The Binary Space Partitioning-Tree Process

Xuhui Fan
School of Mathematics and Statistics
University of New South Wales
xuhui.fan@unsw.edu.au

Bin Li
School of Computer Science
Fudan University
libin@fudan.edu.cn

Scott A. Sisson
School of Mathematics and Statistics
University of New South Wales
scott.sisson@unsw.edu.au

Abstract

The Mondrian process represents an elegant and powerful approach for space partition modelling. However, as it restricts the partitions to be axis-aligned, its modelling flexibility is limited. In this work, we propose a self-consistent Binary Space Partitioning (BSP)-Tree process to generalize the Mondrian process. The BSP-Tree process is an almost surely right continuous Markov jump process that allows uniformly distributed oblique cuts in a two-dimensional convex polygon. The BSP-Tree process can also be extended using a non-uniform probability measure to generate direction differentiated cuts. The process is also self-consistent, maintaining distributional invariance under a restricted subdomain. We use Conditional-Sequential Monte Carlo for inference using the tree structure as the high-dimensional variable. The BSP-Tree process's performance on synthetic data partitioning and relational modelling demonstrates clear inferential improvements over the standard Mondrian process and other related methods.

1 Introduction

In machine learning, some tasks such as constructing decision trees or relational modelling may be interpreted as a space partitioning strategy to identify regular regions in a product space. Models may then be fitted to the data in each regular "block", whereby the data within each block will exhibit certain types of homogeneity. While applications range from relational modeling [8, 1], community detection [19, 7], collaborative filtering [21, 15], and random forests [11], most of the work only focuses on regular grids (rectangular blocks). While some recent work [25, 3] has introduced more flexibility in the partitioning strategy, there are still substantial limitations in the available modelling and inferential capabilities.

An inability to capture complex dependencies between different dimensions is one such limitation. Axis-aligned cuts, as extended from rectangular blocks, consider each division on one dimension only. This is an over-simplified and invalid assumption in many scenarios. For example, in decision tree classification, classifying data through a combination of features (dimensions in our case) is usually more efficient than a recursive single features division. Similarly, in relational data modelling, where the nodes (individuals) correspond to the dimension and the blocks correspond to the communities the nodes belong to. An individual may have asymmetric relations in the communities (e.g., a football match organizer would pay more attention to the community of football while a random participant might be less focused). While the recently proposed Ostomachion Process (OP) [3] also tries to allow oblique cuts, several important properties (e.g. self-consistency, uniform generalization) are missing, thereby limiting its appeal.

To systematically address these issues, we propose a Binary Space Partitioning (BSP)-Tree process to hierarchically partition the space. The BSP-Tree process is an almost surely right continuous Markov jump process in a budget line. Except for some isolated points in the line, any realization of the BSP-Tree process maintains a constant partition between these consecutive points. Instead of axis-aligned cuts, the partition is formed by a series of consecutive oblique cuts. In this sense, the BSP-Tree process can simultaneously capture more complex partition structures with inter-dimensional dependence. The proposed cut described by the generative process can be proved to be uniformly distributed and moreover, the measure over the cut lines is fixed to a scaled sum of the blocks' perimeters.

Further, a particular form of non-uniformly distributed oblique cuts can be obtained by imposing weight functions for the cut directions. This variant can be well suited to cases where the directions of the cut are differentially favored. This variant might be particularly suitable for cer-

Proceedings of the 21^{st} International Conference on Artificial Intelligence and Statistics (AISTATS) 2018, Lanzarote, Spain. PMLR: Volume 84.

tain modelling scenarios. For example, while ensuring the existence of oblique cuts, axis-aligned cuts would be favored more in social networks since some communities would need to have rectangular blocks). All variants of the BSP-Tree process can be proved to be self-consistent, which ensures distributional invariance while restricting the process from a larger domain to a smaller one.

The partition of the BSP-Tree process is inferred through the Conditional-Sequential Monte Carlo (C-SMC) sampler [10, 12]. In particular, we use a tree structure for the blocks to mimic the high-dimensional variables in the C-SMC sampler, where each dimension corresponds to one cut on the existing partition. Its performance is validated in space partition applications on toy data and in relational modelling. The BSP-Tree process provides clear performance gains over all competing methods.

2 Related Work

Stochastic partition processes aim to divide a product space into meaningful blocks. A popular application of such processes is modelling relational data whereby the interactions within each block tend to be homogeneous. For state-of-the-art stochastic partition processes, partitioning strategies vary, including regular-grids [8], hierarchical partitions [25, 24] and entry-by-entry strategies [18].

A regular-grid stochastic partition process constitutes separate partition processes on each dimension of the multi-dimensional array. The resulting orthogonal interactions between two dimensions will exhibit regular grids, which can represent interacting intensities. Typical regular-grid partition models include the infinite relational model (IRM) [8] and the overlapping communities extension of mixed-membership stochastic blockmodels [1]. Regular-grid partition models are widely used in real-world applications for modeling graph data [6, 26, 15].

The Mondrian process (MP) [25, 23] and its variant the Ostomachion process (OP) [3], can produce hierarchical partitions on a product space. The MP recursively generates axis-aligned cuts on a unit hypercube and partitions the space in a hierarchical fashion known as the kd-tree ([24] also considers a tree-consistent partition model, but it is not a Bayesian nonparametric model). While using similar hierarchical partition structures, the OP additionally allows oblique cuts for flexible partitions, however it does not guarantee the important self-consistency property.

3 The BSP-Tree Process

In the Binary Space Partitioning (BSP)-Tree process, we aim to generate partitions $\boxplus$ on an arbitrary two-dimensional convex polygon $\square$. The partitioning result $\boxplus$ can be represented as a set of blocks $\boxplus = \{\{\square^{(k)}\}^{k\in\mathbb{N}^+} : \cup_k \square^{(k)} = \square, \square^{(k')} \cap \square^{(k'')} = \emptyset, \forall k' \neq k''\}$. These blocks are generated by a series of cuts, which recursively divide one of the existing blocks into two new blocks. As a result, these recursive divisions organize the blocks in the manner as a Binary Space Partitioning tree structure, from which the process name is derived.

