Published in final edited form as:

J Comput Graph Stat. 2011 March 1; 20(1): 41–62. doi:10.1198/jcgs.2010.09115.

The Polya Tree Sampler: Towards Efficient and Automatic Independent Metropolis-Hastings Proposals

Timothy E. Hanson[Associate Professor],

Department of Statistics, University of South Carolina, Columbia, SC 29208

João V. D. Monteiro[Research Assistant], and

Division of Biostatistics, School of Public Health, University of Minnesota, Minneapolis, MN 55455

Alejandro Jara[Assistant Professor]

Department of Statistics, Pontificia Universidad Católica de Chile, Santiago, Chile

Timothy E. Hanson: hansomt@stat.sc.edu; João V. D. Monteiro: monte092@umn.edu; Alejandro Jara: atjara@uc.cl

Abstract

We present a simple, efficient, and computationally cheap sampling method for exploring an unnormalized multivariate density on $\mathbb{R}^d$, such as a posterior density, called the Polya tree sampler. The algorithm constructs an independent proposal based on an approximation of the target density. The approximation is built from a set of (initial) support points – data that act as parameters for the approximation – and the predictive density of a finite multivariate Polya tree. In an initial “warming-up” phase, the support points are iteratively relocated to regions of higher support under the target distribution to minimize the distance between the target distribution and the Polya tree predictive distribution. In the “sampling” phase, samples from the final approximating mixture of finite Polya trees are used as candidates which are accepted with a standard Metropolis-Hastings acceptance probability. Several illustrations are presented, including comparisons of the proposed approach to Metropolis-within-Gibbs and delayed rejection adaptive Metropolis algorithm.

Keywords

Adaptive Metropolis algorithm; Metropolis-Hastings algorithm; Density approximation; Polya trees

1 Introduction

Markov chain Monte Carlo (MCMC) has revolutionized Bayesian statistics. Gibbs sampling (Gelfand & Smith, 1990) and Metropolis-Hastings (Tierney, 1994) algorithms, coupled with a steady increase in cheaply available computational power, have allowed routine fitting of models that a few decades ago would have been practically impossible. However, the fitting

Description of files, DRAM details, additional results and DPpackage function: All supplementary materials associated to this article are described in detail in a single file, which includes the following sections: Details of DRAM implementation, autocorrelation plots to accompany Sections 4.1, 4.2, 4.3, and 4.4, description of the DPpackage R function implementing the Polya tree sampler, R codes illustrating the usage of the PTSampler function, description of additional supplementary files (movies and R panel code; see below), and results of the simulation study comparing PTS, DRAM, and Metropolis within-Gibbs (*PTsampler_supplementary.pdf*).

Movies: Seven movies illustrating the evolution of the location of the support points in the “warming-up” phase of the PTS are included in a compressed file. We refer to the reader to Appendix E included in the file *PTsampler_supplementary.pdf* for a description of each movie (*PTsampler_movies.zip*).

R panel code: An R function (rp.partition) that dynamically displays the effect of changing **m**, **S**, and **O** on partition sets associated to the PTS is also given in the supplementary material. We refer to the reader to Appendix E included in the file *PTsampler_supplementary.pdf* for details regarding the usage of the function (*PTsampler_rpanel.R*).

of many models requires a reasonable amount of manual “tuning” to achieve good mixing, ensuring a representative sample from the posterior has been harvested. Further, although broadly applicable, it is well known that Gibbs sampler strategies, implemented e.g. in BUGS (Gilks et al., 1992; Spiegelhalter et al., 2003), are inefficient at sampling highly correlated posteriors. For mixed models, this has led to various fixes advocating reparameterizations (see, e.g., Gelfand et al., 1995, 1996), and block updates (see, e.g., Sargent et al., 2000), typically implemented on a case-by-case basis.

For general models, the Metropolis-Hastings algorithm can address the problem of highly correlated posteriors through correlated proposals $\mathbf{x}^* \sim q(\cdot|\mathbf{x})$, constructed from an initial run of the MCMC algorithm using a crude, uncorrelated proposal, termed a pilot adaption scheme (PAS) by Gilks et al. (1998). More recently, adaptive algorithms have been proposed that update the proposal distribution “on the fly” as the algorithm proceeds, using all information gathered during the run of the chain from the first iterate on. Such algorithms obviate the need for initial pilot or tuning runs. Haario et al. (2006) combine aspects of two adaptive approaches termed *delayed rejection* (Tierney & Mira, 1999) and *adaptive Metropolis* (Haario et al., 2001; Atchadé & Rosenthal, 2005) into *delayed rejection adaptive Metropolis* (DRAM). Haario et al. (2006) also provide an overview of adaptive methods.

It is well-known that an independence Metropolis-Hastings proposal density that is close to the target distribution yields better performance. Whereas Haario et al. (2001) consider adaptation of multivariate Gaussian random walk Metropolis proposals parameterized by the covariance matrix, recent works have proposed adaptive independence proposals based on finite mixtures. Warnes (2001), Cappé et al. (2008), Keith et al. (2008), Holden et al. (2009), Ji & Schmidler (2009) and Giordani & Kohn (2010) consider adaptively updating a Metropolis-Hastings proposal comprised of a finite mixture of densities. In essence $f(\mathbf{x})$ is approximated by a finite mixture and the approximation improves as the algorithm proceeds. Cai et al. (2008) propose adaptive proposals based on simple piecewise approximating functions in the spirit of Metropolised adaptive rejection sampling (Gilks et al., 1995; Meyer et al., 2008). They extend their approach to multivariate settings by considering component at a time samplers in a Gibbs sampling scheme.

In this paper we propose an alternative algorithm, called the Polya tree (PT) sampler, to explore a density $f(\mathbf{x})$, $\mathbf{x} \in \mathbb{R}^d$. The algorithm is based on Bayesian nonparametric methods for density estimation (see, e.g., Müller & Quintana, 2004, for an overview). Specifically, we elaborate on recent advances on mixtures of finite multivariate PT priors to propose a remarkably simple, yet effective algorithm, complementing the vastly growing literature on adaptive Metropolis schemes. The algorithm is based on the automatic construction of a “working guess” $g(\cdot)$ in an initial “warming-up” phase, from which the candidates will be sampled from. The approximation $g(\cdot)$ is the marginal posterior density arising from a finite multivariate PT based on “data” $\mathbf{x}_1, \dots, \mathbf{x}_W$, termed *support points* for reasons explained below. A PT density $g(\cdot)$ can be viewed as an augmentation of an existing family of probability densities, here the d -variate normal family, $N_d(\mathbf{m}, \mathbf{S})$, with mean vector $\mathbf{m}$ and covariance matrix $\mathbf{S}$ obtained as sample moments from the support points $\mathbf{x}_1, \dots, \mathbf{x}_W$. The PT adds random conditional probabilities to the original density allowing for additional shape, akin to adding detail to an initially washed canvas. The result is essentially a random histogram with bins defined by the Gaussian family $N_d(\mathbf{m}, \mathbf{S})$ through partition sets and a particular correlation structure among bin heights. We refer to Hanson (2007) and Christensen et al. (2008) for introductions to PTs and Lavine (1992; 1994) for a formal treatment.

The approximation $g(\cdot)$, which is built using an algorithm with both deterministic and stochastic elements, has several attractive features, including speed and ease of

implementation. The algorithm compares the current approximation $g(\cdot)$ to the unknown density $f(\cdot)$ over the support points $\mathbf{x}_1, \dots, \mathbf{x}_W$. A support point in an over-represented area is simply moved to an under-represented area and the centering moments $(\mathbf{m}, \mathbf{S})$ are accordingly updated. The algorithm continues until the distance between the approximating density and the unknown density is minimized over all support points. In the “sampling” phase, the resulting approximation $g(\cdot)$ is used as a candidate generating distribution and the candidates are accepted with a standard Metropolis-Hastings acceptance probability. We have implemented the Polya tree sampler for 20-dimensional $\mathbf{x}$ with excellent results, although much higher dimensions may be feasible with modifications to the basic algorithm, as we discuss below.

The rest of this article is organized as follows. Section 2 briefly reviews aspects of multivariate Polya trees necessary for Section 3, which develops the Polya tree sampler. Section 4 presents test cases comparing our approach to DRAM and Gibbs sampling algorithms. In Section 5 we conclude the paper and suggest several avenues of possible future research.

2 Finite multivariate Polya trees

Background on finite multivariate PTs and the associated computational issues are presented here. Additional details can be found in Hanson (2006), Hanson et al. (2008), and Jara et al. (2009).

2.1 Definition

A finite d -variate PT prior with J levels, for a probability measure G supported on $\mathbb{R}^d$, is characterized by a collection of increasingly refined partitions of $\mathbb{R}^d$, and associated conditional probabilities that link a parent set in a given level to its 2^d offspring sets in the subsequent level. The sequence of nested partitions is denoted by $\Pi_1, \dots, \Pi_J$ where, for $1 \leq j < J$, Π_{j+1} is a refinement of the partition Π_j in that each set in Π_j is the union of 2^d sets in Π_{j+1} .

