Bayesian Nonparametric Mixture Models Using NIMBLE

Claudia Wehrhahn
Department of Statistics
University of California, Santa Cruz
Santa Cruz, CA, 95064
cwehrhah@ucsc.edu

Abel Rodríguez
Department of Statistics
University of California, Santa Cruz
Santa Cruz, CA, 95064
abel@soe.ucsc.edu

Christopher J. Paciorek
Department of Statistics
University of California, Berkeley
Berkeley, CA, 94720
paciorek@stat.berkeley.edu

Abstract

This paper describes and illustrates new features of the NIMBLE computing environment (de Valpine et al., 2017) that enable simulation-based inference for general nonparametric Bayesian mixture models.

1 Introduction

The introduction of Markov chain Monte Carlo (MCMC) algorithms in the statistical literature (Gelfand & Smith, 1990) revolutionized the discipline by enabling the application of Bayesian methods to increasingly complex models. Approaches such as Gibbs sampling, random walk Metropolis-Hastings or Hamiltonian Monte Carlo provide general templates to derive algorithms that are applicable to large classes of statistical models. However, because of their generality, the application of these templates to specific problems can be time-consuming, as they require tedious model-specific derivations to specialize them to particular problems. This issue is compounded in the case of algorithms for nonparametric Bayesian models, which require careful bookeeping and the manipulation of infinite dimensional objects. This often means that only practicioners with specialized training in statistics and/or machine learning are able to succesfully apply non-parametric Bayesian methods in their work.

The challenge of enabling a more general audience to use Bayesian models has been traditionally tackled in two distinct ways. Starting with WinBUGS (Sturtz et al., 2005), a number of software packages have been introduced to enable automated inference for general statistical models. Examples include OpenBUGS (Lunn et al., 2009), JAGS (Plummer et al., 2003), Stan (Carpenter et al., 2017), Edward (Tran et al., 2017) and Turing (Ge et al., 2018). In particular, both Eward and Turing provide support for nonparametric Bayesian models. This type of software combines a probabilistic programming language that allows users to specify general hierarchical models, with a system that assigns inferential algorithms from among a few options following simple rules. A key shortcoming of this approach is that it usually restricts the type of algorithms that can be used for any specific model, potentially leading to suboptimal choices. An alternative approach involves the design of specialized packages that focus on very specific models and their associated algorithms. Examples of packages for fitting nonparametric Bayesian models include `DPpackage` (Jara et al., 2011), `BNPdensity` (Barrios et al., 2017), and `msBP` (Canale et al., 2017) for the `R` environment (R Core Team, 2018), `Bnpy` (Hughes & Sudderth, 2014) for Python (Van Rossum et al., 2007), and `BNP.jl`

Preprint. Work in progress.

(Trapp, 2015) for Julia (Bezanson et al., 2012). This type of packages usually provide very efficient algorithms, but at the cost of severely limiting the ability of the user to modify the model structure as well as the kind of algorithm that is used to fit the model.

2 Nonparametric mixtures in NIMBLE

NIMBLE (de Valpine et al., 2017), short for Numerical Inference for statistical Models using Bayesian and Likelihood Estimation, is a system for building and sharing analysis methods for statistical models built on top of the `R` statistical software. It aims to provide the user as much flexibility as possible, both in terms of model specification as well as in terms of the computational algorithm to be employed for inference, while preserving user-friendly interfaces. While NIMBLE provides support for both frequentist and Bayesian inference, in this document we focus on its ability to enable simulation-based posterior Bayesian inference.

NIMBLE involves a declarative model-language that is a dialect of the BUGS language. The language allows various parametrizations of distributions in the declaration of the model, as well as user-defined distributions and functions. Additionally, the sampler assignment can be highly customized by the user, including by incorporating user-defined sampling algorithms. This high degree of customizability is one of NIMBLE's distinct features, making it not only a great tool for black-box implementation of various statistical models, but also an excellent platform for research in computational methods for statistical models. Another distinct feature of NIMBLE is that it includes a compiler that is used on model definitions and algorithms to generate and compile C++ code in order to speed up computation. You can find more details about NIMBLE at `https://r-nimble.org`.