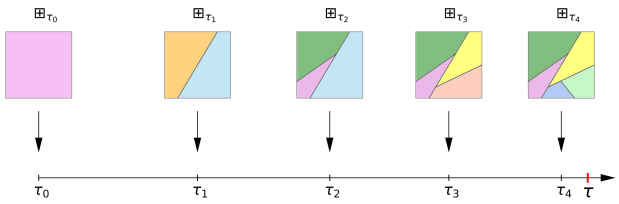

Figure 1: A realization of the BSP-Tree process in $(0, \tau]$.

We first use a pre-fixed budget τ as a termination point in an index line of $(0, \tau]$. The BSP-Tree process is defined as an almost surely right continuous Markov jump process in $(0, \tau]$. Except for some isolated time points (corresponding to the locations of cuts in $(0, \tau]$), the values (partitions) taken in any realization of the BSP-Tree process are constant between these consecutive points (cuts). Let $\{\tau_l\}^l$ denote the locations of these time points in $(0, \tau]$ and $\boxplus_t$ denote the partition at time t. We then have $\boxplus_t = \boxplus_{\tau_l}, \forall t : \tau_l \leq t < \tau_{l+1}$. More precisely, $\boxplus_t$ lies in a measurable state space $(\mathcal{S}, \Sigma)$, where $\mathcal{S}$ refers to the set of all potential Binary Space Partitions and Σ refers to the σ-algebra over the elements of $\mathcal{S}$. Thus, the BSP-Tree process can be interpreted as a $\mathcal{S}^{(0,\tau]}$-valued random variable, where $\mathcal{S}^{(0,\tau]}$ is the space of all possible partitions of $t \in (0, \tau]$ that map from the index line $(0, \tau]$ into the space $\mathcal{S}$. Figure 1 displays a sample function in $\mathcal{S}^{(0,\tau]}$.

As well as the partition at τ_{l-1}, the incremental time for the l-th cut also conditions only on the previous partitions at τ_{l-1}. Given an existing partition at τ_{l-1}, the time to the next cut, $\tau_l - \tau_{l-1}$, follows an Exponential distribution:

$$(\tau_l - \tau_{l-1})|\boxplus_{\tau_{l-1}} \sim \text{Exp}(\sum_{k=1}^{l} PE(\square^{(k)}_{\tau_{l-1}})) \qquad (1)$$

where $PE(\square^{(k)}_{\tau_{l-1}})$ denotes the perimeter of the k-th block in partition $\boxplus_{\tau_{l-1}}$. Each cut divides one block (the block is chosen with probabilities in proportion to their perimeters) into two new blocks and forms a new partition. If the index of the location τ_l of the new cut exceeds the budget τ, the BSP-Tree process terminates and returns the current partition $\boxplus_{\tau_{l-1}}$ as the final realization.

3.1 Generation of the cut line

In each block $\square^{(k)}$ in the partition, the BSP-Tree process defines cuts as straight lines cross through $\square^{(k)}$. This is achieved by generating a random auxiliary line $\boldsymbol{l}(\theta)$ with

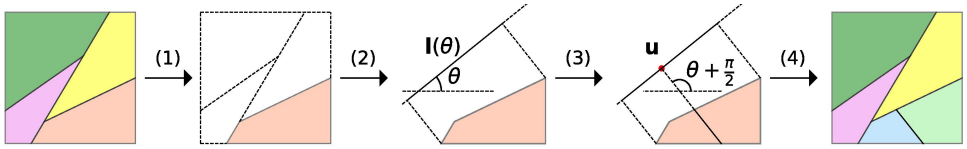

Figure 2: Given the partition $\boxplus_{\tau_3}$, the generative process for the cut line $L(\theta, \boldsymbol{u})$ at the location of τ_4.

direction θ, onto which the block $\square^{(k)}$ is projected, uniformly selecting a cut position $\boldsymbol{u}$ on the projected segment, and then constructing the cut as the line orthogonal to $\boldsymbol{l}(\theta)$ which passes through $\boldsymbol{u}$. In detail, given the partition $\boxplus_{\tau_{l-1}}$ at time τ_{l-1}, the generative process of the line for the l-th cut is:

(1) Sample a candidate block $\square^*$ from the existing blocks $\{\square^{(k)}_{\tau_{l-1}}\}^{k=1,\dots,l}$, with probabilities in proportion to the perimeters (refer to Proposition 1) of these existing blocks (Figure 2 -(1));

(2) Sample a direction θ from $(0, \pi]$, where the probability density function is in proportion to the length of the line segment $\boldsymbol{l}(\theta)$, onto which $\square^*$ is projected in the direction of θ[1]. (Figure 2 -(2));

(3) Sample the cutting position $\boldsymbol{u}$ uniformly on the line segment $\boldsymbol{l}(\theta)$. The proposed cut is formed as the straight line passing through $\boldsymbol{u}$ and crossing through the block $\square^*$, orthogonal to $\boldsymbol{l}(\theta)$ (Figure 2 -(3)).

(4) Sample the incremental time for the proposed cut according to Eq. (1). If $\tau_l > \tau$, reject the cut and return $\{\square^{(k)}_{\tau_{l-1}}\}^{k=1,\dots,l}$ as the final partition structure; otherwise accept the proposed cut, increment l to $l+1$ and go back to Step (1) (Figure 2 -(4)).

It is clear that the blocks generated in this process are convex polygons. Step (4) determines that the whole process terminates only if the accumulated cut cost exceeds the budget, τ. However, notice that $\tau_l \to \infty$ as $l \to \infty$ almost surely (justification in Supplementary Material A). This means an infinite number of cuts would require an infinite budget almost surely. I.e., this block splitting process terminates to any finite budget τ with probability one.

In the following, we analyze the generative process for proposing the cut. For reading convenience, we first consider the case of cutting on a sample block $\square$ (a convex polygon). Its extension to the whole partition $\boxplus$ (i.e., a set of blocks $\{\square^{(k)}\}$) is then straightforward.

