Categorical inputs, sensitivity analysis, multiresolution models, optimization and importance tempering with tgp version 2, an R package for treed Gaussian process models

Matt A. Taddy
Applied Math & Statistics Department
University of California, Santa Cruz, USA
taddy@ams.ucsc.edu

Robert B. Gramacy
Statistical Laboratory
University of Cambridge, United Kingdom
bobby@statslab.cam.ac.uk

April 7, 2008

Abstract

This document describes the new features in version 2.0 of the tgp package for R, implementing treed Gaussian process (GP) models. The topics covered include methods for dealing with categorical inputs and excluding inputs from the tree or GP part of the model; fully Bayesian sensitivity analysis for inputs/covariates; multiresolution (treed) Gaussian process modeling; sequential optimization of black-box functions; and a new Monte carlo method for inference in multi-modal posterior distributions that combines simulated tempering and importance sampling. These additions extend the functionality of tgp across all models in the hierarchy: from Bayesian linear models to CART to treed Gaussian process with jumps to the limiting linear model, except in the case of multiresolution models which apply only to the (treed) GP. It is assumed that the reader is familiar with the features in earlier versions of the package, outlined in the first vignette tgp.pdf [9].

Intended audience

This document is intended as a follow-on to the vignette for version 1.x of the tgp package [9]. It describes five new features that have been added to the package in version 2.x, and is divided into five essentially disjoint sections: on categorical inputs (Section 1), sensitivity analysis (Section 2), multiresolution `tgp` (Section 3), sequential optimization (Section 4), and importance tempering (Section 5). Each section will begin with a short mathematical introduction to the new feature or methodology and commence with extensive examples in R on synthetic and real data. This document has been authored in `Sweave` (try `help(Sweave)`). This means that the code quoted throughout is certified by `R`, and the `Stangle` command can be used to extract it. As with the first vignette, the R code in each of the sections to follow is also available as a demo in the package. Note that this tutorial was not meant to serve as an instruction manual. For more detailed documentation of the functions contained in the package, see the package help–manuals. At an `R` prompt, type `help(package=tgp)`. PDF documentation is also available on the world-wide-web.

`http://www.cran.r-project.org/doc/packages/tgp.pdf`

Throughout this document it is assumed that the reader is familiar with the `tgp` methods and models available in version 1.x of the package, outlined in the vignette `tgp.pdf` [9].

1 Non–real–valued, categorical and other inputs

Early versions of `tgp` worked best with real–valued inputs $\mathbf{X}$. While it was possible to specify ordinal, integer–valued, or even binary inputs, `tgp` would treat them the same as any other real–valued input. Two new arguments to `tgp.default.params`, and thus the ellipses (`...`) argument to the `b*` functions provide a more natural way to model with non–real valued inputs.

Classical treed methods, such as CART [1], can cope quite naturally with categorical, binary, and ordinal, inputs. Categorical inputs can be encoded in binary, and splits can be proposed with rules such as $x_i < 1$. Once a split is made on a binary input, no further process in need, marginally, in that dimension. Ordinal inputs can also be coded in binary, and thus treated as categorical, or treated as real–valued and handled in a default way. GP regression, however, handles such non–real–valued inputs less naturally, unless (perhaps) a custom and non–standard form of the covariance function is used. When inputs are scaled to lie in $[0, 1]$, binary–valued inputs x_i are always a constant distance apart—at the largest possible distance in the range. A separable correlation function width parameter d_i will tend to infinity if the output does not vary with x_i, and will tend to zero if it is. Clearly, this functionality is more parsimoniously served achieved by partitioning, e.g., using a tree. However, trees with fancy regression models at the leaves pose other problems, as discussed below.

Consider as motivation, the following modification of the Friedman data [5] (see also Section 3.5 of [9]). Augment 10 real–valued covariates in the data $(\mathbf{x} = \{x_1, x_2, \dots, x_{10}\})$ with one categorical indicator $I \in \{1, 2, 3, 4\}$ that can be encoded in binary as

$$1 \equiv (0,0,0) \qquad 2 \equiv (0,0,1) \qquad 3 \equiv (0,1,0) \qquad 4 \equiv (1,0,0).$$

Not let the function that describes the responses (Z), observed with standard Normal noise, have a mean

$$E(Z|\mathbf{x}, I) = \begin{cases} 10\sin(\pi x_1 x_2) & \text{if } I = 1 \\ 20(x_3 - 0.5)^2 & \text{if } I = 2 \\ 10x_4 + 5x_5 & \text{if } I = 3 \\ 10x_1 + 5x_2 + 20(x_3 - 0.5)^2 + 10\sin(\pi x_4 x_5) & \text{if } I = 4 \end{cases} \tag{1}$$

that depends on the indicator I. Notice that when $I = 4$ the original Friedman data is recovered, but with the first five input columns in reverse order. Irrespective of I, the response depends only on $\{x_1, \ldots, x_5\}$, thus combining nonlinear, linear, and irrelevant effects. When $I = 3$ the response is linear $\mathbf{x}$.

