Saint-Flour XIII-1983*, pages 1–198, 1985.

[6] S. N. MacEachern. Dependent nonparametric processes. In *Proceedings of the Section on Bayesian Statistical Science*, 1999.

[7] J. E. Griffin and M. F. J. Steel. Order-based dependent Dirichlet processes. *Journal of the American Statistical Association*, 101(473):179–194, 2006.

[8] F. Caron, M. Davy, and A. Doucet. Generalized Polya urn for time-varying Dirichlet process mixtures. In *Proceedings of the Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, 2007.

[9] V. Rao and Y. W. Teh. Spatial normalized Gamma processes. In *Advances in Neural Information Processing Systems (NIPS)*, volume 22. MIT Press, 2009.

[10] D. Lin, E. Grimson, and J. Fisher. Construction of dependent Dirichlet processes based on Poisson processes. In *Advances in Neural Information Processing Systems (NIPS)*, volume 23. MIT Press, 2010.

[11] C. Chen, N. Ding, and W. Buntine. Dependent hierarchical normalized random measures for dynamic topic modeling. In *Proceedings of the International Conference on Machine Learning (ICML)*, Edinburgh, UK, 2012.

[12] C. Chen, V. Rao, W. Buntine, and Y. W. Teh. Dependent normalized random measures. In *Proceedings of the International Conference on Machine Learning (ICML)*, Atlanta, Georgia, USA, 2013.

[13] N. J. Foti, J. D. Futoma, D. N. Rockmore, and S. A. Williamson. A unifying representation for a class of dependent random measures. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, Scottsdale, AZ, USA, 2013.

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

[15] 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, 2000.

[16] D. M. Blei, A. Ng, and M. I. Jordan. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[17] K. A. Heller and Z. Ghahrahmani. Bayesian hierarchical clustering. In *Proceedings of the International Conference on Machine Learning (ICML)*, Bonn, Germany, 2005.

[18] E. Regazzini, A. Lijoi, and I. Prünster. Distriubtional results for means of normalized random measures with independent increments. *The Annals of Statistics*, 31(2):560–585, 2003.

[19] J. F. C. Kingman. Completely random measures. *Pacific Journal of Mathematics*, 21(1):59–78, 1967.

[20] L. F. James. Bayesian Poisson process partition calculus with an application to Bayesian Lévy moving averages. *The Annals of Statistics*, 33(4):1771–1799, 2005.

[21] S. Favaro and Y. W. Teh. MCMC for normalized random measure mixture models. *Statistical Science*, 28(3):335–359, 2013.

[22] J. E. Griffin. Sequential Monte Carlo methods for normalized random measure with independent increments mixtures, 2011. Preprint.

[23] H. M. Wallach, S. T. Jensen, L. Dicker, and K. A. Heller. An alternative prior process for nonparametric Bayesian clustering. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, Sardinia, Italy, 2010.

[24] R. M. Neal. Slice sampling. *The Annals of Statistics*, 31(3):705–767, 2003.

[25] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fuaa, and S. Süsstrunk. SLIC superpixels compared to state-of-the-art superpixel methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2274–2282, 2012.

[26] C. Xu and J. J. Corso. Evaluation of super-voxel methods for early video processing. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, Providence, Rhode Island, USA, 2012.