[1]This can be achieved using rejection sampling. We use the uniform distribution $g(\theta) = 1/\pi$ over $(0, \pi]$. The scaling parameter is determined as $M = \pi/2$, which guarantees a tight upper bound such that $(l(\theta)/PE(\square))/(M * g(\theta)) <= 1$. The expected number of sampling times is then $\pi/2$

3.2 Cut definitions and notations

From step (3) in the generative process, the cut line is defined as a set of points in the block $L(\theta, \boldsymbol{u}) := \{\boldsymbol{x} \in \square | (\boldsymbol{x} - \boldsymbol{u})^\top \cdot (1; \tan\theta) = 0\}$. The set of cut lines for all the potential cuts crossing into block $\square$ can be subsequently denoted as $C_\square = \{L(\theta, \boldsymbol{u}) | \theta \in [0, \pi), \boldsymbol{u}$ lies on the line segment $\boldsymbol{l}(\theta)\}$.

Each of the element in $C_\square$ corresponds to a partition on the block $\square$. This is described by a one-to-one mapping ϕ: $C_\square \to T_\square$, where $T_\square$ denotes the set of one-cut partitions on the block $\square$. The measures over $C_\square$ and $T_\square$ are described as: $\overline{\nu}_\square(T_\square) := \overline{\nu}_\square \circ \phi(C_\square) = \lambda_\square(C_\square)$, where $\overline{\nu}_\square(\cdot)$ denotes the normalized probability measure on $T_\square$ and $\lambda_\square(C_\square)$ denotes the probability measure on $C_\square$.

The direction θ and the cut position $\boldsymbol{u}$ are sequentially sampled in steps (2) and (3) in the generative process, where $\boldsymbol{u}$ is located on the image of the polygon projected onto the line $\boldsymbol{l}(\theta)$. For step (2), we denote by $C^\theta_\square = \{L(\theta, \boldsymbol{u}) | \theta$ is fixed, $\boldsymbol{u}$ lies on the line segment$\}$ the set of all cut lines with fixed direction θ and by $\lambda_\square(C^\theta_\square)$ the associated probability measure. In Step (3), we use $\lambda_\square(L(\theta, \boldsymbol{u}) | C^\theta_\square)$ to denote the conditional probability measure on the line $L(\theta, \boldsymbol{u})$ given direction θ.

3.3 Uniformly distributed cut lines $L(\theta, \boldsymbol{u})$

It is easy to demonstrate that the above strategy produces a cut line $L(\theta, \boldsymbol{u})$ that is uniformly distributed on $C_\square$.

Marginal probability measure $\lambda_\square(C^\theta_\square)$: We restrict the marginal probability measure $\lambda_\square(C^\theta_\square)$ of $C^\theta_\square$ to the family of functions that remains invariant under the following three operations on the block $\square$ (their mathematical definitions are provided in Supplementary Material B):

1. Translation t: $\lambda_\square(C^\theta_\square) = \lambda_{t_{\boldsymbol{v}}\square} \circ t_{\boldsymbol{v}}(C^\theta_\square)$, where $t_{\boldsymbol{v}}(\cdot)$ denotes incrementing the set of points by a vector $\boldsymbol{v}, \forall \boldsymbol{v} \in \mathbb{R}^2$;
2. Rotation r: $\lambda_\square(C^\theta_\square) = \lambda_{r_{\theta'}\square} \circ r_{\theta'}(C^\theta_\square)$, where $r_{\theta'}(\cdot)$ denotes rotating the set of points by an angle $\theta', \forall \theta' \in [0, \pi)$;
3. Restriction ψ: $\lambda_\square(C^\theta_\triangle) = \lambda_{\psi_\triangle\square} \circ \psi_\triangle(C^\theta_\triangle)$, where $\triangle \subseteq \square$ refers to a sub-region of $\square$; $\psi_\triangle(\cdot)$ retains the set of points in $\triangle$, and $C^\theta_\triangle$ refers to the set of cut lines (orthogonal to $\boldsymbol{l}(\theta)$) that cross through $\triangle$.

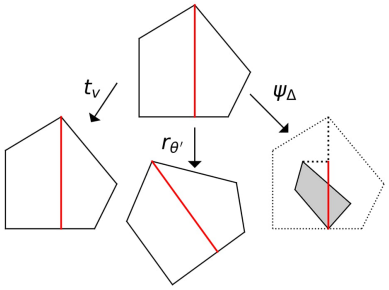

Figure 3: $|\boldsymbol{l}(\theta)|$ remains invariant under the operations of translation ($t_{\boldsymbol{v}}$), rotation ($r_{\theta'}$) and restriction ($\psi_\triangle$).

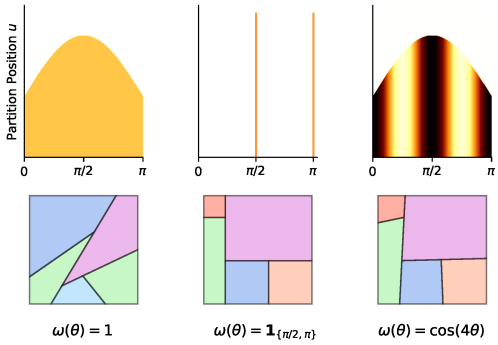

Figure 4: Various settings of $\omega(\theta)$ and example partitions.

Define a set function as $f_\square(C^\theta_\square) = |\boldsymbol{l}_\square(\theta)|$, where $|\boldsymbol{l}_\square(\theta)|$ is the length of an image of the polygon $\square$ projected onto the line with direction θ. It is clear that $f_\square(C^\theta_\square)$ remains invariant under the first two operations. For the restriction ψ, we have $f_\square(C^\theta_\triangle) = f_\square(\{L(\theta, \boldsymbol{u}|\boldsymbol{u} \text{ lies on } \boldsymbol{l}_{\psi_\triangle(\square)}(\theta))\}) = |\boldsymbol{l}_\triangle(\theta)| = f_\triangle(\{L(\theta, \boldsymbol{u}|\boldsymbol{u} \text{ lies on } \boldsymbol{l}_{\psi_\triangle(\square)}(\theta))\})$. A visualization can be seen in Figure 3. Further, the following result shows $\lambda_\square\left(C^\theta_\square\right)$ is fixed to a scaled form of $f_\square(C^\theta_\square)$ (proof in Supplementary Material C):

Proposition 1 *The family of functions $\lambda_\square\left(C^\theta_\square\right)$ remains invariant under the operations of translation, rotation and restriction if and only if there is a constant C such that $\lambda_\square\left(C^\theta_\square\right) = C \cdot f_\square(C^\theta_\square) = C \cdot |\boldsymbol{l}(\theta)|, \forall C \in \mathbb{R}^+$.*

Following Proposition 1, the measure of all cut lines on the whole block $\square$ may be obtained by integrating out the direction θ. This produces the result that the measure over the cuts on the block is fixed to the scale of the perimeter of the block (proof in Supplementary Material D):

Proposition 2 *Assuming the direction θ has a distribution of $|\boldsymbol{l}(\theta)|/\int_0^\pi |\boldsymbol{l}(\theta)|d\theta$, the BSP-Tree process has the partition measure $\lambda_\square(C_\square) = C \cdot \int_0^\pi |\boldsymbol{l}(\theta)|d\theta = C \cdot PE(\square)$.*

Combining Proposition 1 and Proposition 2, we obtain the probability measure as $\lambda_\square\left(C^\theta_\square\right) = |\boldsymbol{l}(\theta)|/PE(\square)$.