A new function has been included in the `tgp` package which facilitates generating random realizations from (1). Below we obtain 500 such random realizations for training purposes, and a further 1000 for testing.

```
> fb.train <- fried.bool(500)
> X <- fb.train[, 1:13]
> Z <- fb.train$Y
> fb.test <- fried.bool(1000)
> XX <- fb.test[, 1:13]
> ZZ <- fb.test$Ytrue
```

A separation into training and testing sets will be useful for later comparisons by RMSE. The names of the data frame show that the first ten columns encode $\mathbf{x}$ and columns 11–13 encode the boolean representation of I.

```
> names(X)
```

```
 [1] "X.1"  "X.2"  "X.3"  "X.4"  "X.5"  "X.6"  "X.7"  "X.8"
 [9] "X.9"  "X.10" "I.1"  "I.2"  "I.3"
```

One, naïve approach to fitting this data would be to fit a treed GP LLM model ignoring the categorical inputs. But this model can only account for the noise, giving high RMSE, and so is not illustrated here. Clearly, the indicators must be included. One simple way to do so would be to posit a Bayesian CART model.

```
> fit1 <- bcart(X = X, Z = Z, XX = XX, m0r1 = TRUE,
+     verb = 0)
> rmse1 <- sqrt(mean((fit1$ZZ.mean - ZZ)^2))
> rmse1
```

```
[1] 2.983012
```

In this case the indicators are treated appropriately (as indicators), but in some sense so are the real–valued inputs as only constant models are fit at the leaves of the tree. Figure 1 shows that the tree does indeed partition on the indicators, and the other inputs, as expected.

One might expect a much better fit from a treed linear model to this data, since the response in linear in some of its inputs.

```
> tgp.trees(fit1, "map")
```

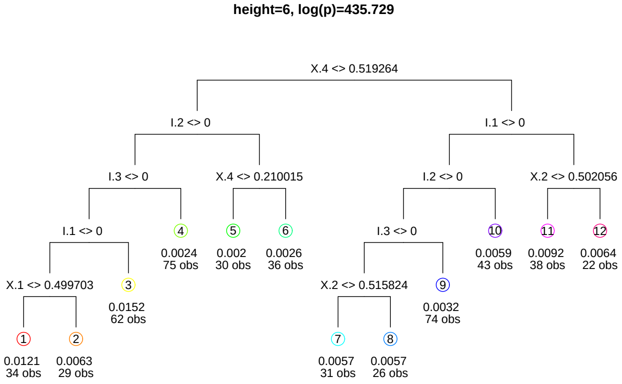

Figure 1: Diagrammatic depiction of the maximum *a' posteriori* (MAP) tree for the boolean indicator version of the Friedman data in Eq. (1) using Bayesian CART.

```
> fit2 <- btlm(X = X, Z = Z, XX = XX, m0r1 = TRUE,
+     verb = 0)
> rmse2 <- sqrt(mean((fit2$ZZ.mean - ZZ)^2))
> rmse2

[1] 2.410959
```

Unfortunately, this is not the case—-the RMSEs are similar to those for the CART model. Figure 2 shows that the tree does indeed partition, but not on the indicator variables. When a linear model is used at the leaves of the tree the boolean indicators cannot be partitioned upon because doing so would cause the design matrix to become rank–deficient at the leaves of the tree (there would be a column of all zeros or all ones). A treed GP would have the same problem.

A new feature in tgp makes dealing with indicators such as these more natural, by including them as candidates for treed partitioning, but ignoring them when it comes to fitting the models at the leaves of the tree. The argument basemax to tgp.default.params, and thus the ellipses (...) argument to the b* functions, allows for the specification of the last column of X to be considered under the base (LM or GP) model. In the context of our example, specifying basemax = 10 ensures that only the first 10 inputs, i.e., $\mathbf{X}$ only (excluding I), are used to predict the response under the GPs at the leaves. Both the columns of $\mathbf{X}$ and the columns of the boolean representation of the (categorical) indicators I are (still) candidates for partitioning. This way, whenever the boolean indicators are partitioned upon, the design matrix (for the GP or LM) will not contain the corresponding column of zeros or ones, and therefore will be of full rank.