We have recently added support for BNP mixture modeling to NIMBLE (see Section 10 of the NIMBLE manual at `https://r-nimble.org/manuals/NimbleUserManual.pdf`). The current implementation provides support for hierarchical models involving Dirichlet process (DP) mixture priors (Ferguson, 1973, 1974; Lo, 1984; Escobar, 1994; Escobar & West, 1995). The simplest such model is

$$y_i \mid \theta_i, \phi \overset{ind}{\sim} \psi(y_i \mid \theta_i, \phi), \quad \theta_i \mid G \overset{ind}{\sim} G, \quad G \mid \alpha, H_\eta \sim DP(\alpha, H_\eta), \quad i = 1, \ldots, n, \tag{1}$$

where $\psi(\cdot \mid \theta, \phi)$ is a suitable kernel that depends on random effects θ and fixed effects ϕ. α is the concentration parameter of the Dirichlet process prior, and H_η is a parametric base distribution indexed by the vector of parameters η. More sophisticated versions of the model include additional levels in the hierarchy, such as priors for the hyperparameters ϕ, α and η.

Our implementation of models involving DP mixtures uses the Chinese Restaurant Process (CRP) representation (Blackwell & MacQueen, 1973; Pitman, 1995, 1996). Introducing a vector of auxiliary variables $z = (z_1, \ldots, z_n)$ that indicate which component of the mixture generated each observation, and integrating over the random measure G, the model in (1) can be rewritten as

$$y_i \mid z_i, \phi, \tilde{\theta}_1, \tilde{\theta}_2, \cdots \overset{ind}{\sim} \psi(y_i \mid \tilde{\theta}_{z_i}, \phi), \quad z \mid \alpha \sim \text{CRP}(\alpha), \quad \tilde{\theta}_j \mid \eta \overset{iid}{\sim} H_\eta, \quad i = 1, \ldots, n, \tag{2}$$

where $\text{CRP}(\alpha)$ denotes the CRP distribution with concentration parameter α, with probability mass function

$$p(z \mid \alpha) = \frac{\Gamma(\alpha)}{\Gamma(\alpha + n)} \alpha^{K(z)} \prod_k \Gamma\left(m_k(z)\right), \tag{3}$$

where $K(z) \leq n$ is the number of unique values in the vector z, and $m_k(z)$ is the number of times the k-th unique value appears in z.

Alternatively, DP mixture models can be specified in NIMBLE using a (truncated) stick-breaking representation of the random distribution G (Sethuraman, 1994):

$$y_i \mid \xi_i, v, \theta_1^\star, \theta_2^\star, \ldots \overset{ind}{\sim} \psi(y_i \mid \theta_{\xi_i}^\star, \phi), \quad \Pr(\xi_i = l \mid v) = v_l \prod_{m<l} (1 - v_m), \quad i = 1, \ldots, n, \tag{4}$$

where $v_l \mid \alpha \overset{iid}{\sim} \text{Beta}(1, \alpha)$ for $l = 1, \ldots, L-1$ and $v_L = 1$, while $\theta_l^\star \overset{iid}{\sim} H_\eta$ for $l = 1, \ldots, L$.

Each of the two representations leads to a different default choice for the MCMC algorithm. Formulations based on the CRP representation use a collapsed Gibbs sampler (Neal, 2000). Specifically, either algorithm 2 or algorithm 8 from (Neal, 2000) is used depending on whether ψ and H form a conjugate pair or not. When the truncated stick-breaking representation is used, a blocked Gibbs sampler is used (Ishwaran & James, 2001, 2002).