Conditional probability $\lambda_\square\left(L(\theta, \boldsymbol{u})|C^\theta_\square\right)$: Given the direction θ, define the conditional probability $\lambda_\square\left(L(\theta, \boldsymbol{u})|C^\theta_\square\right)$ as a uniform distribution over $\boldsymbol{l}(\theta)$. That is: $\lambda_\square\left(L(\theta, \boldsymbol{u})|C^\theta_\square\right) = \frac{1}{|\boldsymbol{l}(\theta)|}$.

As a result, the uniform distribution of $L(\theta, \boldsymbol{u})$ can be obtained as $\lambda_\square\left(L(\theta, \boldsymbol{u})\right) = \lambda_\square\left(C^\theta_\square\right) \cdot \lambda_\square\left(L(\theta, \boldsymbol{u})|C^\theta_\square\right) = 1/PE(\square)$. Further, $\lambda_\square\left(L(\theta, \boldsymbol{u})\right) = \overline{\nu}_\square \circ \phi\left(L(\theta, \boldsymbol{u})\right) = \overline{\nu}_\square\left(\boxplus^*\right)$, where $\boxplus^* \in T_\square$. That is to say, the new partition over $T_\square$ is also uniformly distributed.

This uniform distribution does not require the use of particular partition priors. Without additional knowledge about the cut, all potential partitions are equally favored.

Extension to the partition $\boxplus$: The extension from the single block $\square$ to the whole partition $\boxplus = \{\square^{(k)}\}$ is completed by the exponentially distributed incremental time of $\{\tau_l^{(k)} - \tau_{l-1}^{(k)}\}^l$ for each block $\square^{(k)}$, where with an abuse of notation $\tau_l^{(k)}$ refers to the location of the l-th cut in block $\square^{(k)}$. In terms of partition $\boxplus$, the minimum incremental time for all the blocks is distributed as as $\min\{\tau_l^{(k)} - \tau_{l-1}^{(k)}\}^k \sim \text{Exp}(\sum_k PE(\square^{(k)}))$ since $P(\min\{\tau_l^{(k)} - \tau_{l-1}^{(k)}\}^k > t) = \prod_k P(\tau_l^{(k)} - \tau_{l-1}^{(k)} > t) = \exp(-t\sum_k PE(\square^{(k)}))$, which is the complementary CDF of the exponential distribution. Further, we have that $P(k^* = \arg\min_k\{\tau_l^{(k)} - \tau_{l-1}^{(k)}\}^k) = PE(\square^{(k^*)})/\sum_k PE(\square^{(k)})$, which is the probability of selecting the block to be cut. These results correspond to Step (1) and Step (4) in the generative process.

3.4 Non-uniformly distributed cut lines $L(\theta, \boldsymbol{u})$

The BSP-Tree process does not require that the cut line is uniformly distributed over $C_\square$. In some cases (e.g. in social network partitioning, where the blocks refer to communities), some blocks may tend to have regular shapes (rectangular-shaped communities with equal contributions from the nodes). In these scenarios, axis-aligned cuts would have larger contributions than any others. In the following, we generalize the uniformly distributed cuts to a particular form of non-uniformly distributed cuts, placing arbitrary weights on choosing the directions of the cut line.

Non-uniform measure This generalization is achieved by relaxing the invariance restriction on the measure of $C^\theta_\square$, under the rotation operation. We use an arbitrary non-negative finite function $\omega(\theta)$ as prior weight on the direction θ. Under the rotation $r_{\theta'}$, the probability measure $C^\theta_\square$ requires to have the form of $\omega(\theta + \theta') \cdot \lambda_\square(C^\theta_\square) = \omega(\theta) \cdot \lambda_{r_{\theta'}\square} \circ r_{\theta'}(C^\theta_\square)$. Following similar arguments as proposition 1, this implies that $\lambda_\square(C^\theta_\square) = C \cdot \omega(\theta) \cdot |\boldsymbol{l}(\theta)|$. Given the uniform conditional distribution of $\lambda_\square(L(\theta, \boldsymbol{u})|C^\theta_\square)$, the probability measure over $L(\theta, \boldsymbol{u})$ is $\lambda_\square\left(L(\theta, \boldsymbol{u})\right) \propto \omega(\theta)$. Clearly, $\lambda_\square\left(L(\theta, \boldsymbol{u})\right)$ is a non-uniform distribution

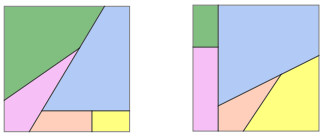

Figure 5: Example partitions on mixed measures.

and it is determined by the weight function $\omega(\theta)$.

Figure 4 displays some examples of the probability function of $\lambda_{\square}(L(\theta, \boldsymbol{u}))$ and corresponding partition visualizations under different settings of $\omega(\theta)$. The color indicates the value of the PDF at the orthogonal slope θ, with darker colors indicating larger values. While $\omega(\theta) = 1$, the cuts follows the uniform distribution; while with $\omega(\theta) = 1_{\{\pi/2,\pi\}}$, the BSP-Tree process is reduced to the Mondrian process, whereby only axis-aligned partitions are allowed; when $\omega(\theta) = \cos(4\theta)$, the weighted probability distribution adjusts the original one into shapes of stripes along the θ-axis. In each example, the colour on the same direction θ is constant. This illustrates the uniform distribution of the partition position $\boldsymbol{u}$ given the direction θ.