A simple, natural class of partitions starts with base sets that are Cartesian products of intervals obtained as quantiles from the standard normal distribution. For $j = 1, \dots, J$, the base sets that comprise level j are given by

$$B_0(j; \mathbf{k}) = (\Phi^{-1}(k_1/2^j), \Phi^{-1}((k_1+1)/2^j)) \times \dots \times (\Phi^{-1}(k_d/2^j), \Phi^{-1}((k_d+1)/2^j)),$$

for vectors $\mathbf{k} = (k_1, \dots, k_d) \in \kappa_j = \{0, \dots, 2^{j-1}\}^d$. For each $j = 1, \dots, J$, the class $\{B_0(j; \mathbf{k}) : \mathbf{k} \in \kappa_j\}$ partitions $\mathbb{R}^d$ except on a set of Lebesgue measure zero. The case of $\mathbb{R}^2$ is easily visualized; Figure 1(a) shows partition sets $B_0(1; 0, 0)$ (lower left), $B_0(1; 1, 0)$ (lower right), $B_0(1; 1, 1)$ (upper right), and $B_0(1; 0, 1) = B_0(2; 0, 2) \cup B_0(2; 1, 2) \cup B_0(2; 0, 3) \cup B_0(2; 1, 3)$ (upper left).

Given a vector mean $\mathbf{m}$ and a square root, $\mathbf{U}$, of the covariance matrix $\mathbf{S} = \mathbf{U}\mathbf{U}'$, a location-scale transformation is applied to each base set yielding

$$B(j; \mathbf{k}) = \{\mathbf{m} + \mathbf{U}\mathbf{z} : \mathbf{z} \in B_0(j; \mathbf{k})\},$$

with $\Pi_j = \{B(j; \mathbf{k}) : \mathbf{k} \in \kappa_j\}$. Figure 1(b) shows the base sets in 1(a) transformed by an arbitrary $\mathbf{m}$ and $\mathbf{U} = \mathbf{S}^{1/2}$ where $\mathbf{S}^{1/2}$ is the unique symmetric square root of

$$\mathbf{S} = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}.$$

The partitions $\Pi_1, \dots, \Pi_J$ are augmented with random conditional probabilities. Let $\mathbf{x} = (x_1, \dots, x_d)'$ be an observation from G . Given that $\mathbf{x}$ is in set $B(j; \mathbf{k})$ at level j , then $\mathbf{x}$ belongs to one of the 2^d off-spring sets in Π_{j+1} that partition $B(j; \mathbf{k})$. Denote these offspring sets by $\{B(j+1; 2\mathbf{k} + \mathbf{s}): \mathbf{s} \in \kappa_1\}$, and let the corresponding conditional probabilities of being in these sets be grouped into the vector $\mathbf{Y}(j; \mathbf{k}) = \{Y(j+1; 2\mathbf{k} + \mathbf{s}): \mathbf{s} \in \kappa_1\}$, so that $P(\mathbf{x} \in B(j+1; 2\mathbf{k} + \mathbf{s}) | \mathbf{x} \in B(j; \mathbf{k})) = Y(j+1; 2\mathbf{k} + \mathbf{s})$. For all $\mathbf{k} \in \kappa_j$, the $\mathbf{Y}(j; \mathbf{k})$ are independent Dirichlet random vectors,

$$\mathbf{Y}(j; \mathbf{k}) \sim \text{Dirichlet}(\mathbf{1}_{2^d} c \rho(j)),$$

where $\mathbf{1}_{2^d}$ denotes a vector of length 2^d containing all ones and $\rho(j) = j^2$. Motivated by continuity arguments, the function $\rho(j) = j^2$ is typically used (e.g., Hanson, 2006; Jara et al., 2009), but not always (e.g., Berger & Guglielmi, 2001; Hanson et al., 2008). The parameter $c > 0$ controls how much weight is placed on the centering family, playing a similar role to the precision parameter of the widely known Dirichlet process prior. Finally, a finite d -variate PT assumes that G follows the $N_d(\mathbf{m}, \mathbf{S})$ distribution on sets at level J . The construction previously described centers the PT prior for G at the $N_d(\mathbf{m}, \mathbf{S})$ distribution.

2.2 Useful properties of finite multivariate Polya trees

Finite d -variate PT priors enjoy an important conjugacy property. Given data generated from G ,

$$\mathbf{x}_1, \dots, \mathbf{x}_W | G \stackrel{iid}{\sim} G,$$

the posterior distribution for G is a PT prior, and the updating is performed in the Dirichlet hyper-parameters such that

$$\mathbf{Y}(j; \mathbf{k}) | \mathbf{x}_1, \dots, \mathbf{x}_W \sim \text{Dirichlet}(\mathbf{1}_{2^d} c \rho(j) + \mathbf{n}(j, \mathbf{k}; \mathbf{x}_{1:W})), \quad (1)$$

where the elements of $\mathbf{n}(j, \mathbf{k}; \mathbf{x}_{1:W})$, with $\mathbf{x}_{1:W} = [\mathbf{x}_1 \mathbf{x}_2 \dots \mathbf{x}_W]$ being the data matrix, count how many data vectors $\mathbf{x}_1, \dots, \mathbf{x}_W$ fall into each of the offspring sets $\{B(j+1; 2\mathbf{k} + \mathbf{s}): \mathbf{s} \in \kappa_1\}$. The counting needed to update PT prior can be performed in an efficient manner as described next.

Let $\mathbf{k}(\mathbf{x}, j, \mathbf{m}, \mathbf{U})$ be the set in Π_j that contains $\mathbf{x}$, i.e., $\mathbf{x} \in B(j; \mathbf{k}(\mathbf{x}, j, \mathbf{m}, \mathbf{U}))$. This set is given by

$$\mathbf{k}(\mathbf{x}, j, \mathbf{m}, \mathbf{U}) = (\lfloor 2^j \Phi(z_1) \rfloor, \dots, \lfloor 2^j \Phi(z_d) \rfloor)',$$

where $\lfloor x \rfloor$ is the floor function giving the integer part of x , and $\mathbf{z} = (z_1, \dots, z_d)' = \mathbf{U}^{-1}(\mathbf{x} - \mathbf{m})$. Let $n(j, \mathbf{k}; \mathbf{x}_{1:W})$ count the number of vectors $\mathbf{x}_1, \dots, \mathbf{x}_W$ that are in set $B(j, \mathbf{k})$, given by

$$n(j, \mathbf{k}; \mathbf{x}_{1:W}) = \sum_{i=1}^W I\{\mathbf{k}(\mathbf{x}_i, j, \mathbf{m}, \mathbf{U}) = \mathbf{k}\}. \quad (2)$$

Notice that the numbers falling into the offspring sets of $B(j, \mathbf{k})$ depend on $\mathbf{m}$ and $\mathbf{S}$, the latter through its decomposition $\mathbf{U}$, and are given by $\mathbf{n}(j, \mathbf{k}; \mathbf{x}_{1:W}) = \{n(j, 2\mathbf{k} + \mathbf{s}; \mathbf{x}_{1:W}) : \mathbf{s} \in \mathcal{K}_1\}$.

The PT predictive distribution is a key component of our algorithm. The predictive density of a new observation $\mathbf{x}$ given data $\mathbf{x}_{1:i}$ and fixed parameters $(c, \mathbf{m}, \mathbf{U})$, obtained by integrating over all $\mathbf{Y}(j, \mathbf{k})$ vectors in (1), is given by

$$p(\mathbf{x}|\mathbf{x}_{1:i}, c, \mathbf{m}, \mathbf{U}) = \varphi_d(\mathbf{x}|\mathbf{m}, \mathbf{S}) \prod_{j=1}^J \frac{2^d c \rho(j) + 2^d n(j, \mathbf{k}(\mathbf{x}, j, \mathbf{m}, \mathbf{U}); \mathbf{x}_{1:i})}{2^d c \rho(j) + n(j-1, \mathbf{k}(\mathbf{x}, j-1, \mathbf{m}, \mathbf{U}), \mathbf{x}_{1:i})}, \quad (3)$$

where by definition $n(0, 0, \mathbf{x}_{1:i}) = i$. Through expression (3), the induced density for the observed data is then given by

$$\begin{aligned} p(\mathbf{x}_{1:W}|c, \mathbf{m}, \mathbf{U}) &= \prod_{i=1}^W p(\mathbf{x}_i|\mathbf{x}_{1:i-1}, c, \mathbf{m}, \mathbf{U}), \\ &= \left[\prod_{i=1}^W \varphi_d(\mathbf{x}_i|\mathbf{m}, \mathbf{S}) \right] \left[\prod_{i=2}^W \prod_{j=1}^J \frac{2^d c \rho(j) + 2^d n(j, \mathbf{k}(\mathbf{x}_i, j, \mathbf{m}, \mathbf{U}); \mathbf{x}_{1:i-1})}{2^d c \rho(j) + n(j-1, \mathbf{k}(\mathbf{x}_i, j-1, \mathbf{m}, \mathbf{U}), \mathbf{x}_{1:i-1})} \right], \\ &\stackrel{\text{def}}{=} \left[\prod_{i=1}^W \varphi_d(\mathbf{x}_i|\mathbf{m}, \mathbf{S}) \right] q(\mathbf{x}_{1:W}|c, \mathbf{m}, \mathbf{U}). \end{aligned} \quad (4)$$