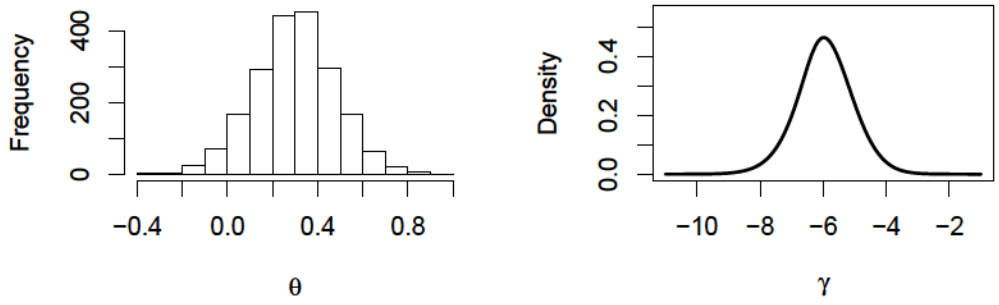

Figure 1: Left: Posterior distribution of the effect of Avandia on MI. Right: Nonparametric estimate of the random effects distribution

3 Illustration

We illustrate the use of NIMBLE for fitting nonparametric models in the context of a meta-analysis of the side effects of a formerly very popular drug for diabetes called Avandia. Here we only present the main highlihgts, a full description of the code used for this example can be found on the online suplementary materials. The question of interest is whether Avandia use increases the risk of myocardial infarction (heart attack). There are 48 studies (the 49th study in the data file is different in some ways and excluded here), each with treatment and control arms. The data corresponds to four vectors: n and x contain, respectively, the total number of patients and the number suffering from myocardial infarctions in the control group of each study, while vectors m and y contain similar information for patients receiving the drug Avandia. The model for the data takes the form

$$x_i \mid \theta, \gamma_i \sim \text{Bin}\left(n_i, \frac{1}{1+\exp\{-\gamma_i\}}\right), \qquad y_i \mid \theta, \gamma_i \sim \text{Bin}\left(m_i, \frac{1}{1+\exp\{-(\theta+\gamma_i)\}}\right),$$

for $i = 1, \ldots, 49$. The random effects $\gamma_1, \ldots, \gamma_J$ follow a Dirichlet process mixture with Gaussian kernels and a product of Gaussian and inverse gamma base distributions. The parameter θ (which is the fixed effect quantifying the difference in risk between the control and treatment arms) is given a flat prior. The following `nimbleCode` function provides the specification of the model using the CRP representation of the model:

```
codeBNP <- nimbleCode({
    for(i in 1:I) {
        y[i] ~ dbin(size = m[i], prob = q[i]) # avandia MIs
        x[i] ~ dbin(size = n[i], prob = p[i]) # control MIs
        q[i] <- expit(theta + gamma[i])       # Avandia log-odds
        p[i] <- expit(gamma[i])               # control log-odds
        gamma[i] ~ dnorm(mu[i], var = tau[i])
        mu[i] <- muTilde[xi[i]]
        tau[i] <- tauTilde[xi[i]]
        muTilde[i] ~ dnorm(mu0, sd = sd0)
        tauTilde[i] ~ dinvgamma(a0, b0)}
    xi[1:I] ~ dCRP(alpha, size = I)
    alpha ~ dgamma(1, 1)
    mu0 ~ dflat()
    sd0 ~ dunif(0, 100)
    a0 ~ dunif(0, 100)
    b0 ~ dunif(0, 100)
    theta ~ dflat()})
```

Most of the model specification above uses standard NIMBLE distributions. The main addition is the function `dCRP`, which assigns the CRP prior in (3) to the vector of indicators `xi`. Model compilation and execution then proceeds in the same way as for any other NIMBLE model. Since the CRP representation of a nonparametric mixture model integrates out the random mixing measure, general inferences for the distribution of the random effects is not directly available from the MCMC output. To address this gap we have implemented the sampling approach described in Gelfand & Kottas (2002) in the function `getSamplesDPmeasure()`. This function takes as its argument a compiled or uncompiled MCMC object, and generates samples for the weights and atoms of (a truncated version of) the underlying random measure.