Mixed measure Naturally, we may use mixed distributions on θ for greater modelling specificity. If we have two different non-negative weight functions $\omega_1(\theta)$ and $\omega_2(\theta)$, the partition measure over these two sets can be written as $\lambda^1_{\square}(C^{\theta}_{\square}) = C_1 \cdot \int_{[0,\pi)} \omega_1(\theta) \cdot |l(\theta)| d\theta, \lambda^2_{\square}(C^{\theta}_{\square}) = C_2 \cdot \int_{[0,\pi)} \omega_2(\theta) \cdot |l(\theta)| d\theta$, where C_1 and C_2 are non-negative constants. For instance, let $\omega_1(\theta) = \mathbf{1}_{\{\pi/2,\pi\}}, \omega_2(\theta) = \mathbf{1}_{(0,\pi]}$. The direction θ is then sampled as:

$$\theta \sim \begin{cases} \sum_{\frac{\pi}{2},\pi} \mathbf{1}_{\theta} \frac{|\boldsymbol{l}(\theta)|}{|\boldsymbol{l}(0)|+|\boldsymbol{l}(\frac{\pi}{2})|}, & z = 1; \\ \frac{|\boldsymbol{l}(\theta)|}{PE(\square)}, & z = 0. \end{cases} \tag{2}$$

where $z \sim \text{Bernoulli}(\frac{C^1}{C^1+C^2})$, indicating which distribution for θ samples from. That is, θ is sampled either from the discrete set of $\{\pi/2, \pi\}$ or the continuous set of $(0, \pi]$. Figure 5 shows example partition visualizations based on this particular mixed measure.

3.5 Self-consistency

The BSP-Tree process is defined on a finite convex polygon. To further extend the BSP-Tree process to the infinite two-dimensional space $\mathbb{R}^2$, an essential property is self-consistency. That is, while restricting the BSP-Tree process on a finite two-dimensional convex polygon $\square$, to its sub-region $\triangle, \triangle \subseteq \square$, the resulting partitions restricted to $\triangle$ are distributed as if they are directly generated on $\triangle$ through the generative process (see Figure 6).

For both of the uniform-distributed and non-uniform distributed cut lines in the BSP-Tree process, we have the following result (justification in Supplementary Material E):

Proposition 3 *The BSP-Tree process (including the uniformly distributed and non-uniformly distributed cut lines described above) is self-consistent, which maintains distributional invariance when restricting its domain from a larger convex polygon to its belonging smaller one.*

The Kolmogorov Extension Theorem [9] ensures that the self-consistency property enables the BSP-Tree process to be defined on the infinite two-dimensional space. This property can be suited to some cases, such as online learning, domain expansion.

Algorithm 1 C-SMC for inferring $\boxplus$ at $(t+1)$-th iteration

Input: Training data X, Budget $\tau > 0$, Number of particles C, Partition sequence $\{\tau^*_l, \boxplus^*_{\tau_l}\}^l$ at the t-th iteration , Likelihood function $P(X|\boxplus)$.

Initialize partitions for each particle $P^c = \square$ and weight $\omega^c_0 = 1$, stage $l = 1$, $\tau^c_l = 0$
while $\exists c, s.t.\ \tau^c_l < \tau$ **do**
  for $c = 1 : C$ **do**
    if $\tau^c_l < \tau$ **then**
      if $c = 1$ **then**
        Set $P^1_l = \boxplus^*_{\tau_l}, \tau^1_l = \tau^*_l$
      else
        Sample P^c_l, τ^c_l according to the generative process of the cut line in Section 3.1
      end if
      Set $\omega^c_l = \omega^c_{l-1} \cdot \frac{P(X|P^c_l)}{P(X|P^c_{l-1})}$
    else
      Set $P^c_l = P^c_{l-1}, \omega^c_l = \omega^c_{l-1}, \tau^c_l = \tau^c_{l-1}$
    end if
  end for
  Normalize weights $\overline{\omega}^c_l = \frac{\omega^c_l}{W_l}, W_l = \sum_c \omega^c_l$
  Set $j_1 = 1$.
  For $c = 2 : C$, resample indices j_c from $\sum_c \overline{\omega}^c_l \delta_{P^c_l}$
  $\forall c, P^c_l = P^{j_c}_l; \omega^c_l = W_l/C; l = l + 1$
end while
Return partition at $(t+1)$-th iteration: $\boxplus \sim \sum_c \overline{\omega}^c_l \delta_{P^c_l}$

4 C-SMC Sampler for the BSP-Tree Process

Inference for the BSP-Tree type hierarchical partition process is hard, since early partitions would heavily influence subsequent blocks and consequently their partitions. That is, all later blocks must be considered while making inference about these early stage-cuts (or intra-nodes cuts from the perspective of the tree structure). Previous approaches have typically used local structure movement strategies to slowly reach the posterior structure, including the Metropolis-Hastings algorithm [25], Reversible-Jump Markov chain Monte Carlo [27, 22]. These algorithms suffer either slow mixing rates or a heavy computational cost.

To avoid this issue, we use a Conditional-Sequential Monte Carlo (C-SMC) sampler [2] to infer the structure of the

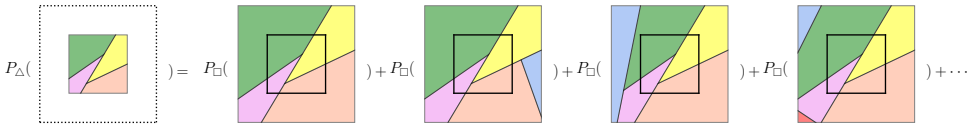

Figure 6: Self-Consistency of the BSP-Tree process.

BSP-Tree process. Here the tree structure is taken as the high-dimensional variable in the sampler. The cut line can be taken as the dimension dependence between the consecutive partition states of the particles. In this way, a completely new tree can be sampled in each iteration, rather than the local movements in the previous approaches.

Algorithm 1 displays the detailed strategy to infer the tree structure in the $(t+1)$-th iteration, given the generative partition sequence $\{\tau_l^*, \boxplus_{\tau_l}^*\}^l$ at the t-th iteration. We should note the likelihood $P(X|\boxplus)$ varies in different applications. For example, $P(X|\boxplus)$ would be a Multinomial probability in Section 5.1 and a Bernoulli probability in Section 5.2. Line 2 ensures that algorithm continues until the generative process of all particles terminates. Line 6 fixes the 1-st particle as the partition sequence in the t-th iteration in all stages. Lines $15-18$ refer to the resampling step.