Note that recursively, $\lfloor 2^{j-1} \Phi(\mathbf{z}) \rfloor = \lfloor 0.5 \lfloor 2^j \Phi(\mathbf{z}) \rfloor \rfloor$, implying $\lfloor 2^j \Phi(\mathbf{z}) \rfloor = \lfloor \lfloor 2^J \Phi(\mathbf{z}) \rfloor / 2^{J-j} \rfloor$, where $\lfloor \cdot \rfloor$ operates elementwise. Therefore, expression (2) implies that $q(\mathbf{x}_{1:W}|c, \mathbf{m}, \mathbf{U})$ in (4) can be computed as

$$q(\mathbf{x}_{1:W}|c, \mathbf{m}, \mathbf{U}) = \prod_{i=2}^W \prod_{j=1}^J \frac{2^d c \rho(j) + 2^d \sum_{t=1}^{i-1} I\{\lfloor 2^{j-J} \mathbf{k}_i \rfloor = \lfloor 2^{j-J} \mathbf{k}_t \rfloor\}}{2^d c \rho(j) + \sum_{t=1}^{i-1} I\{\lfloor 2^{j-1-J} \mathbf{k}_i \rfloor = \lfloor 2^{j-1-J} \mathbf{k}_t \rfloor\}}, \quad (5)$$

where $\mathbf{k}_i = \mathbf{k}(\mathbf{x}_i, J, \mathbf{m}, \mathbf{U})$. The evaluation of $q(\mathbf{x}_{1:W}|\mathbf{m}, \mathbf{U})$ requires

$$\sum_{i=1}^W (i-1) = 0.5W(W-1) \text{ comparisons of vectors of size } d.$$

Finally, use of the approximation $g(\mathbf{x}) = p(\mathbf{x}|\mathbf{x}_{1:W}, c, \mathbf{m}, \mathbf{U})$, for a fixed $(c, \mathbf{m}, \mathbf{U})$ as an independence proposal requires sampling $\mathbf{x}^* \sim g$ and evaluating $g(\mathbf{x}^*)$ and $g(\mathbf{x})$. The latter is accomplished through expression (3). Sampling from the predictive density $\mathbf{x}_{W+1} \sim p(\cdot|\mathbf{x}_{1:W}, c, \mathbf{m}, \mathbf{U})$ proceeds by successively sampling sets $\mathbb{R}^d \supset B(1, \mathbf{k}_1) \supset B(2, \mathbf{k}_2) \supset \dots \supset B(J, \mathbf{k}_J)$ through sampling $\mathbf{s}_1, \mathbf{s}_2, \dots, \mathbf{s}_J$ and setting $\mathbf{k}_j = 2\mathbf{k}_{j-1} + \mathbf{s}_j$, obtained through the expectation of the Dirichlet distribution given by expression (1). Let $\mathbf{k}_0 = \mathbf{0}$. Then

$$P(\mathbf{s}_{j+1}=\mathbf{s})=\frac{c\rho(j)+n(j+1, 2\mathbf{k}_j+\mathbf{s};\mathbf{x}_{1:W})}{2^d c\rho(j)+n(j, \mathbf{k}_j;\mathbf{x}_{1:W})}, \quad \mathbf{s} \in \mathcal{K}_1.$$

Conditionally on $\mathbf{k}_j$, $\mathbf{x}_{W+1}$ is simulated from a $N_d(\mathbf{m}, \mathbf{S})$ distribution restricted to the set $B(J; \mathbf{k}_j)$.

The computations just described effectively use one $d \times W$ data matrix. For very high-dimensional problems, Hanson et al. (2008) suggest a different matrix of dimension $J \times W$; the algorithms can be modified to use this matrix instead, but at slightly increased computational cost.

2.3 Centering family and smoothing via orthogonal matrices

The prior mean of the process is the centering $N_d(\mathbf{m}, \mathbf{S})$ distribution. The Gaussian family guides the approximation in terms of overall shape and where to look for more probability mass. A Gaussian centering family is motivated and justified by the Bayesian central limit theorem, yielding approximately Gaussian posteriors in a variety of models (see Section 5.3.2 in Bernardo & Smith, 2000). Other centering families could be considered, e.g. a multivariate t gives heavier tails and might be more appropriate for some posteriors.

Individual parameters, i.e. elements of $\mathbf{x}$, with support other than $\mathbb{R}$ can be transformed, but this is not strictly necessary for highly focused posteriors $f(\cdot)$ as long as the initial points $\mathbf{x}_1$,

..., $\mathbf{x}_W$ obey support constraints. Otherwise, the Fisher transformation $x=\log\left\{\frac{1-\rho}{1+\rho}\right\}$ can be used for a correlation ρ , precisions τ and other strictly positive parameters can be log-transformed using $x=\log(\tau)$, and for proportions the logistic or probit transformations $x=\log\left\{\frac{\pi}{1-\pi}\right\}$ and $x=\Phi^{-1}(\pi)$ can be used.

Even though the prior mean of the multivariate PT may have a smooth Lebesgue density, the posterior predictive density for a future observation have noticeable jumps at the boundaries of partitioned sets. In the multivariate PT defined in the previous section, the partition sets are functions of the centering mean and the covariance matrix. In particular, the direction of the sets are defined through the decomposition of the covariance matrix. Hanson (2006) and Hanson et al. (2008) chose $\mathbf{U} = \mathbf{U}' = \mathbf{S}^{1/2}$, where $\mathbf{S}^{1/2} = \mathbf{M}\mathbf{\Lambda}^{1/2}\mathbf{M}'$ is the unique symmetric square root of $\mathbf{S}$. Here, the columns of $\mathbf{M}$ are the orthonormal eigenvectors of $\mathbf{S}$ and the diagonal elements of $\mathbf{\Lambda}$ house the corresponding eigenvalues. We consider a multivariate PT that separates the direction of the partition sets $\mathbf{U}$ and the centering covariance matrix $\mathbf{S}$ by a further decomposition in the form of an orthogonal matrix $\mathbf{O}$. The random $\mathbf{O}$ smooths over partitioning effects yielding a smooth density. Specifically, we consider $\mathbf{U} = [\mathbf{M}\mathbf{\Lambda}^{1/2}]\mathbf{O}$, where $[\mathbf{M}\mathbf{\Lambda}^{1/2}]$ is a particular square root of $\mathbf{S}$, and place a Haar prior on $\mathbf{O}$; this is described in greater detail in Section 3.1.1.

Figure 1 panels (b), (c), and (d) continue the $\mathbb{R}^2$ example from Section 2.1 considering

$$\mathbf{S} = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}, \text{ and (b) } \mathbf{O} = \mathbf{M}', \text{ (c) } \mathbf{O} = \mathbf{I}_2, \text{ and (d) } \mathbf{O} = \begin{bmatrix} \cos(\pi/4) & -\sin(\pi/4) \\ \sin(\pi/4) & \cos(\pi/4) \end{bmatrix}.$$

Note that different $\mathbf{O}$ essentially rotate the partition sets, but the prior probability of each set $B(j; \mathbf{k})$ remains 2^{-jd} regardless of $\mathbf{O}$.

3 The Polya tree sampler

The Polya tree sampler (PTS) is described in this section. The basic algorithm has two separate iterative sub-algorithms, called the “warming-up” and “sampling” phases. To fix notation, let $f(\cdot)$ denote the unnormalized density to be explored by sampling, and $g(\cdot)$ be the approximation given by the predictive density of a finite PT with parameters $(c, \mathbf{m}, \mathbf{U})$. The algorithm is initialized by sampling $\mathbf{x}_1, \dots, \mathbf{x}_W$ from a best guess at $f(\cdot)$. A Gibbs sampling run or large sample considerations work well here. However, as we will show later, these points do not need to be a representative sample from $f(\cdot)$.

3.1 The warming-up phase

In the adaptive phase of the algorithm, the approximation $g(\cdot)$ is iteratively generated. The algorithm operates like an archaeological dig. A promising area $(\mathbf{m}, \mathbf{S})$ is staked out, gridded in certain directions $(\mathbf{U})$, yielding sections $\{B(j, \mathbf{k})\}$ delineated by ropes, and initially dug at locations $\mathbf{x}_1, \dots, \mathbf{x}_W$, and the (relative) density of archaeological treasures $f(\mathbf{x}_i)$ recorded. More promising dig sites $\mathbf{x}_u$ are allocated resources that are currently at more exhausted sites $\mathbf{x}_o$, until the more promising sites are excavated; the grids $\mathbf{U}$ are continually being moved as excavation proceeds. Note that if the entire initial dig site is centered on an area of low density, then digging focuses on the outer edges and the whole dig site moves in the direction of higher density. The process continues until all artifacts are unearthed according to some criterion.

Define discrete approximations to $f(\cdot)$ and $g(\cdot)$, $F_a(\cdot) = \sum_{i=1}^W w_f(i) \delta_{\mathbf{x}_i}(\cdot)$ and $G_a(\cdot) = \sum_{i=1}^W w_g(i) \delta_{\mathbf{x}_i}(\cdot)$, respectively, where $\delta_{\mathbf{x}}$ is the Dirac measure at $\mathbf{x}$ and the weights are

$$w_f(i) = \frac{f(\mathbf{x}_i)}{\sum_{j=1}^W f(\mathbf{x}_j)}, \quad w_g(i) = \frac{g(\mathbf{x}_i)}{\sum_{j=1}^W g(\mathbf{x}_j)}.$$

At each iteration of the algorithm a support point $\mathbf{x}_i$ in an over-represented region is moved to an under-represented region, according to the discrete approximations, and the centering parameters $(\mathbf{m}, \mathbf{S})$ for $g(\cdot)$ are updated. This very simple algorithm does a surprisingly good job at seeking out regions of high probability and approximating $f(\cdot)$. In the basic algorithm the number of support points W is fixed; areas with too many support points relative to how much mass is actually present are thus thinned out. There are several aspects to the algorithm which are discussed in turn after presenting the basic steps.