Let us revisit the treed LM model with basemax = 10.

```
> fit3 <- btlm(X = X, Z = Z, XX = XX, basemax = 10,
> tgp.trees(fit2, "map")
```

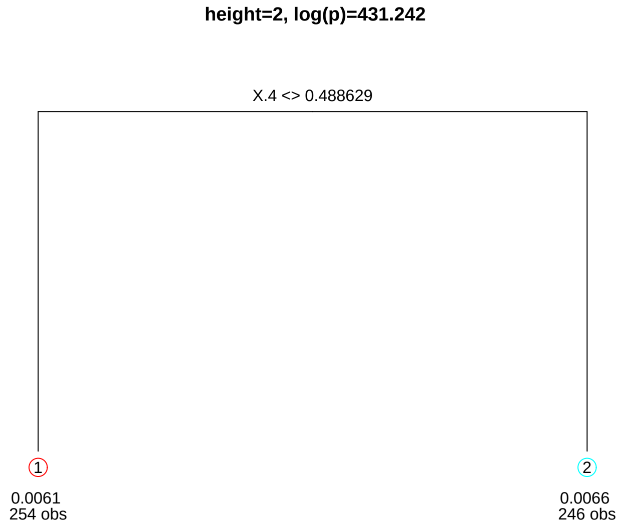

Figure 2: Diagrammatic depiction of the maximum *a' posteriori* (MAP) tree for the boolean indicator version of the Friedman data in Eq. (1) using a Bayesian treed linear model.

```
+     m0r1 = TRUE, verb = 0)
> rmse3 <- sqrt(mean((fit3$ZZ.mean - ZZ)^2))
> rmse3
```

```
[1] 1.649079
```

Figure 3 shows that the MAP tree does indeed partition on the indicators in an appropriate way—as well as on some other real–valued inputs—and the result is the lower RMSE we would expect.

A more high–powered approach would clearly be to treat all inputs as as real–valued by fitting a GP at the leaves of the tree. Binary partitions are allowed on all inputs, $\mathbf{X}$ and I, but treating the boolean indicators as real–valued in the GP is clearly inappropriate since it is known that the process does not vary smoothly over the 0 and 1 settings of the three boolean indicators representing the categorical input I.

```
> fit4 <- btgpllm(X = X, Z = Z, XX = XX, m0r1 = TRUE,
+     verb = 0)
> rmse4 <- sqrt(mean((fit4$ZZ.mean - ZZ)^2))
> rmse4
```

```
[1] 1.247021
```

Since the design matrices would become rank–deficient if the boolean indicators are partitioned upon, there was no partitioning in this example. Since there are large covariance matrices to invert, the MCMC inference is very slow.

```
> tgp.trees(fit3, "map")
```

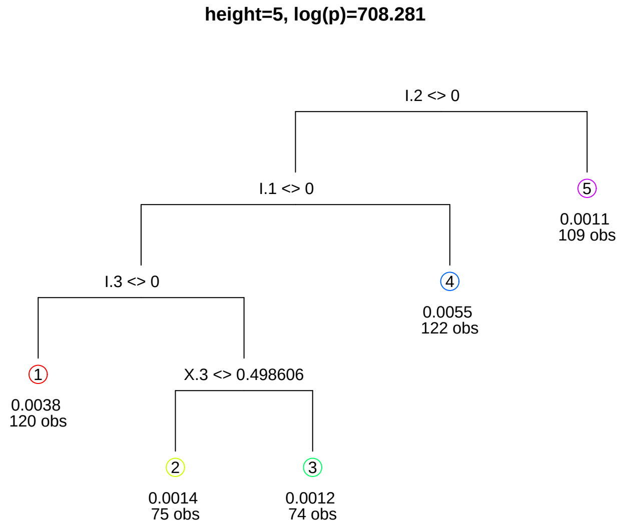

Figure 3: Diagrammatic depiction of the maximum *a' posteriori* (MAP) tree for the boolean indicator version of the Friedman data in Eq. (1) using a Bayesian treed linear model with the setting `basemax = 10`.

```
> fit4$gpcs

  grow prune change swap
1    0    NA     NA   NA
```

Still, the resulting fit (obtained with much patience) is a much better that the Bayesian CART and treed LM (with `basemax = 10`) ones, as indicated by the RMSE.

We would expect to get the best of both worlds if the setting `basemax = 10` were used when fitting the treed GP model, thus allowing partitioning on the indicators by guarding against rank deficient design matrices.

```
> fit5 <- btgpllm(X = X, Z = Z, XX = XX, m0r1 = TRUE,
+     basemax = 10, verb = 0)
> rmse5 <- sqrt(mean((fit5$ZZ.mean - ZZ)^2))
> rmse5

[1] 0.4805412
```

And indeed this is the case.

The benefits go beyond producing full rank design matrices at the leaves of the tree. Loosely speaking, removing the boolean indicators from the GP part of the treed GP gives a more parsimonious model, without sacrificing any flexibility. The tree is able to capture all of the dependence in the response as a function of the indicator input, and the GP is the appropriate non–linear model for accounting for the remaining relationship between the real–valued inputs and outputs. We can look at the maximum *a' posteriori* (MAP) tree,

```
> h <- fit1$post$height[which.max(fit1$posts$lpost)]
> tgp.trees(fit5, "map")
```