5 Applications

5.1 Toy data partition visualization

Three different cases of toy data are generated as: **Case 1:** Dense uniform data on $[0,1]^2$. The labels for these points are determined by a realization of BSP-Tree process on $[0,1]^2$; **Case 2:** $1,000$ data points generated from 5 bivariate Gaussian distributions, with an additional 100 uniformly distributed noisy points; **Case 3:** same setting with case 2, however with an additional 500 uniformly distributed noisy points.

Using $\{(\xi_i, \eta_i)\}_{i=1}^N$to denote the coordinates of the data points, the data are generated as: (1) Generate an realization from the BSP-Tree process $\boxplus = \{\square^{(k)}\}^{k\in\mathbb{N}^+}$; (2) Generate the parameter of the multinomial distribution in $\square^{(k)}$, $\boldsymbol{\phi}_k \sim \text{Dirichlet}(\boldsymbol{\alpha}), \forall k \in \mathbb{N}^+$; (3) Generate the labels for the data points $z_i \sim \text{Multinomial}(\boldsymbol{\phi}_{h(\xi_i,\eta_i|\{\square^{(k)}\})}), \forall i \in \{1,\cdots,N\}$, where $h(\xi_i,\eta_i)$ is a function mapping i-th data point to the corresponding block.

For Case 1, we set $\tau = 10$; for Case 2 and 3, we set $\tau = 3$. In all these cases, the weight function $\omega(\theta)$ is set as the uniform distribution $\omega(\theta) = 1/\pi, \forall\theta \in (0,\pi]$. Figure 7 shows example partitions drawn from the posterior distribution on each dataset (we put the visualization of Case 1 in the Supplementary Material). For the dense uniform data, the partitions can identify the subtle structure, while it might be accused of more cutting budget. In the other two cases, the nominated Gaussian distributed data are well classified, regardless of the noise contamination.

5.2 Relational Modelling

5.2.1 Model Construction

A typical application for the BSP-Tree process is relational modelling [8]. The observed data in relational modelling is an asymmetric matrix $R \in \{0,1\}^{N\times N}$. The rows and columns in the matrix represent the nodes in the interaction network and an entry $R_{ij} = 1$ indicates a relation from node i to node j. Partitions over the nodes in rows and columns will jointly form the blocks in this observed matrix. It is expected that the relations within each block share homogeneity, compared to the relations between blocks.

The Aldous-Hoover representation theorem [20] indicates that these types of exchangeable relational data can be modelled by a function on the unit space $[0,1]^2$ and coordinates in the unit interval $[0,1]$. In particular, the coordinates in the unit interval $[0,1]$ represent the nodes and we use the block-wise function to denote the function on $[0,1]^2$. Here, the blocks generated in the BSP-Tree process are used to infer the communities for these relations. In this way, the generative process is constructed as: (1) Generate an realization from the BSP-Tree process $\boxplus = \{\square^{(k)}\}^{k\in\mathbb{N}^+}$; (2) Generate the parameter of the Bernoulli distribution $\phi_k \sim \text{Beta}(\alpha_0,\beta_0), \forall k \in \mathbb{N}^+$; (3) Generate the coordinates for the data points $\xi_i, \eta_j \sim \text{uniform}[0,1], \forall i,j \in \{1,\cdots,N\}$;(4) Generate the relations $R_{ij} \sim \text{Bernoulli}(\phi_{h(\xi_i,\eta_j)}), \forall i,j \in \{1,\cdots,N\}$.

The details of the inference procedure can be found in the Supplementary Material F.

5.2.2 Experiments

We compare the BSP-Tree process in relational modelling (BSP-RM) with one "flat"-partition method and one latent feature method: (1) IRM [8], which produces a regular-grid partition structure; (2) Latent Feature Relational Model (LFRM) [17], which uses latent features to represent the nodes and these features' interactions to describe the relations; (3), the Mondrian Process (MP-RM), which uses the Mondrian Process to infer the structure of communities; (4),the Matrix Tile Analysis [4] Relational Model (MTA-RM). For IRM and LFRM, we adopt the collapsed Gibbs sampling algorithms for inference [8]; we implement RJ-

Table 1: Relational modeling (link prediction) comparison results (AUC±std)

Data Sets	IRM	LFRM	MP-RM	MTA-RM	BSP-RM
Digg	0.792 ± 0.011	0.801 ± 0.031	0.784 ± 0.020	0.793 ± 0.005	$\mathbf{0.820} \pm 0.016$
Flickr	0.870 ± 0.003	0.881 ± 0.006	0.868 ± 0.011	0.872 ± 0.004	$\mathbf{0.929} \pm 0.015$
Gplus	0.857 ± 0.002	0.860 ± 0.008	0.855 ± 0.007	0.857 ± 0.002	$\mathbf{0.885} \pm 0.017$
Facebook	0.872 ± 0.013	0.881 ± 0.023	0.876 ± 0.028	0.885 ± 0.010	$\mathbf{0.931} \pm 0.020$
Twitter	0.860 ± 0.003	0.868 ± 0.021	0.815 ± 0.055	0.870 ± 0.006	$\mathbf{0.896} \pm 0.008$

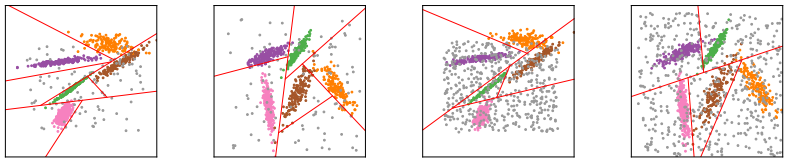

Figure 7: Toy Data Partition Visualization (left two: Case 2; right two: Case 3).

MCMC [5, 27] for the Mondrian Process and the Iterative Condition Modes algorithm [4] for MTA-RM.

Datasets. We use 5 social network datasets, Digg, Flickr [28], Gplus [16], and Facebook, Twitter [14]. We extract a subset of nodes from each network dataset: We select the top $1,000$ active nodes based on their interactions with others; then randomly sample 500 from these $1,000$ nodes for constructing the relational data matrix.