Let $\mathbf{s}_1 = \sum_{i=1}^W \mathbf{x}_i$ and $\mathbf{S}_2 = \sum_{i=1}^W \mathbf{x}_i \mathbf{x}_i'$ and compute the density values $f(\mathbf{x}_i)$ for $i = 1, \dots, W$. Notice that the normalizing constant need not be included. Set $L_1 = 1$ and initialize c to some large value, e.g. $c = 1000$. The warming-up phase consists of iterating through the following eight steps until $L_1 < \delta$ plus an additional fixed number N of iterations,

1. Compute $\mathbf{m} = \mathbf{s}_1/W$ and $\mathbf{S} = \mathbf{S}_2/W - \mathbf{m}\mathbf{m}'$. Compute the spectral decomposition of $\mathbf{S}$, $\mathbf{S} = \mathbf{M}\mathbf{\Lambda}\mathbf{M}'$.
2. Generate $\mathbf{O} \sim \text{Haar}(d)$. Set $\mathbf{U} = \mathbf{M}\mathbf{\Lambda}^{1/2}\mathbf{O}$ and $\mathbf{U}^{-1} = [\mathbf{M}\mathbf{\Lambda}^{-1/2}\mathbf{O}]'$.
3. Compute $w_g(i) = p(\mathbf{x}_i | \mathbf{x}_{1:W}, c, \mathbf{m}, \mathbf{U}) / \sum_{j=1}^W p(\mathbf{x}_j | \mathbf{x}_{1:W}, c, \mathbf{m}, \mathbf{U})$, $w_f(i) = f(\mathbf{x}_i) / \sum_{j=1}^W f(\mathbf{x}_j)$, and $r(i) = w_f(i)/w_g(i)$ for $i = 1, \dots, W$.

4. Compute $L_1 = 0.5 \sum_{i=1}^W |w_f(i) - w_g(i)|$.
5. $o = \operatorname{argmin}\{r(i)\}$ and $u = \operatorname{argmax}\{r(i)\}$; $\mathbf{s}_1 = \mathbf{s}_1 - \mathbf{x}_o$; $\mathbf{S}_2 = \mathbf{S}_2 - \mathbf{x}_o \mathbf{x}_o'$.
6. Generate $\mathbf{x}_o \sim N_d(\mathbf{m}, \mathbf{S})$ restricted to set $B(J, \mathbf{k}(\mathbf{x}_u, J, \mathbf{m}, \mathbf{U}))$.
7. $\mathbf{s}_1 = \mathbf{s}_1 + \mathbf{x}_o$; $\mathbf{S}_2 = \mathbf{S}_2 + \mathbf{x}_o \mathbf{x}_o'$.
8. Generate $\log c^* \sim N(\log c, s_c^2)$. If $q(\mathbf{x}_{1:W}|c^*, \mathbf{m}, \mathbf{U}) > q(\mathbf{x}_{1:W}|c, \mathbf{m}, \mathbf{U})$ then $c = \max\{c^*, 1\}$.

Step 3 requires the evaluation of expression (3) while step 8 requires the evaluation of expression (5). For computationally expensive evaluations of $f(\cdot)$, one can keep a vector of evaluations $\mathbf{f} = (f(\mathbf{x}_1), \dots, f(\mathbf{x}_W))'$. Initially, $s_f = \mathbf{1}_W' \mathbf{f}$. When $\mathbf{x}_o$ replaces $\mathbf{x}_u$ in step 6, simply replace element u in $\mathbf{f}$ by $f(\mathbf{x}_o)$ and take $s_f = s_f - f(\mathbf{x}_u) + f(\mathbf{x}_o)$; then $\mathbf{w}_f = \mathbf{f}/s_f$. Also note that the normalizing constants in the denominators of $w_g(i)$ and $w_f(i)$ when computing $r(i)$ in step 3 are not needed when taking the maximum and minimum in step 5. We have implemented the algorithm using log-densities instead of densities for greater numerical stability.

3.1.1 The partition—As discussed earlier, the asymmetric square root $\mathbf{U}$ used to decompose the centering covariance $\mathbf{S} = \mathbf{U}\mathbf{U}'$ in step 2 determines the overall shape and direction of partition set borders. The basic partition sets $B_0(j, \mathbf{k})$ are bordered by hyperplanes perpendicular to the coordinate axes, in two dimensions forming a sort of checkerboard. These sets are mapped to partition sets by the affine transformation $B(j, \mathbf{k}) = \{\mathbf{m} + \mathbf{U}\mathbf{z} : \mathbf{z} \in B_0(j, \mathbf{k})\}$. The decomposition $\mathbf{U} = \mathbf{S}^{1/2} = \mathbf{M}\mathbf{\Lambda}^{1/2}\mathbf{M}'$, however, limits the shape of the partition sets (see, Jara et al., 2009). Roughly, this hampers the algorithm's ability to search around the under represented support points by focusing only in certain directions determined by $\mathbf{S}^{1/2}$. The class of decompositions is broadened by considering all asymmetric square roots, given by $\mathbf{U} = \mathbf{M}\mathbf{\Lambda}^{1/2}\mathbf{O}$ where $\mathbf{O}$ is any orthogonal matrix, i.e. $\mathbf{O}'\mathbf{O} = \mathbf{O}\mathbf{O}' = \mathbf{I}$. By drawing $\mathbf{O}$ randomly from some distribution, the algorithm looks in different, random directions, and the predictive density is smoothed a great deal. Note that the generated $\mathbf{O}$ could be rejected according to a Metropolis-Hastings-type criterion; however, we do not pursue this here.

There are several ways of sampling from the Haar distribution, essentially a uniform distribution on orthogonal matrices, including methods based on Givens rotations (Anderson et al., 1987; Lundberg & Svensson, 2004). We consider an approach based on a $\mathbf{QR}$ decomposition on a $d \times d$ matrix of *iid* $N(0, 1)$ entries, normalized such that the diagonal elements of $\mathbf{R}$ are positive (see, e.g., Stewart, 1980; Mezzadri, 2007). The resulting $\mathbf{Q}$ matrix is a draw from the Haar(d) measure. This approach is fairly fast for lower dimensions, but much higher dimensions would require a different method for generating from the Haar measure, or a simpler distribution on $\mathbf{O}$, e.g. perhaps a subset of Givens rotations, which are fast to compute and have shortcuts for computing products.

In the online supplementary material we include an R function (please see the file PTS_rpanel.R) that dynamically illustrates the effect of changing $\mathbf{m}$, $\mathbf{S}$, and

$$\mathbf{O} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix},$$

across different values of θ on partition sets in Π_1 and Π_2 in $\mathbb{R}^2$. Note that $\theta = 0$ gives $\mathbf{O} = \mathbf{I}_2$.

3.1.2 Sampling under-represented areas—The simplest version of the algorithm randomly draws from the approximating density truncated to the set that contains $\mathbf{x}_u$ in the finest partition, $B(J, \mathbf{k}(\mathbf{x}_u, J, \mathbf{m}, \mathbf{U}))$. On this set, the approximation is the $N_d(\mathbf{m}, \mathbf{S})$ distribution. There are many ways to refine the algorithm to sample larger sets or in other areas of $\mathbb{R}^d$.

Sampling the partition set around $\mathbf{x}_u$ at level J tightly focuses the search. An approach that instead samples the partition set j that includes $\mathbf{x}_u$ adaptively according to a denseness criterion, e.g. how many points are in the same set at level j , allows more ground to be covered. If there are few points in the coarsest set $B(1, \mathbf{k}(\mathbf{x}_u, 1, \mathbf{m}, \mathbf{U}))$, it makes sense to randomly draw from this larger set rather than the smallest $B(J, \mathbf{k}(\mathbf{x}_u, J, \mathbf{m}, \mathbf{U}))$. A possible criterion would be to start $j = 1$, and increase $j \leftarrow j + 1$ until $n(j, \mathbf{k}(\mathbf{x}_u, j, \mathbf{m}, \mathbf{U}), \mathbf{x}_{1:W}) < K$, say $K = 5$ or 10 . At this level j , $\mathbf{x}_o$ is replaced by sampling from $N_d(\mathbf{m}, \mathbf{S})$ – essentially $g(\cdot)$ since there are few support points in the vicinity – restricted to $B(j, \mathbf{k}(\mathbf{x}_u, j, \mathbf{m}, \mathbf{U}))$. The latter can more easily sample areas of low probability mass under $g(\cdot)$.

Possibly, a check could be made, and the new $\mathbf{x}_o^*$ accepted only if the new $\mathbf{x}_o^*$ is truly in a more under represented area, i.e. if $f(\mathbf{x}_o^*)/g(\mathbf{x}_o^*) > f(\mathbf{x}_o)/g(\mathbf{x}_o)$ or accepted with Metropolis-Hastings probability

$$1 \wedge \frac{f(\mathbf{x}_o^*)g(\mathbf{x}_o)}{g(\mathbf{x}_o)g(\mathbf{x}_o^*)},$$

in which $\mathbf{x}_o^*$ is always accepted when $f(\mathbf{x}_o^*)/g(\mathbf{x}_o^*) > f(\mathbf{x}_o)/g(\mathbf{x}_o)$, but randomly accepted otherwise, allowing more global exploration at the cost of exploring the most immediately promising area. Taking this idea one step further, if instead of restricting the sampling of $\mathbf{x}_o^*$ to a partition set, it is drawn from unrestricted $g(\cdot)$ and accepted with probability ρ the result is a “moving window” adaptive independence sampler, where earlier drawn, perhaps “unrepresentative” values are traded in for values more reflective of $f(\cdot)$ as the algorithm continues. If earlier values are kept, a straightforward adaptive independence proposal is enacted. This idea is further discussed in Section 5.