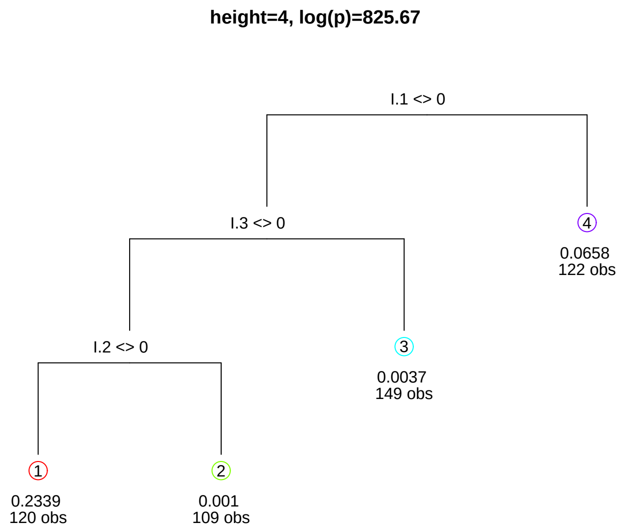

Figure 4: Diagrammatic depiction of the maximum *a' posteriori* (MAP) tree for the boolean indicator version of the Friedman data in Eq. (1) using `basemax=10`.

to see that only (and all of) the indicators were partitioned upon in Figure 4. Further advantages to this approach include speed (a partitioned model gives smaller covariance matrices to invert) and improved mixing in the Markov chain when a separable covariance function is used. Note that using a non–separable covariance function in the presence of indicators would result in a poor fit. Good range (d) settings for the indicators would not necessarily coincide with a a good range setting for the real–valued inputs.

A complimentary setting, `splitmin`, allows the user to specify the first column of the inputs `X` on which treed partitioning is allowed. From Section 3.5 of the first `tgp` vignette [9], it was concluded that the original formulation of Friedman data was stationary, and thus treed partitioning is not required to obtain a good fit. The same would be true of the response in (1) after conditioning on the indicators. Therefore, the most parsimonious model would use `splitmin = 11`, in addition to `basemax = 10`, so that only $\mathbf{X}$ are under the GP, and only I under the tree. Fewer viable candidate inputs for treed partitioning should yield improved mixing in the Markov chain.

```
> fit6 <- btgpllm(X = X, Z = Z, XX = XX, m0r1 = TRUE,
+     basemax = 10, splitmin = 11, verb = 0)
```

```
> rmse6 <- sqrt(mean((fit6$ZZ.mean - ZZ)^2))
> rmse6

[1] 0.3844569
```

Needless to say, it is important that the input `X` have columns which are ordered appropriately before the `basemax` and `splitmin` arguments can be properly applied. Future versions of `tgp` will have a formula–based interface to handle categorical (`factors`) and other inputs more like other R regression routines, e.g., `lm` and `glm`.

2 Analysis of sensitivity to inputs

3 Multiresolution `tgp`

4 Optimization of black box functions

5 Importance tempering

Importance tempering (IT) is a combination of the classic method of importance sampling (IS) [14] and *simulated tempering* (ST) [8]. The ST algorithm is methodology for efficiently sampling from a multimodal density $\pi(\theta)$ where standard methods, such as Metropolis–Hastings (MH) [15, 11] and Gibbs Sampling (GS) [6], fail. It is well–known that MCMC inference in Bayesian treed methods suffer from poor mixing. For example, Chipman et al. [2, 3] recommend periodically restarting the MCMC to avoid chains becoming stuck in local modes of the posterior distribution (particularly in tree space). The `R` argument to the `b*` functions exists precisely to facilitate such restarts within the `tgp` package.

ST is an application of the MH algorithm on the product space of parameters and inverse temperatures $k \in [0, 1]$. That is, ST uses MH to sample from the joint chain $\pi(\theta, k) \propto \pi(\theta)^k p(k)$. The inverse temperature is allowed to take on a discrete set of values $k \in \{k_1, \dots, k_m : k_1 = 1,\ k_i > k_{i+1} \geq 0\}$, called the *temperature ladder*. Typically, ST calls for sampling $(\theta, k)^{(t+1)}$ by first updating $\theta^{(t+1)}$ conditional on $k^{(t)}$ and (possibly) on $\theta^{(t)}$, using MH or GS. Then, for a proposed $k' \sim q(k^{(t)} \to k')$, usually giving equal probability to the nearest inverse temperatures greater and less than $k^{(t)}$, an acceptance ratio is calculated:

$$A^{(t+1)} = \frac{\pi(\theta^{(t+1)})^{k'} p(k') q(k' \to k^{(t)})}{\pi(\theta^{(t+1)})^{k^{(t)}} p(k^{(t)}) q(k^{(t)} \to k')}.$$

Finally, $k^{(t+1)}$ is determined according to the MH accept/reject rule: set $k^{(t+1)} = k'$ with probability $\alpha^{(t+1)} = \min\{1, A^{(t+1)}\}$, or $k^{(t+1)} = k^{(t)}$ otherwise. Standard theory for MH and GS gives that samples from the marginals π_{k_i} can be obtained by collecting samples $\theta^{(t)}$ where $k^{(t)} = k_i$. Samples from $\pi(\theta)$ are obtained when $k^{(t)} = 1$.