Experimental Setting. In IRM, we let α be sampled from a gamma prior $\Gamma(1,1)$ and the row and column partitions be sampled from two independent Dirichlet processes; In LFRM, we let α be sampled from a gamma prior $\Gamma(2,1)$. As the budget parameter of MP-RM is hard to sample [13], we set it to 3, which suggests that around $(3+1)\times(3+1)$ blocks would be generated. For parametric model MTA-RM, we simply set the number of tiles to 16. For the BSP-Tree process, we set the the budget τ as 8, which is the same as MP, and $\omega(\theta)$ to be the form of the mixed distribution Eq. (2). We compare the results in terms of training Log-likelihood (community detection) and testing AUC (link prediction). The reported performance is averaged over 10 randomly selected hold-out test sets (Train : Test $= 9:1$).

Experimental Results. Table 1 presents the performance comparison results on these datasets. As can be seen, the BSP-Tree process with the C-SMC strategy clearly perform better than the comparison models. The AUC link prediction score is improved by around $3\% \sim 5\%$.

Figure 8 (rows 1-5) illustrates the sample partitions drawn from the resulting posteriors. The partition from the BSP-Tree process looks to capture dense irregular blocks and smaller numbers of sparse regions, showing the efficiency of the oblique cuts. While the two representative cutting-based methods, IRM and MP-RM, cut sparse regions into bigger number of blocks. Another observation is that regular and irregular blocks co-exist in Flickr and Facebook under BSP-RM. Thus, in addition to improved performance, BSP-RM also produces a more efficient partition strategy.

Figure 8 (rows 6-7) plots the average performance versus wall-clock time for two measures of performance. IRM and LFRM converge fastest because of efficient collapsed Gibbs sampling. MTA-RM also converges fast because it is trained using a simple iterative algorithm. Although BSP-RM takes a bit longer time to converge, it is clear that it ultimately produces the best performance against other methods in terms of both AUC value and training log-likelihood.

6 Conclusions

The BSP-Tree process is a generalization of the Mondrian process that allows oblique cuts within space partition models and also provably maintains the important self-consistency property. It incorporates a general (non-uniform) weighting function $\omega(\theta)$, allowing for flexible modelling of the direction of the cuts, with the axis-aligned only cuts of the Mondrian process given as a special case. Experimental results on both toy data and real-world relational data shows clear inferential improvements over the Mondrian process and other related methods.

Two aspects worth further investigation: (1) learning the budget τ to make the number of partitions better fit to the data; (2) learning the weight function $\omega(\theta)$ within the C-SMC algorithm to improve algorithm efficiency.

7 Acknowledgement

Xuhui Fan and Scott Anthony Sisson are supported by the Australian Research Council through the Australian Centre of Excellence in Mathematical and Statistical Frontiers (ACEMS, CE140100049), and Scott Anthony Sisson through the Discovery Project Scheme (DP160102544). Bin Li is supported by the Fudan University Startup Research Grant (SXH2301005) and Shanghai Municipal Science & Technology Commission (16JC1420401).