A related way to pick off clumps of outer lying mass removed from the centering distribution is to complete Step 6 with some probability τ that might be a function of stopping criteria and/or the number of adaptive iterates completed. With probability $1 - \tau$ the current over-represented support point $\mathbf{x}_o$ is instead placed at a random draw from a set with much larger support, as a “Hail Mary” to try and pick off outlying clumps of probability, e.g. either sampling from $g(\cdot)$ or $N_d(\mathbf{m}, \mathbf{S})$. This has the effect of broadly placing support points, including places where current support points are scarce, but where the parametric centering family suggests there is mass. A Metropolis-Hastings type acceptance probability could instead be used here as described above. We have tried $\tau = 0.95$ and 0.9 , with and without τ increasing to unity as the number of adaptive iterations increases. Best results are obtained when τ starts out relatively small, and increases to unity as the approximation improves (e.g. L_1 gets smaller, described below). Tempering (Geyer & Thompson, 1995) provides another possible avenue toward exploring $f(\cdot)$ with well separated modes.

3.1.3 Stopping rules—At some point the warming-up phase of the algorithm needs to be terminated and the approximation used for inference. This proved to be the most difficult aspect of the algorithm to fine tune across different test cases. Many stopping rules were considered, and many others could be conjectured; the most promising of the ones we considered are described below. These stopping rules produce a fairly crude approximation, but one that often reproduces major features, including skew and multiple modes, quite well (see, for example, Figures 2 and 3). Notice that features of the parametric family drive the approximation in areas of little probability mass.

The warming-up phase should stop when all of the “probability mass shifting” has come to a standstill and the approximation has reached a steady state. Such a steady state would occur when the current approximating density is “representative” of $f(\mathbf{x})$ on the support points, i.e. when the ratios $f(\mathbf{x}_i)/g(\mathbf{x}_i)$ are close to one. For example, if the densities differ by less than a factor of 5% across all $\mathbf{x}_1, \dots, \mathbf{x}_W$, we require

$$D_1 = \max_{i=1, \dots, W} |\log\{f(\mathbf{x}_i)\} - \log\{g(\mathbf{x}_i)\}| < \log(1.05) \approx 0.05 = \delta. \quad (6)$$

Unfortunately, this requires that $\int_{\mathbb{R}^d} f(\mathbf{x}) d\mathbf{x} = 1$. Since expression (6) cannot be checked, $f(\mathbf{x}_i)$ and $g(\mathbf{x}_i)$ could be replaced by $w_f(i)$ and $w_g(i)$ instead. Alternatively, the largest factor could be compared to the smallest through the ratio (the normalizing constant for $f(\cdot)$ cancels):

$$D_2 = \max_{i=1, \dots, W} \left\{ \frac{f(\mathbf{x}_i)}{g(\mathbf{x}_i)} \right\} / \min_{i=1, \dots, W} \left\{ \frac{f(\mathbf{x}_i)}{g(\mathbf{x}_i)} \right\}, \quad (7)$$

and the algorithm stops when this is within some factor, say 2.

An ideal discrepancy measure would be the total variation distance between $f(\cdot)$ and $g(\cdot)$, $0.5 \int_{\mathbb{R}^d} |f(\mathbf{x}) - g(\mathbf{x})| d\mathbf{x}$, although this is problematic to compute for many reasons. Along these lines we instead consider the total variation distance between F_a and G_a , given by

$$L_1 = 0.5 \sum_{i=1}^W |w_f(i) - w_g(i)|.$$

Necessarily, $0 \leq L_1 \leq 1$, and although we denote the measure L_1 , the L_1 -distance is actually twice this norm. All stopping rules fared roughly the same, but L_1 seemed to produce slightly better estimates and, therefore, we use this measure here.

A related stopping rule could consider the total variation distance between successive – or based on successive windows of fixed iteration length – estimators $g(\cdot)$ and stop when this discrepancy is small. For PT centered at the same $N_d(\mathbf{m}, \mathbf{S})$ distribution, this discrepancy simplifies to a function of counts of $\mathbf{x}_1, \dots, \mathbf{x}_W$ falling into partition sets. For two PT with different centering parameters $(\mathbf{m}, \mathbf{S})$, the measure would be impractical to compute.

For $f(\cdot)$ that are highly “irregular”, the approximation $g(\cdot)$ may not be able to capture aspects of $f(\cdot)$ well enough for smaller values, e.g. $L_1 < 0.1$ unless c is allowed to approach zero and/or larger W is considered. This can manifest itself in $\mathbf{x}_u$ and $\mathbf{x}_o$ alternately swapping at each iteration. This “switching” phenomenon can be checked for along with computation of

a distance. We found picking $\delta = 0.3$ to always produce an approximation rather quickly, whereas decreasing to $\delta = 0.2$ or $\delta = 0.1$ improves the approximation but may take an order of magnitude longer to reach, or never be reached, with a fixed W .

3.1.4 Updating c , number of support points W , and follow-up N —Starting c at a larger value forces the initial approximation to be more Gaussian. As the warming-up phase proceeds, the c parameter often tends to a small (e.g. $c < 1$) value for functions $f(\cdot)$ that are very unlike the Gaussian distribution. Bounding c from below, say by unity, helps make the approximation overall a bit smoother and fatten the tails, useful for an independence proposal. The proposal variance s_c^2 affects how rapidly c tends to unity. Setting s_c^2 too small will allow $g(\cdot)$ to be “too flexible” too quickly, i.e. $g(\cdot)$ will quickly adapt to what is essentially the outer reaches of the tail of $f(\cdot)$, preventing rapid movement toward regions of greater probability mass under $f(\cdot)$. We have found $s_c = 0.05$ to work over several test cases. The results of the comparison of $s_c \in \{0.05, 0.1, 1\}$ are given in the Appendix A (online supplementary material).

We have experimented with $W = 200, 300, 500, 1000$. For smaller dimensional $\mathbf{x} \in \mathbb{R}^d$, e.g. $d = 2, 3, 4$, the number does not seem to grossly affect the final approximation $g(\cdot)$ except for very “extreme” looking $f(\cdot)$. However, it would make sense to increase W with dimensionality. Every support point should have an opportunity to be moved at least once, implying that N should be at least as large as $(W - a)$ where a is the number of iterates until $L_1 < \delta$; simply taking $N \geq W$ also works. These “follow-up” iterations allow the algorithm to include some finer detail beyond $L_1 < \delta$. This criterion produces crude (but workable) approximations, but we have found that following up with an additional fixed number of iterates N to greatly improve the approximation $g(\cdot)$, and in some cases is necessary to achieve a good approximation $g(\cdot)$. An alternative approach would be to take δ much smaller, say $\delta = 0.1$, and allow W itself to increase adaptively, as necessary, to accommodate finer detail. In the examples in Sections 4 where $d = 2$ or $d = 4$ we fix $W = 300$ and $N = 500$. This, however, does not work well for the example where $d \geq 5$. In this case $W = 100d$ and $N = 200 + W$ works well.

We stress that the algorithm as described is a basic recipe that has produced very good results across many test cases. However, any recipe can be modified, and simple modifications of the algorithm have produced dramatically improved results in certain situations; we detail one such improvement in Section 4.1. Obtaining an optimal adapting strategy covering all possible posterior densities $f(\cdot)$ is of course impossible, and so we focus on the basic algorithm in what follows.

3.2 The sampling phase

The warm-up phase produces the precision parameter c , the support points $\mathbf{x}_{1:W}$, and through the support points the centering parameters $(\mathbf{m}, \mathbf{S})$. This gives a simple algorithm for generating candidates for $f(\cdot)$ which are accepted using standard Metropolis-Hastings probabilities. The candidate generating kernel is a mixture of finite multivariate PT with predictive density given by $g(\cdot) = p(\cdot | \mathbf{x}_{1:W}, c, \mathbf{m}, \mathbf{U})$ where $\mathbf{U} = \mathbf{M}\mathbf{\Lambda}^{1/2}\mathbf{O}$, and $\mathbf{O} \sim \text{Haar}(d)$. Let $\mathbf{x}^t$ and $\mathbf{O}^t$ the current value of the parameters of interest. The sampling phase is described by the following steps

1. Generate $\mathbf{O}^* \sim \text{Haar}(d)$. Set $\mathbf{U}^* = \mathbf{M}\mathbf{\Lambda}^{1/2}\mathbf{O}^*$ and $[\mathbf{U}^*]^{-1} = [\mathbf{M}\mathbf{\Lambda}^{-1/2}\mathbf{O}^*]^t$.
2. Generate $\mathbf{x}^* \sim p(\cdot | \mathbf{x}_{1:W}, c, \mathbf{m}, \mathbf{U}^*)$.
3. The candidates are accepted, $\mathbf{x}^{t+1} = \mathbf{x}^*$ and $\mathbf{O}^{t+1} = \mathbf{O}^*$, with Metropolis-Hastings probability

$$1 \wedge \frac{f(\mathbf{x}^*)p(\mathbf{x}^t|\mathbf{x}_{1:W}, c, \mathbf{m}, \mathbf{U}^t)}{f(\mathbf{x}^t)p(\mathbf{x}^*|\mathbf{x}_{1:W}, c, \mathbf{m}, \mathbf{U}^*)}.$$

The ratio $p(\mathbf{O}^t)/p(\mathbf{O}^*) = 1$ vanishes from the acceptance probability. Sampling from $p(\cdot|\mathbf{x}_{1:W}, c, \mathbf{m}, \mathbf{U}^*)$ in the step 2 is carried out as described in Section 2.1, and $p(\cdot|\mathbf{x}_{1:W}, c, \mathbf{m}, \mathbf{U})$ is evaluated as in expression (3). Continuing the use of the Haar-based decomposition $\mathbf{U} = \mathbf{M}\mathbf{\Lambda}^{1/2}\mathbf{O}$ during sampling improves mixing across all test cases we tried. Notice that since $(\mathbf{m}, \mathbf{S})$ are fixed and the elements of the spectral decomposition of $\mathbf{S}$ no longer need be computed, the sampling portion of the algorithm is necessarily much faster than the warming-up phase.