Using the code presented in the online supplement we generate Figures 1. Note that the posterior for θ is a unimodal, symmetric distribution centered around 0.3 and with little mass on negative values. This suggests that there is a positive overall difference in risk between the treatment and control arms, i.e., that Avandia may increase the risk of myocardial infarction. The random effect distribution is readily estimated using the `getSamplesDPmeasure()` function, and while it shows little evidence of non-normality, doing the nonparametric analysis has ensured robustness to the random effect specification.

The same model can also be estimated using a truncated stick-breaking representation:

```
codeBNP <- nimbleCode({
    for(i in 1:I) {
        y[i] ~ dbin(size = m[i], prob = q[i]) # avandia MIs
        x[i] ~ dbin(size = n[i], prob = p[i]) # control MIs
        q[i] <- expit(theta + gamma[i])       # Avandia log-odds
        p[i] <- expit(gamma[i])               # control log-odds
        gamma[i] ~ dnorm(mu[i], var = tau[i])
        xi[i] ~ dcat(w[1:L])
        mu[i] <- muTilde[xi[i]]
        tau[i] <- tauTilde[xi[i]]
        muTilde[i] ~ dnorm(mu0, sd = sd0)
        tauTilde[i] ~ dinvgamma(a0, b0)}
    w[1:L] <- stick_breaking(v[1:(L-1)])
    for(i in 1:(L-1)){
        v[i] ~ dbeta(1, alpha)}
    alpha ~ dgamma(1, 1)
    mu0 ~ dflat()
    sd0 ~ dunif(0, 100)
    a0 ~ dunif(0, 100)
    b0 ~ dunif(0, 100)
    theta ~ dflat()})
```

The function `stick_breaking()` builds the stick breaking weights from the stick-breaking ratios contained in the vector `v`. Note that the construction is general and does not depend on the fact that the stick-breaking ratios are Beta distributed. Hence, the function allows for the implementation of more general stick-breaking priors.

4 Future work

We are currently working to extend NIMBLE to accommodate more general nonparametric priors and their associated species sampling models (e.g., Poisson-Dirichlet process, normalized random measures), as well as models for collections of distributions (e.g., hierarchical Dirichlet processes).

5 Acknowledgments

The authors were supported by Center for Research in Open Source Software at UCSC and by NSF/DMS-1622444 and.NSF/CCF-1740850.

References

BARRIOS, E., LIJOI, A., NIETO-BARAJAS, L. E. & PRÜNSTER, I. (2017). *BNPdensity: Ferguson-Klass Type Algorithm for Posterior Normalized Random Measures*. R package version 2017.03, URL `https://CRAN.R-project.org/package=BNPdensity`.

BEZANSON, J., KARPINSKI, S., SHAH, V. B. & EDELMAN, A. (2012). Julia: A fast dynamic language for technical computing. *arXiv preprint arXiv:1209.5145* .

BLACKWELL, D. & MACQUEEN, J. (1973). Ferguson distributions via Pólya urn schemes. *The Annals of Statistics* 1 353–355.

CANALE, A. ET AL. (2017). msBP: An R package to perform Bayesian nonparametric inference using multiscale Bernstein polynomials mixtures. *Journal of Statistical Software* 78 1–19.

CARPENTER, B., GELMAN, A., HOFFMAN, M. D., LEE, D., GOODRICH, B., BETANCOURT, M., BRUBAKER, M., GUO, J., LI, P. & RIDDELL, A. (2017). Stan: A probabilistic programming language. *Journal of Statistical Software* 76.

DE VALPINE, P., TUREK, D., PACIOREK, C. J., ANDERSON-BERGMAN, C., LANG, D. T. & BODIK, R. (2017). Programming with models: writing statistical algorithms for general model structures with NIMBLE. *Journal of Computational and Graphical Statistics* 26 403–413.