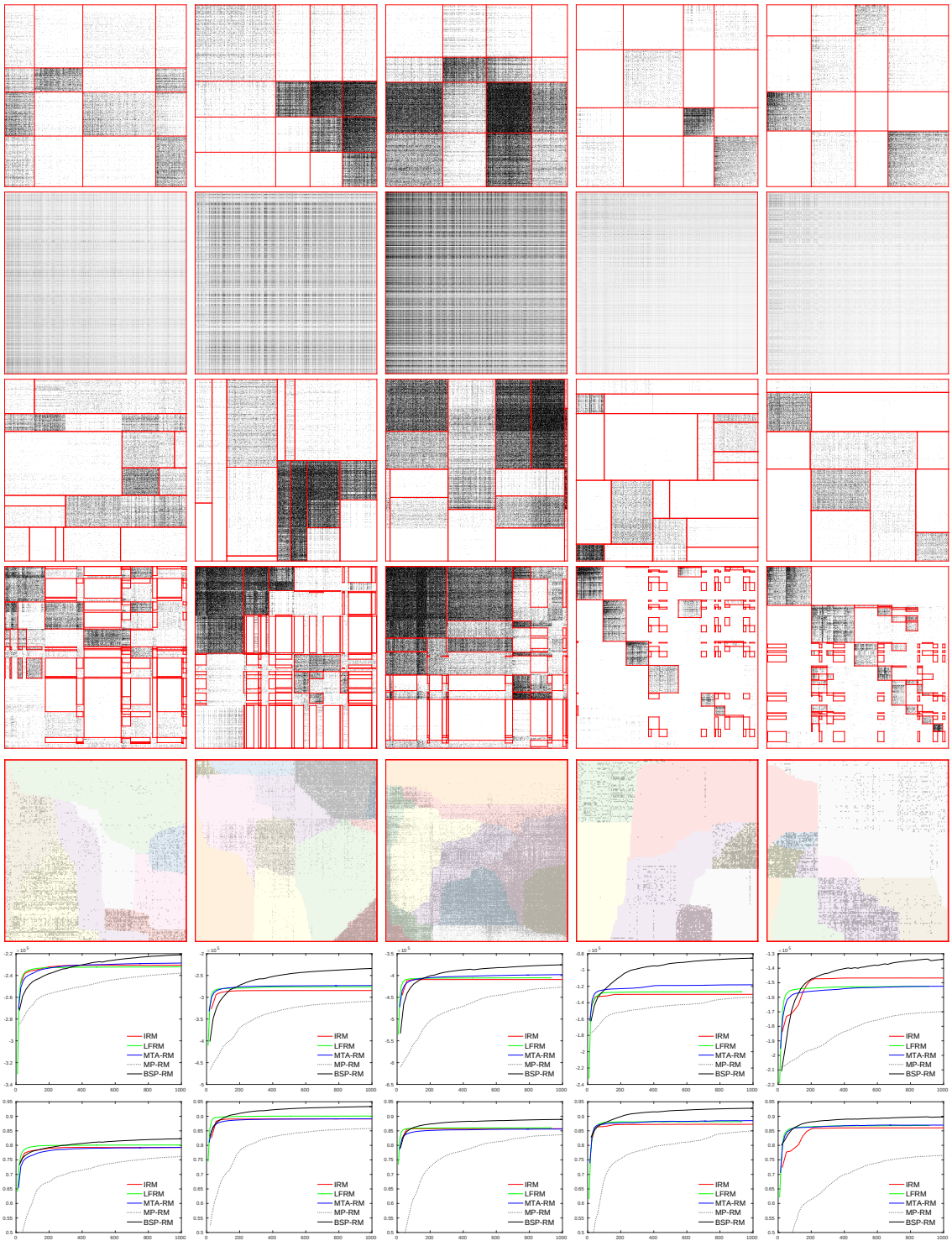

Figure 8: Partition structure visualization and performance comparison on the five data sets: (from left to right) Digg, Flickr, Gplus, Facebook and Twitter. The rows correspond to (from top to bottom) (1) IRM, (2) LFRM (refer to trained densities for each entry in the relational data), (3) MP-RM, (4) MTA-RM, (5) BSP-RM, (6) training log-likelihood vs. wall-clock time (s) and (7) testing AUC vs. wall-clock time (s). In BSP-RM, the colors behind the data points refer to the blocks and the cut lines are formed as curved lines rather than straight lines, since we are ranking the data points based on their coordinates and displaying this permutated relational matrix.

References

[1] Edoardo M. Airoldi, David M. Blei, Stephen E. Fienberg, and Eric P. Xing. Mixed membership stochastic blockmodels. In *NIPS*, pages 33–40, 2009.

[2] Christophe Andrieu, Arnaud Doucet, and Roman Holenstein. Particle Markov chain Monte Carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72(3):269–342, 2010.

[3] Xuhui Fan, Bin Li, Yi Wang, Yang Wang, and Fang Chen. The Ostomachion Process. In *AAAI*, pages 1547–1553, 2016.

[4] Inmar Givoni, Vincent Cheung, and Brendan Frey. Matrix tile analysis. In *UAI*, pages 200–207, 2006.

[5] Peter J. Green. Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika*, 82(4):711–732, 1995.

[6] Katsuhiko Ishiguro, Tomoharu Iwata, Naonori Ueda, and Joshua B. Tenenbaum. Dynamic infinite relational model for time-varying relational data analysis. In *NIPS*, pages 919–927, 2010.

[7] Brian Karrer and Mark E.J. Newman. Stochastic blockmodels and community structure in networks. *Physical Review E*, 83(1):016107, 2011.

[8] Charles Kemp, Joshua B. Tenenbaum, Thomas L. Griffiths, Takeshi Yamada, and Naonori Ueda. Learning systems of concepts with an infinite relational model. In *AAAI*, volume 3, pages 381–388, 2006.

[9] Kai lai Chung. *A Course in Probability Theory*. Academic Press, 2001.

[10] Balaji Lakshminarayanan, Daniel M. Roy, and Yee Whye Teh. Top-down particle filtering for Bayesian decision trees. In *ICML*, pages 280–288, 2013.

[11] Balaji Lakshminarayanan, Daniel M. Roy, and Yee Whye Teh. Mondrian forests: Efficient online random forests. In *NIPS*, pages 3140–3148, 2014.

[12] Balaji Lakshminarayanan, Daniel M. Roy, and Yee Whye Teh. Particle Gibbs for Bayesian additive regression trees. In *AISTATS*, pages 553–561, 2015.

[13] Balaji Lakshminarayanan, Daniel M. Roy, and Yee Whye Teh. Mondrian forests for large-scale regression when uncertainty matters. In *AISTATS*, pages 1478–1487, 2016.

[14] Jure Leskovec, Daniel Huttenlocher, and Jon Kleinberg. Predicting positive and negative links in online social networks. In *WWW*, pages 641–650, 2010.

[15] Bin Li, Qiang Yang, and Xiangyang Xue. Transfer learning for collaborative filtering via a rating-matrix generative model. In *ICML*, pages 617–624, 2009.

[16] Julian J. McAuley and Jure Leskovec. Learning to discover social circles in ego networks. In *NIPS*, volume 2012, pages 548–556, 2012.

[17] Kurt Miller, Michael I. Jordan, and Thomas L. Griffiths. Nonparametric latent feature models for link prediction. In *NIPS*, pages 1276–1284, 2009.

[18] Masahiro Nakano, Katsuhiko Ishiguro, Akisato Kimura, Takeshi Yamada, and Naonori Ueda. Rectangular tiling process. In *ICML*, pages 361–369, 2014.

[19] Krzysztof Nowicki and Tom A.B. Snijders. Estimation and prediction for stochastic block structures. *Journal of the American Statistical Association*, 96(455):1077–1087, 2001.

[20] Peter Orbanz and Daniel M. Roy. Bayesian models of graphs, arrays and other exchangeable random structures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(02):437–461, 2015.

[21] Ian Porteous, Evgeniy Bart, and Max Welling. Multi-HDP: A non parametric Bayesian model for tensor factorization. In *AAAI*, pages 1487–1490, 2008.

[22] Matthew T. Pratola. Efficient MetropolisâĂŞHastings Proposal Mechanisms for Bayesian Regression Tree Models. *Bayesian Analysis*, 11:885–911, 2016.

[23] Daniel M. Roy. *Computability, Inference and Modeling in Probabilistic Programming*. PhD thesis, MIT, 2011.

[24] Daniel M. Roy, Charles Kemp, Vikash Mansinghka, and Joshua B. Tenenbaum. Learning annotated hierarchies from relational data. In *NIPS*, pages 1185–1192, 2007.

[25] Daniel M. Roy and Yee Whye Teh. The Mondrian process. In *NIPS*, pages 1377–1384, 2009.

[26] Mikkel N. Schmidt and Morten Mørup. Nonparametric Bayesian modeling of complex networks: An introduction. *IEEE Signal Processing Magazine*, 30(3):110–128, 2013.

[27] Pu Wang, Kathryn B. Laskey, Carlotta Domeniconi, and Michael I. Jordan. Nonparametric Bayesian co-clustering ensembles. In *SDM*, pages 331–342, 2011.

[28] Reza Zafarani and Huan Liu. Social computing data repository at ASU, 2009.