4 Test cases

We evaluate the performance of the PTS on several test cases. To keep the exposition focused, only the basic algorithm is considered except for Section 4.1. The approximations obtained after the adaptive phase, $g(\cdot)$, are plotted for visual comparison in the bivariate cases. The PTS is compared to standard Metropolis-within-Gibbs approaches and an implementation of delayed rejection adaptive metropolis (DRAM, Haario et al., 2006), which demonstrably combines the best aspects of delayed rejection (Tierney & Mira, 1999) and adaptive Metropolis (Haario et al., 2001). Details of DRAM are given in the online supplementary material (see Appendix A), along with autocorrelation plots (see Appendix B). Effective sample sizes (ESS) are compared across the approaches. Sargent et al. (2000) use an ESS defined by $\text{ESS} = n/\tau$, where n is the actual number of MCMC iterates kept after

burn-in, $\tau = \sum_{i=0}^k \hat{\rho}(i)$, $k = \min\{i: \hat{\rho}(i) < 0.1\}$, and $\hat{\rho}(i)$ is the sample autocorrelation at lag i . The ESS is an estimate of the equivalent number of *iid* samples the MCMC generated. Necessarily, $\text{ESS} \leq n$ with equality holding only when (assuming monotonic decay) $\hat{\rho}(i) < 0.1$ for $i \geq 1$. Although we take many more MCMC iterates, the ESS is reported as the effective sample size out of $n = 10,000$, a typical number that might be used under *iid* sampling.

In theory DRAM does not require burn-in, and as implemented by Haario et al. (2006), all iterates are kept. However, we have found this can negatively affect the performance of DRAM in the test cases and, therefore, take a burn-in for both DRAM and Gibbs sampling to place these approaches in the best possible light, except for Appendix F (online supplementary material). All examples except for Section 4.3 use $W = 300$, $J = 5$, $N = 500$, and $L_1 < 0.3$ as a stopping rule. At least $n = 100,000$ MCMC iterates were kept in each case. Burn-in varied across the cases, but was always at least 10,000; burn-in is not required of the PTS.

A function implementing the PTS described here, PTsampler, was written in a compiled language and incorporated into the R (R Development Core Team, 2009) library “DPpackage” (Jara, 2007). This library, including the source files, is available from Comprehensive R Archive Network (CRAN) at <http://cran.r-project.org/>. A description of the PTsampler function is given in Appendix C (online supplementary material). R codes illustrating the usage of this function are given in Appendix D (online supplementary material). Note, however, that the PTsampler function was not used for analyses in this paper in order to compare competing approaches on common ground.

4.1 Dog bowl in $\mathbb{R}^2$

Consider a density shaped like a dog bowl or donut in $\mathbb{R}^2$ given by

$$f(x, y) = (2\pi)^{-3/2} \exp\{-0.5(\sqrt{x^2 + y^2} - 10)^2 (x^2 + y^2)^{-1/2}\},$$

which follows from letting $r \sim N(10, 1)$ independent of $t \sim U(0, 2\pi)$ and taking $\tan(t) = y/x$ and $r^2 = x^2 + y^2$. We tried two sets of initial support points for the PTS: (a) overdispersed

but centered at the true density, $\mathbf{x}_1, \dots, \mathbf{x}_{300} \stackrel{iid}{\sim} U(-15, 15) \times U(-15, 15)$, and (b) rather focused, but relatively distant from the bulk of the probability mass,

$\mathbf{x}_1, \dots, \mathbf{x}_{300} \stackrel{iid}{\sim} N_2((15, 15), \mathbf{I}_2)$. For (a) and (b) the warming-up phase used 97 and 371 iterations, respectively, until $L_1 < 0.3$, followed by 500 additional iterates. Both cases had a final value $c = 1$. Figure 2 shows the final approximation $g(\cdot)$ from the two sets of starting values. The $\mathbf{x}_1, \dots, \mathbf{x}_{300}$ shown in each plot “warp” the centering $N_2(\mathbf{m}, \mathbf{S})$ distribution to give the density shown. Although not perfect, the algorithm clearly is capturing the overall shape of the density. Movies showing the progression of the support points toward the target show there is little “waste” in each adaptive iteration, available in the online supplementary material (see Appendix E for a description of the files).

Rounded to the nearest 50 and averaged over x and y , the ESS for (a) PTS with uniform, overdispersed starting values, (b) PTS with focused, wrong starting values, DRAM, and Metropolis-within-Gibbs were 2800, 2600, 500, and 400, respectively. The autocorrelation plots in Appendix C (online supplementary material) show that the PTS provides better overall mixing than DRAM or Metropolis-within-Gibbs, at least as implemented here. In general, starting values that are overdispersed relative to the target $f(\cdot)$ can accelerate convergence in the approximation of $g(\cdot)$ to $f(\cdot)$ as compared to starting values that are highly focused in an area away from the bulk of $f(\cdot)$ mass. We have also found that for $f(\cdot)$ with multiple well-separated modes, if support points fall into all areas of non-negligible mass under $f(\cdot)$, the full function will be approximated. Otherwise the approximation will tend to a subset of modal regions corresponding to where the initial $\mathbf{x}_1, \dots, \mathbf{x}_W$ are located.

For implementing DRAM, the initial mean and covariance used in a non adaptive stage to generate $\mathbf{x}_1, \dots, \mathbf{x}_W$ were $\boldsymbol{\mu}_0 = (0, 0)'$ and $\mathbf{C}_0 = \mathbf{I}_2$, respectively. For Metropolis-within-Gibbs, a random walk proposal was used centered at the last value with standard deviation 10. We also tried $\sigma = 0.5, 1, 5, 15$; $\sigma = 10$ gave the fastest decay for the autocorrelation function (ACF) (see Appendix B, online supplementary material). Both DRAM and Metropolis-within-Gibbs were allowed a conservative burn-in of 300,000.

Several movies in the online supplementary material illustrate aspects of the algorithm for the dog bowl density. Three movies illustrate how the adaptive phase develops with static $\mathbf{O} = \mathbf{I}_2$ versus $\mathbf{O} \sim \text{Haar}(2)$, for the three values $s_c \in \{0.05, 0.1, 1\}$. Clearly, random $\mathbf{O}$ accelerates the adaptation markedly. Three more movies show the adaption phase from different starting support points in $\mathbb{R}^2$. Finally, the last movie shows one simple strategy that significantly accelerates convergence to $f(\cdot)$, at least for the dog bowl: Steps 5, 6, and 7 are modified so that instead of only one support point being updated, initially 30% of the support points are updated simultaneously at *each iteration*, allowing for much faster movement of the support point “cloud” toward the target. The percentage of support points updated is halved every 50 iterates, e.g. 30% of the support points are moved for each of iterations 1–50, 15% for 51–100, 7.5% for 100–150, etc. The movie shows successful adaptation within 200 iterations with almost no increase in computational time. This is in contrast to at least 500 iterations required when updating one support point at a time. This illustrates that the basic algorithm can be tweaked on a case-by-case basis in very simple ways to massively improve performance.

Since the target density is known and very large sample sizes were collected (as in the remaining examples too), we are confident that fully representative samples were obtained and trust the ESS values and autocorrelation plots for the donut density. However, to further compare methods we reran each of the three samplers 100 times, stopping when first and second moments were estimated within a fixed tolerance. Both the number of iterates needed and the CPU time were recorded for the dog bowl function and reported in the online supplementary material. The PTS requires significantly fewer iterates but takes much longer to estimate the moments accurately. This is undoubtedly due to the fact that $f(\cdot)$ can be evaluated almost instantaneously for this example and so the PTS adaptive phase and subsequent counting involved in the sampling phase outweighs the evaluation of $f(\cdot)$ in terms of time. This would indicate – at least for the basic algorithm – that the PTS would be most useful in sampling posteriors that are relatively cumbersome to evaluate, due to the complexity of the model, the sample size, or both; computer experiments come to mind.

To place the PTS in a more realistic light, we conducted a second simulation using a generalized linear model where $(x, y) \in \mathbb{R}^2$, as for the dog bowl, but evaluation of $f(\cdot)$ takes longer (see Appendix F, online supplementary material). For this case, the PTS outperforms Metropolis-with-Gibbs and grossly outperforms DRAM in terms of both time and numbers of iterates. This illustrates for a real, moderately-sized data set that among the three approaches, the PTS can achieve dramatic results.