ESCOBAR, M. D. (1994). Estimating normal means with a Dirichlet process prior. *Journal of the American Statistical Association* 89 268–277.

ESCOBAR, M. D. & WEST, M. (1995). Bayesian density estimation and inference using mixtures. *Journal of the American Statistical Association* 90 577–588.

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

FERGUSON, T. S. (1974). Prior distribution on the spaces of probability measures. *Annals of Statistics* 2 615–629.

GE, H., XU, K. & GHAHRAMANI, Z. (2018). Turing: Composable inference for probabilistic programming. In *International Conference on Artificial Intelligence and Statistics*. 1682–1690.

GELFAND, A. E. & KOTTAS, A. (2002). A computational approach for full nonparametric bayesian inference under dirichlet process mixture models. *Journal of Computational and Graphical Statistics* 11 289–305.

GELFAND, A. E. & SMITH, A. F. (1990). Sampling-based approaches to calculating marginal densities. *Journal of the American Statistical Association* 85 398–409.

HUGHES, M. C. & SUDDERTH, E. B. (2014). Bnpy: Reliable and scalable variational inference for bayesian nonparametric models. In *NIPS Probabilistic Programimming Workshop*.

ISHWARAN, H. & JAMES, L. F. (2001). Gibbs sampling methods for stick-breaking priors. *Journal of the American Statistical Association* **96** 161–173.

ISHWARAN, H. & JAMES, L. F. (2002). Approximate Dirichlet process computing in finite normal mixtures: Smoothing and prior information. *Journal of Computational and Graphical Statistics* **11** 508–532.

JARA, A., HANSON, T. E., QUINTANA, F. A., MÜLLER, P. & ROSNER, G. L. (2011). DPpackage: Bayesian semi-and nonparametric modeling in R. *Journal of Statistical Software* 40 1.

LO, A. Y. (1984). On a class of Bayesian nonparametric estimates I: Density estimates. *The Annals of Statistics* 12 351–357.

LUNN, D., SPIEGELHALTER, D., THOMAS, A. & BEST, N. (2009). The BUGS project: Evolution, critique and future directions. *Statistics in Medicine* 28 3049–3067.

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

PITMAN, J. (1995). Exchangeable and partially exchangeable random partitions. *Probability Theory and Related Fields* **102** 145–158.

PITMAN, J. (1996). Some developments of the Blackwell-MacQueen urn scheme. In T. S. Ferguson, L. S. Shapeley & J. B. MacQueen, eds., *Statistics, Probability and Game Theory. Papers in Honor of David Blackwell*. Hayward, CA:IMS, 245–268.

PLUMMER, M. ET AL. (2003). JAGS: A program for analysis of Bayesian graphical models using Gibbs sampling. In *Proceedings of the 3rd International Workshop on Distributed Statistical Computing*, vol. 124. Vienna, Austria.

R CORE TEAM (2018). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL `https://www.R-project.org/`.

SETHURAMAN, J. (1994). A constructive definition of Dirichlet prior. *Statistica Sinica* 2 639–650.

STURTZ, S., LIGGES, U. & GELMAN, A. (2005). R2winBUGS: A package for running WinBUGS from R. *Journal of Statistical Software* 12 1–16. URL `http://www.jstatsoft.org`.

TRAN, D., HOFFMAN, M. D., SAUROUS, R. A., BREVDO, E., MURPHY, K. & BLEI, D. M. (2017). Deep probabilistic programming. *arXiv preprint arXiv:1701.03757* .

TRAPP, M. (2015). Bnp.jl: Bayesian nonparametrics in julia. In *Bayesian Nonparametrics: The Next Generation Workshop at NIPS*.

VAN ROSSUM, G. ET AL. (2007). Python programming language. In *USENIX Annual Technical Conference*, vol. 41. 36.