The success of ST depends crucially on the ability of the Markov chain frequently to: (a) visit high temperatures (low k) where the probability of escaping local modes is increased; (b) visit $k = 1$ to obtain samples from π. The algorithm can be tuned by: (i.) adjusting the number and location of the rungs of the temperature ladder; or (ii.) setting the pseudo-prior $p(k)$ for inverse temperature.

Geyer & Thompson [8] give ways of adjusting the spacing of the rungs of the ladder so that the ST algorithm achieves between–temperature acceptance rates of 20–40%. More recently, authors have preferred to rely on defaults, e.g.,

$$k_i = \begin{cases} (1+\Delta_k)^{1-i} & \text{geometric spacing} \\ \{1+\Delta_k(i-1)\}^{-1} & \text{harmonic spacing} \end{cases} \qquad i = 1, \dots, m. \tag{2}$$

Motivation for such default spacings is outlined by Liu [14]. Geometric spacing, or uniform spacing of $\log(k_i)$, is also advocated by Neal [16, 17] to encourage the Markov chain to rapidly traverse the breadth of the temperature ladder. Harmonic spacing is more often used by a related method called Metropolis coupled Markov chain Monte Carlo (MC^3) [7]. Both defaults are implemented in the `tgp` package, through the provided `default.itemps` function. A new "sigmoidal" option is also implemented, as discussed below. The rate parameter $\Delta_k > 0$ can be problem specific. Rather than work with Δ_k the `default.itemps` function allows the ladder to be specified via m and the hottest temperature k_m, thus fixing Δ_k implicitly. I.e., for the geometric ladder $\Delta_k = (k_m)^{1/(1-m)} - 1$, and for the harmonic ladder $\Delta_k = \frac{(k_m)^{-1}-1}{m-1}$.

A sigmoidal ladder can provide a higher concentration of temperatures near $k = 1$ without sacrificing the other nice properties of the geometric and harmonic ladders. It is specified by first situating m indices $j_i \in \mathbb{R}$ so that $k_1 = k(j_1) = 1$ and $k_m = k(j_m) = k_\mathrm{m}$ under

$$k(j_i) = 1.01 - \frac{1}{1+e^{j_i}}.$$

The remaining $j_i, i = 2, \dots, (m-1)$ are spaced evenly between j_1 and j_m to fill out the ladder $k_i = k(j_i), i = 1, \dots, (m-1)$.

By way of comparison, consider generating the three different types of ladder with identical minimum inverse temperature $k_\mathrm{m} = 0.1$, the default setting in `tgp`.

```
> geo <- default.itemps(type = "geometric")
> har <- default.itemps(type = "harmonic")
> sig <- default.itemps(type = "sigmoidal")
```

The plots in Figure 5 show the resulting inverse temperature ladders, and their logarithms. Observe how, relative to the geometric ladder, the harmonic ladder has a higher concentration of inverse temperatures near zero, whereas the sigmoidal ladder has a higher concentration near one.

```
> par(mfrow = c(2, 1))
> all <- cbind(geo$k, har$k, sig$k)
> matplot(all, pch = 21:23, main = "inv-temp ladders",
+     xlab = "indx", ylab = "itemp")
> legend("topright", pch = 21:23, c("geometric", "harmonic",
+     "sigmoidal"), col = 1:3)
> matplot(log(all), pch = 21:23, main = "log(inv-temp) ladders",
+     xlab = "indx", ylab = "itemp")
```