4.2 Ping-pong paddle in $\mathbb{R}^2$

The mixture of three bivariate normal densities considered by Gilks et al. (1998) is used here. The mixture is given by

$$0.34N_2(\mathbf{0}, \mathbf{I}_2) + 0.33N_2\left(\begin{bmatrix} -3 \\ -3 \end{bmatrix}, \begin{bmatrix} 1 & 0.9 \\ 0.9 & 1 \end{bmatrix}\right) + 0.33N_2\left(\begin{bmatrix} 2 \\ 2 \end{bmatrix}, \begin{bmatrix} 1 & -0.9 \\ -0.9 & 1 \end{bmatrix}\right).$$

The plot looks something like a ping-pong paddle and ball, and so we call it the ping-pong density for short. We refer to Figure 3 in Gilks et al. (1998) for true density contours. Their Figure 4 shows autocorrelation plots for a particular PAS and their regeneration scheme.

Again, two sets of initial support points were used for the PTS: (a) overdispersed but centered at the true density: $\mathbf{x}_1, \dots, \mathbf{x}_{300} \stackrel{iid}{\sim} U(-6, 6) \times U(-6, 6)$ and (b) rather focused, but relatively distant from the bulk of the probability mass $\mathbf{x}_1, \dots, \mathbf{x}_{300} \stackrel{iid}{\sim} N_2\{(\pm 6, 6), \mathbf{I}_2\}$. We obtain an effective sample sizes of 3900 and 3900 for the PTS (a) and (b), respectively, 700 for DRAM, and 50 for Metropolis-within-Gibbs. The autocorrelation plots (online supplementary material) shows Gibbs sampling to be relatively inefficient compared to DRAM and PTS, which is not surprising given the high correlation in target density $f(\cdot)$. The number of adaptive updates before $L_1 < 0.3$ was 88 for (a) and 274 for (b), respectively. In both cases $N = 500$ iterations were additionally used. The same initial μ_0 and $\mathbf{C}_0$ were used for DRAM as in Section 4.1. Although the full conditionals are a mixture of normals, we implemented a Metropolis-within-Gibbs approach with univariate Gaussian random-walk proposals having standard deviation $\sigma = 10$.

Figure 3 shows the approximating densities and final support points, after the warming-up phase, from the two sets of starting values. This example shows an interesting aspect of the approximation: to capture the true density shape, the support points go beyond the region of highest density. This is because the support points warp *beyond* the bivariate Gaussian centering distribution's shape. That is, support points should not necessarily be viewed as

data that are representative of $f(\cdot)$, but rather where one might have to hit a surface that's already Gaussian-shaped to obtain the desired $f(\cdot)$.

4.3 Correlated multivariate normal

We consider $f(\cdot) = \varphi_d(\cdot | \mathbf{0}, \Sigma)$, where Σ has diagonal elements 1 and off-diagonal elements 0.9. Dimensions $d = 3, 5, 10, 15$, and 20 were considered. We tried fixing $W = 300$ and $N = 500$ as in the previous examples (PT1), but also tried increasing both with the dimensionality. Specifically we used $W = d(100)$ and $N = 200 + W$ (PT2), obtaining quite different results. Table 1 shows that fixed $W = 300$ and follow-up $N = 500$ does about as well as Gibbs and DRAM; ACF plots are in the online supplemental material. Having W and N increase with the dimensionality shows *drastic* improvement over the other approaches. In multiple runs of the PTS, the ESS and autocorrelation varied but stayed within an order of magnitude within the values shown (see Appendix B, online supplementary material).

4.4 Nonlinear regression

Carlin & Gelfand (1991) analyzed data on the length y_i and age x_i of $i = 1, \dots, 27$ dugongs (an aquatic mammal) through the model

$$y_i | \alpha, \beta, \gamma, \sigma \stackrel{\text{ind.}}{\sim} N(\alpha - \beta\gamma^{x_i}, \sigma^2), \quad i=1, \dots, 27.$$

These data and model are also in the WinBUGS Examples volume II. Fitting the model using any of the methods exhibits a relatively strong, nonlinear relationship between α and γ in the posterior. These parameters also display autocorrelation that dies down relatively slowly using WinBUGS; in contrast β and σ mix well.

Here we considered the Gibbs sampling algorithm as implemented in BUGS (Gilks et al., 1992; Spiegelhalter et al., 2003). For β , but especially σ , the Gibbs sampler outperforms PTS and DRAM according to Table 2. PTS MCMC mixing is about the same for all four parameters, reflecting that an entire proposed vector of parameters $(\alpha^*, \beta^*, \gamma^*, \sigma^*)$ is either accepted or rejected simultaneously (see Appendix B, online supplementary material). DRAM also rejects or accepts the entire proposed vector of values, but fares worse than PTS. DRAM fares about the same as Gibbs sampling for the parameters α and γ ; these show moderate autocorrelation under DRAM and Gibbs sampling, and have a relatively strong, nonlinear relationship in the posterior $f(\cdot)$. PTS has less trouble with these parameters because the proposed vectors from $g(\cdot)$ have this nonlinear relationship built-in.

The starting mean and covariance for DRAM were provided by univariate posterior means and variances, $\mu_0 = (2.66, 0.974, 0.865, 101)$ and $\mathbf{C}_0 = \text{diag}(0.0748^2, 0.0773^2, 0.0327^2, 30^2)$, respectively. These were also used to obtain *iid* starting support points $\mathbf{x}_1, \dots, \mathbf{x}_{300}$. Only 87 adaptive iterations were performed before $L_1 < 0.3$, followed by 500 more iterations.

5 Concluding remarks

We present a simple and efficient sampling method for exploring an unnormalized multivariate density $f(\cdot)$ with support $\mathbb{R}^d$. The algorithm constructs an independent proposal based on a Bayesian nonparametric approximation of the target density: the predictive density from a multivariate PT. To the best of our knowledge, this is the first application of a Bayesian nonparametric prior to the exploration of posterior distributions. Aspects of the PT prior make is especially attractive in this light: the marginalized version has virtually no curse of dimensionality, the predictive distribution is computed through simple counting of

observations in sets, and the prior is easily centered at the Gaussian distribution, which approximately holds, at least in part, for large classes of posteriors.

The precision parameter of the PT, c , and the W support points are the parameters of the approximation, which is smoothed by the decomposition $\mathbf{S} = \mathbf{U}\mathbf{U}'$, yielding a mixture of finite PT. The approach can be used to approximate quite non-normal densities, as evidenced by the dog bowl and mixture of normals examples. The basic PTS discussed here is built from a set of initial support points and is not an adaptive MCMC algorithm in the sense of, for instance, Haario et al. (2001) and Haario et al. (2006). Therefore standard theory on Markov chains generated from Metropolis with independent proposals applies. The PTS can grossly outperform component-at-a-time samplers and the implementation of DRAM presented by Haario et al. (2006).

Several improvements of the basic are immediate. It seems likely that further fine-tuning of the algorithm allowing for adaptive W will improve the algorithm's performance. Further, fully adaptive algorithms in the spirit of Haario et al. (2001) and Ji & Schmidler (2009) can be pursued. In this setup, accepted candidates in the sampling phase can be used in order to further improve the quality of the approximation to the target density. In this case, the mean $\mathbf{m}$ and covariance $\mathbf{S}$ could be updated every n iterations, speeding up the process considerably. Alternatively, as suggested by Gilks et al. (1998), $g(\cdot)$ could be continually updated at regeneration times.

The evaluation of the proposed approximation in different context is the subject of future research. For instance, $g(\cdot)$ could be used as the first stage in a delayed rejection scheme as suggested by Tierney & Mira (1999). The approximation $g(\cdot)$ captures broad aspects of the unknown $f(\cdot)$ and is sampled in the first stage; if rejected, finer detail can be captured by a second random walk proposal, e.g. $\mathbf{x}_j^* \sim N_d(\mathbf{x}_{j-1}, \gamma\mathbf{S})$ where $\gamma < 1$. This helps obviate poor mixing, especially in the tails where independence proposals typically fail. Finally, the comparison of our PTS to other samplers is the subject of future research.

This paper embarks on an admittedly ambitious undertaking: to quickly approximate unnormalized densities in a black box manner. Although not providing a perfect solution (in which $g(\cdot)$ would approximate $f(\cdot)$ up to some pre-specified error bound and provide essentially *iid* samples), even the simplest implementation of the PTS can grossly outperform component-at-a-time samplers and the implementation of DRAM presented by Haario et al. (2006). It seems likely that further fine-tuning of the algorithm allowing for adaptive sampling at levels $j \in \{1, \dots, J\}$, adaptive W , or updating a percentage of support points at each iteration will improve the algorithm's performance. Further, minor modifications of the basic algorithm yield obvious fully adaptive strategies. Alone, or combined with other strategies such as regeneration, delayed acceptance, or tempering will likely yield very powerful, fast algorithms for large classes of problems.

Supplementary Material

Refer to Web version on PubMed Central for supplementary material.

Acknowledgments

The first and second authors were supported in part by NIH grant 2-R01-CA95955-05. The third author was supported by Fondecyt grant 3095003.