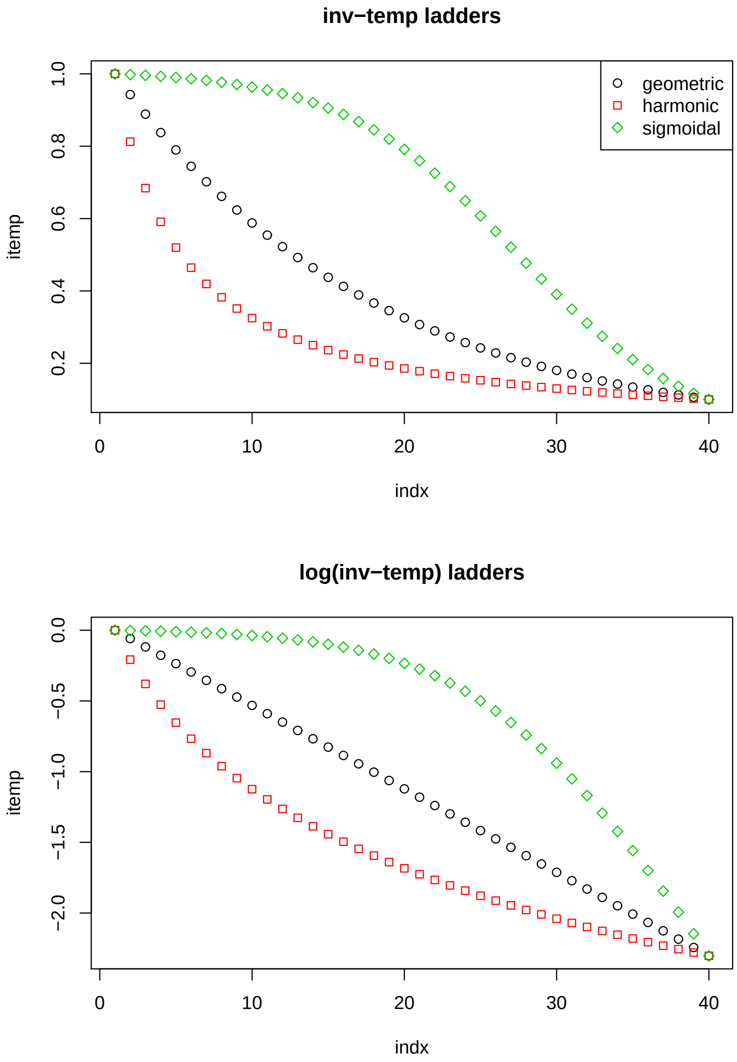

Figure 5: Three different inverse temperature ladders, each with $m = 40$ temperatures starting at $k_1 = 1$ and ending at $k_m = 0.1$

Once a suitable ladder has been chosen, the `tgp` package implementation of ST follows the suggestions of Geyer & Thompson [8] in setting the pseudo–prior, starting from a uniform p_0. First, p_0 is adjusted by *stochastic approximation*: add $c_0/[m(t+n_0)]$ to $\log p_0(k)$ for each $k_i \neq k^{(t)}$ and subtract $c_0/(t+n_0)$ from $\log p_0(k^{(t)})$ over $t = 1, \dots, B$ *burn–in* MCMC rounds sampling from the joint posterior of (θ, k). Then, p_0 is normalized to obtain p_1. Before subsequent runs, specified via an `R >= 2` argument, *occupation numbers* $o(k_i) = \sum_{t=1}^{B} 1_{\{k^{(t)}=k_i\}}$, are used update $p(k_i) \propto p_1(k_i)/o(k_i)$. Note that, in this setting, the `R` argument is used to update the pseudo–prior only, not to restart the Markov chain. Examples are deferred to Section 5.2.

5.1 A new optimal way to combine IS estimators

ST provides us with $\{(\theta^{(t)}, k^{(t)}) : t = 1, \dots, T\}$, where $\theta^{(t)}$ is an observation from $\pi_{k^{(t)}}$. It is convenient to write $\mathcal{T}_i = \{t : k^{(t)} = k_i\}$ for the index set of observations at the i^{th} temperature, and let $T_i = |\mathcal{T}_i|$. Let the vector of observations at the i^{th} temperature collect in $\boldsymbol{\theta}_i = (\theta_{i1}, \dots, \theta_{iT_i})$, so that $\{\theta_{ij}\}_{j=1}^{T_i} \sim \pi_{k_i}$. Each vector $\boldsymbol{\theta}_i$ can be used to construct an IS estimator of $E_\pi\{h(\theta)\}$ by setting

$$\hat{h}_i = \frac{\sum_{j=1}^{T_i} w_i(\theta_{ij}) h(\theta_{ij})}{\sum_{j=1}^{T_i} w_i(\theta_{ij})} \equiv \frac{\sum_{j=1}^{T_i} w_{ij} h(\theta_{ij})}{W_i},$$

say. The efficiency of this estimator can be measured through it's variance, but unfortunately this can be difficult to calculate in general. As a result, the notion of *effective sample size* [14] plays an important role in the study of IS estimators. Denote the the vector of IS weights at the i^{th} temperature is $\mathbf{w}_i = \mathbf{w}_i(\boldsymbol{\theta}_i) = (w_i(\theta_{i1}), \dots, w_i(\theta_{iT_i}))$, where $w_i(\theta) = \pi(\theta)/\pi_{k_i}(\theta)$. The ESS of $\hat{h}_i$ is defined by

$$\mathrm{ESS}(\mathbf{w}_i) = \frac{T}{1 + \mathrm{cv}^2(\mathbf{w}_i)}, \tag{3}$$

where $\mathrm{cv}(\mathbf{w}_i)$ is the *coefficient of variation* of the weights (in the i^{th} temperature), given by

$$\mathrm{cv}^2(\mathbf{w}_i) = \frac{\sum_{t=1}^{T} (w(\theta^{(t)}) - \bar{w})^2}{(T-1)\bar{w}^2}, \qquad \text{where} \qquad \bar{w} = T^{-1} \sum_{t=1}^{T} w(\theta^{(t)}).$$

In R:

```
> ESS <- function(w) {
+     mw <- mean(w)
+     cv2 <- sum((w - mw)^2)/((length(w) - 1) * mw^2)
+     ess <- length(w)/(1 + cv2)
+     return(ess)
+ }
```

This should not be confused with the concept of *effective sample size due to autocorrelation* [13] (due to serially correlated samples coming from a Markov chain as in MCMC).

Jennison [12] was the first to suggest using a *single* tempered distribution as a proposal in IS, though the question of how to choose the best temperature was neither posed or resolved. It is clear that larger k leads to larger ESS, but at the expense of poorer mixing in the Markov chain. It can be shown that the optimal inverse temperature k^* for IS, in the sense of constructing a minimum variance estimator, may be significantly lower than one [10]. This means that the discarded samples obtained when $k^{(t)} < 1$ may actually lead to more efficient estimators than the ones saved from the cold distribution. However, the variance of such an estimator will indeed become unbounded as $k \to 0$, just as ESS $\to 0$. Generally speaking, calculating the variance of IS estimators is difficult, anyways, so it will be hard to pin down the optimal temperature for arbitrary π.

Rather than focus on choosing a single temperature, the method of *importance tempering* (IT) combines the "trans-temporal" method of ST with IS. An optimal heuristic based on ESS, which is easy to calculate, is used to combine the estimators realized at each temperature. The idea is that small ESS will indicate high variance IS estimators which should be relegated to having only a small influence on the overall estimator. Low k_i may be helpful in the context of ST, i.e., for improving mixing in the the Markov chain and communication between modes of the posterior, but they can be harmful if given too much influence when used to calculate expectations.

It is natural to consider an overall estimator of $E_\pi\{h(\theta)\}$ defined by a convex combination:

$$\hat{h}_\lambda = \sum_{i=1}^m \lambda_i \hat{h}_i, \qquad \text{where} \qquad 0 \leq \lambda_i \leq \sum_{i=1}^m \lambda_i = 1. \tag{4}$$

Unfortunately, if $\lambda_1, \dots, \lambda_m$ are not chosen carefully, $\mathrm{Var}(\hat{h}_\lambda)$, can be nearly as large as the largest $\mathrm{Var}(\hat{h}_i)$ [18]. Notice that ST is recovered as a special case when $\lambda_1 = 1$ and $\lambda_2, \dots, \lambda_m = 0$. It may be tempting to choose $\lambda_i = W_i/W$, where $W = \sum_{i=1}^m W_i$. The resulting estimator is equivalent to

$$\hat{h} = W^{-1} \sum_{t=1}^T w(\theta^{(t)}, k^{(t)}) h(\theta^{(t)}), \qquad \text{where} \qquad W = \sum_{t=1}^T w(\theta^{(t)}, k^{(t)}), \tag{5}$$

and $w(\theta, k) = \pi(\theta)/\pi(\theta)^k = \pi(\theta)^{1-k}$. It can lead to a very poor estimator, even compared to ST, as will be demonstrated empirically in the examples to follow shortly.

Observe that we can equivalently write

$$\hat{h}_\lambda = \sum_{i=1}^m \sum_{j=1}^{T_i} w_{ij}^\lambda h(\theta_{ij}), \qquad \text{where} \qquad w_{ij}^\lambda = \lambda_i w_{ij}/W_i. \tag{6}$$

Let $\mathbf{w}^\lambda = (w^\lambda_{11}, \dots, w^\lambda_{1T_1}, w^\lambda_{21}, \dots, w^\lambda_{2T_2}, \dots, w^\lambda_{m1}, \dots, w^\lambda_{mT_m})$. Attempting to choose $\lambda_1, \dots, \lambda_m$ to minimize $\mathrm{Var}(\hat{h}_\lambda)$ directly can be difficult. Moreover, for the applications that we have in mind, it is important that our estimator can be constructed without knowledge of the normalizing constants of $\pi_{k_1}, \dots, \pi_{k_m}$, and without evaluating the MH transition kernels $\mathcal{K}_{\pi_{k_i}}(\cdot, \cdot)$. It is for this reason that methods like the *balance heuristic* [19], MCV [18], or population Monte Carlo (PMC) [4] cannot be applied. Instead, we seek maximize the effective sample size of $\hat{h}_\lambda$ in (4), and look for an $O(T)$ operation to determine the optimal λ^*.

Among estimators of the form (4), it can be shown [10] that $\mathrm{ESS}(\mathbf{w}^\lambda)$ is maximized by $\lambda = \lambda^*$, where, for $i = 1, \dots, m$,

$$\lambda^*_i = \frac{\ell_i}{\sum_{i=1}^m \ell_i}, \qquad \text{and} \qquad \ell_i = \frac{W_i^2}{\sum_{j=1}^{T_i} w_{ij}^2}.$$

The efficiency of each IS estimator $\hat{h}_i$ can be measured through $\mathrm{ESS}(\mathbf{w}_i)$. Intuitively, we hope that with a good choice of λ, the ESS (3) of $\hat{h}_\lambda$, would be close to the sum over i of the effective sample sizes each of $\hat{h}_i$. This is indeed the case for $\hat{h}_{\lambda^*}$, because it can be shown [10] that

$$\mathrm{ESS}(\mathbf{w}^{\lambda^*}) \geq \sum_{i=1}^m \mathrm{ESS}(\mathbf{w}_i) - \frac{1}{4} - \frac{1}{T}.$$

In practice we have found that this bound is conservative and that in fact $\mathrm{ESS}(\mathbf{w}^{\lambda^*}) \geq \sum_{i=1}^m \mathrm{ESS}(\mathbf{w}_i)$, as will be shown empirically in the examples that follow. Thus our optimally–combined IS estimator has a highly desirable and intuitive property in terms of its effective sample size: that the whole is greater than the sum of its parts.