References

- Anderson TW, Olkin I, Underhill LG. Generation of random orthogonal matrices. *SIAM Journal on Scientific and Statistical Computing*. 1987; 8:625–629.
- Atchadé YF, Rosenthal JS. On adaptive Markov chain Monte Carlo algorithms. *Bernoulli*. 2005; 11:815–828.
- Berger JO, Guglielmi A. Bayesian and conditional frequentist testing of a parametric model versus nonparametric alternatives. *Journal of the American Statistical Association*. 2001; 96:174–184.
- Bernardo, JM.; Smith, AFM. *Bayesian Theory*. Wiley; Chichester, West Sussex: 2000.
- Cai B, Meyer R, Perron F. Metropolis-Hastings algorithms with adaptive proposals. *Statistics and Computing*. 2008; 18:421–433.
- Cappé O, Douc R, Guillin A, Marin JM, Robert CP. Adaptive importance sampling in general mixture classes. *Statistics and Computing*. 2008; 18:447–549.
- Carlin BP, Gelfand AE. An iterative Monte Carlo method for nonconjugate Bayesian analysis. *Statistics and Computing*. 1991; 1:119–128.
- Christensen R, Hanson T, Jara A. Parametric nonparametric statistics: An introduction to mixtures of finite Polya trees. *The American Statistician*. 2008; 62:296–306.
- Gelfand AE, Sahu SK, Carlin BP. Efficient parameterizations for normal linear mixed models. *Biometrika*. 1995; 82:479–488.
- Gelfand, AE.; Sahu, SK.; Carlin, BP. Efficient parameterizations for generalized linear mixed models. In: Bernadro, JM.; Berger, JO.; Dawid, AP.; SAFM, editors. *Bayesian Statistics*. Vol. 5. Oxford, UK: Oxford University Press; 1996. p. 165–180.
- Gelfand AE, Smith AFM. Sampling-based approaches to calculating marginal densities. *Journal of the American Statistical Association*. 1990; 85:398–409.
- Geyer CJ, Thompson EA. Annealing Markov chain Monte Carlo with applications to ancestral inference. *Journal of the American Statistical Association*. 1995; 90:909–920.
- Gilks WR, Best NG, Tan KKC. Adaptive rejection Metropolis sampling. *Applied Statistics*. 1995; 44:455–472.
- Gilks WR, Roberts GO, Sahu SK. Adaptive Markov chain Monte Carlo through regeneration. *Journal of the American Statistical Association*. 1998; 93:1045–1054.
- Gilks WR, Thomas A, Spiegelhalter DJ. Software for Gibbs sampler. *Computing Science and Statistics*. 1992; 24:439–448.
- Giordani P, Kohn R. Adaptive independent Metropolis-Hastings by fast estimation of mixtures of normals. *Journal of Computational and Graphical Statistics*. 2010 in press.
- Haario H, Laine M, Mira A, Saksman E. Dram: Efficient adaptive MCMC. *Statistics and Computing*. 2006; 16:339–354.
- Haario H, Saksman E, Tamminen J. An adaptive Metropolis algorithm. *Bernoulli*. 2001; 7:223–242.
- Hanson T. Inference for mixtures of finite Polya tree models. *Journal of the American Statistical Association*. 2006; 101:1548–1565.
- Hanson, T. Polya trees and their use in reliability and survival analysis. In: Ruggeri, F.; Kenett, R.; Faltin, FW., editors. *Encyclopedia of Statistics in Quality and Reliability*. Chichester, UK: John Wiley and Sons; 2007. p. 1385–1390.
- Hanson T, Branscum A, Gardner I. Multivariate mixtures of Polya trees for modelling ROC data. *Statistical Modelling*. 2008; 8:81–96.
- Holden L, Hauge R, Holden M. Adaptive independent Metropolis-Hastings. *The Annals of Applied Probability*. 2009; 19:395–413.
- Jara A. Applied Bayesian non- and semi-parametric inference using DPpackage. *Rnews*. 2007; 7:17–26.
- Jara A, Hanson T, Lesaffre E. Robustifying generalized linear mixed models using a new class of mixtures of multivariate Polya trees. *Journal of Computational and Graphical Statistics*. 2009; 18:838–860.
- Ji, C.; Schmidler, SC. Tech rep. Department of Statistics, Duke University; 2009. Adaptive Markov chain Monte Carlo for Bayesian variable selection.

- Keith JM, Kroese DP, Sofronov GY. Adaptive independence samplers. *Statistics and Computing*. 2008; 18:409–420.
- Lavine M. Some aspects of Polya tree distributions for statistical modeling. *The Annals of Statistics*. 1992; 20:1222–1235.
- Lavine M. More aspects of Polya tree distributions for statistical modeling. *The Annals of Statistics*. 1994; 22:1161–1176.
- Lundberg, M.; Svensson, L. The Haar measure and the generation of random unitary matrices. *Proceedings of the 2004 IEEE Sensor Array and Multichannel Signal Processing Workshop*; 2004. p. 114–118.
- Meyer R, Cai B, Perron F. Adaptive rejection Metropolis sampling using Lagrange interpolation polynomials of degree 2. *Computational Statistics and Data Analysis*. 2008; 52:3408–3423.
- Mezzadri F. How to generate random matrices from the classical compact groups. *Notices of the American Mathematical Society*. 2007; 54:592–604.
- Müller P, Quintana FA. Nonparametric Bayesian data analysis. *Statistical Science*. 2004; 19:95–110.
- R Development Core Team. R: A language and environment for statistical computing. R Foundation for Statistical Computing; Vienna, Austria: 2009. URL <http://www.R-project.org>
- Sargent DJ, Hodges JS, Carlin BP. Structured Markov chain Monte Carlo. *Journal of Computational and Graphical Statistics*. 2000; 9:217–234.
- Spiegelhalter, DJ.; Thomas, A.; Best, NG.; Lunn, D. WinBUGS version 1.4 user manual. MRC Biostatistics Unit; 2003.
- Stewart GW. The efficient generation of random orthogonal matrices with an application to condition estimators. *SIAM Journal on Numerical Analysis*. 1980; 17:403–409.
- Tierney L. Markov chains for exploring posterior distributions. *The Annals of Statistics*. 1994; 22:1701–1762.
- Tierney L, Mira A. Some adaptive Monte Carlo methods for Bayesian inference. *Statistics in Medicine*. 1999; 18:2507–2515. [PubMed: 10474156]
- Warnes, GR. Tech rep. Department of Statistics, University of Washington; 2001. The normal kernel coupler: An adaptive Markov chain Monte Carlo method for efficiently sampling from multi-modal distributions; p. 395

Figure 1.

Illustrations of Polya tree partitions. $B(j, k) = \{\mathbf{x} = \mathbf{m} + \mathbf{U}\mathbf{z}; \mathbf{z} \in B_0(j, k)\}$, where $\mathbf{S} = \mathbf{M}\mathbf{\Lambda}^{1/2}\mathbf{M}'$ and $\mathbf{U} = \mathbf{M}\mathbf{\Lambda}^{1/2}\mathbf{O}$. $\mathbf{S}$ has elements $s_{11} = s_{22} = 2$, $s_{12} = s_{21} = 1$. (a) Partition sets $B_0(j, k)$, (b) $\mathbf{O} = \mathbf{M}'$ so $\mathbf{U} = \mathbf{S}^{1/2}$, (c) $\mathbf{O} = \mathbf{I}_2$, (d) $\mathbf{O}$ has elements $o_{11} = o_{22} = \cos(\pi/4)$ and $o_{21} = -o_{12} = \sin(\pi/4)$. Dashed ellipses are level curves enclosing 20%, 40%, and 60% probability under $N_2(\mathbf{0}, \mathbf{I}_2)$ for (a), or $N_2(\mathbf{m}, \mathbf{S})$ for (b), (c), and (d).

(a) Starting values $\mathbf{x}_1, \dots, \mathbf{x}_{300} \stackrel{iid}{\sim} N_2\{(15, 15), \mathbf{I}_2\}$.

(b) Starting values $\mathbf{x}_1, \dots, \mathbf{x}_{300} \stackrel{iid}{\sim} U(-15, 15) \times U(-15, 15)$.

Figure 2.
Dog bowl density estimates and ending support points.

(a) Starting values $\mathbf{x}_1, \dots, \mathbf{x}_{300} \stackrel{iid}{\sim} N_2\{(6, 6), \mathbf{I}_2\}$.

(b) Starting values $\mathbf{x}_1, \dots, \mathbf{x}_{300} \stackrel{iid}{\sim} U(-6, 6) \times U(-6, 6)$.

Figure 3.
Ping-pong density estimates and ending support points.

Table 1

Correlated multivariate normal: ESS for the PolyA tree sampler with W and N fixed, PolyA tree sampler with W and N varying, Gibbs sampler and DRAM, using a d -variate correlated normal target distribution.

Sampler	$d = 3$	$d = 5$	$d = 10$	$d = 15$	$d = 20$
PolyA tree with W and N fixed	10000	6795	3049	10	10
PolyA tree with W and N varying	10000	6843	3615	4540	6900
Gibbs	631	337	143	92	67
DRAM	602	390	184	130	88

Dugongs example: Posterior summary statistics and ESS for the nonlinear model parameters under the Polya tree sampler, Gibbs sampler and DRAM.

Table 2

Sampler	Parameter	2.5%	Median	97.5%	ESS
Polya tree	α	2.532	2.646	2.798	2675
	β	0.832	0.969	1.112	2827
	γ	0.792	0.867	0.913	1706
	σ	0.073	0.093	0.124	2242
Gibbs	α	2.525	2.643	2.826	520
	β	0.827	0.974	1.135	10000
	γ	0.783	0.865	0.918	508
	σ	0.075	0.098	0.135	7948
DRAM	α	2.536	2.647	2.799	394
	β	0.836	0.972	1.120	406
	γ	0.791	0.866	0.914	357
	σ	0.072	0.093	0.125	409