$\mathrm{ESS}(\mathbf{w}^{\lambda^*})$ depends on $\mathrm{ESS}(\mathbf{w}_i)$ which in turn depend on the k_i. Smaller k_i will lead to better mixing in the Markov chain, but lower $\mathrm{ESS}(\mathbf{w}_i)$. Therefore, we can expect that the geometric and sigmoidal ladders will fare better than the harmonic ones, so long as the desired improvements in mixing are achieved. In the examples to follow, we shall see that the sigmoidal ladder does indeed leader to higher $\mathrm{ESS}(\mathbf{w}^{\lambda^*})$.

5.2 Examples

Here the IT method shown in action for `tgp` models. IT is controlled in `b*` function via the `itemps` argument: a `data.frame` coinciding with the output of the `default.itemps` function. The `lambda` argument to `default.itemps` can be used to base posterior predictive inference the other IT heuristics: ST and the naïve approach (5). Whenever the argument `m = 1` is used with $k.min! = 1$ the resulting estimator is constructed via tempered importance sampling at inverse temperature `k.min`. The parameters c_0 and n_0 for stochastic approximation of the pseudo–prior can be specified as a 2–vector `c0n0` argument to `default.itemps`. In the examples which follow we simply use the default configuration of the IT method, adjusting only the minimum inverse temperature via the `k.min` argument.

Before delving into more involved examples, we illustrate the stages involved in a small run of importance tempering (IT) on the exponential data from Section 3.3 of Gramacy (2007). The data can be obtained as:

```
> exp2d.data <- exp2d.rand()
> X <- exp2d.data$X
> Z <- exp2d.data$Z
```

Now, consider applying IT to the Bayesian treed LM with a small geometric ladder. A warning will be given if the default setting of `bprior="bflat"` is used, as this (numerically) improper prior can lead to improper posterior inference at high temperatures.

```
> its <- default.itemps(m = 10)
> exp.btlm <- btlm(X = X, Z = Z, bprior = "b0", R = 2,
+     itemps = its, pred.n = FALSE)
```

```
burn in: [with stoch approx (c0,n0)=(100,1000)]
**GROW** @depth 0: [1,0.45], n=(58,22)
**GROW** @depth 1: [1,0.25], n=(37,21)
**PRUNE** @depth 1: [1,0.25]
**GROW** @depth 1: [1,0.2], n=(31,27)
**PRUNE** @depth 1: [1,0.25]
**GROW** @depth 1: [2,0.5], n=(50,12)
**PRUNE** @depth 1: [2,0.45]
**GROW** @depth 1: [2,0.5], n=(50,13)
**PRUNE** @depth 1: [2,0.5]
**GROW** @depth 1: [2,0.45], n=(48,15)
**PRUNE** @depth 1: [2,0.45]
**GROW** @depth 1: [2,0.55], n=(51,11)
r=1000 d=[0] [0] [0]; n=(51,11,18) k=0.1
r=2000 d=[0] [0] [0]; n=(48,14,18) k=0.599484
**GROW** @depth 2: [2,0.3], n=(36,15)
**PRUNE** @depth 2: [2,0.35]
**PRUNE** @depth 1: [2,0.5]
**GROW** @depth 1: [2,0.5], n=(50,13)
**PRUNE** @depth 1: [2,0.5]
**PRUNE** @depth 0: [1,0.5]
r=3000 d=[0]; n=80 k=0.278256
**GROW** @depth 0: [1,0.4], n=(54,26)
**GROW** @depth 1: [1,0.6], n=(12,14)
**GROW** @depth 2: [2,0.5], n=(43,11)
**CPRUNE** @depth 0: var=1, val=0.6->0.55, n=(63,17)
**GROW** @depth 2: [2,0.3], n=(36,15)
**PRUNE** @depth 2: [2,0.3]
r=4000 d=[0] [0] [0]; n=(48,15,17) k=0.774264
**PRUNE** @depth 1: [2,0.45]
```

```
**GROW** @depth 1: [2,0.45], n=(48,14)
**PRUNE** @depth 1: [2,0.45]
**GROW** @depth 1: [2,0.5], n=(50,12)
**PRUNE** @depth 1: [2,0.5]
**GROW** @depth 1: [2,0.5], n=(50,12)
r=5000 d=[0] [0] [0]; n=(51,12,17) k=0.278256
**PRUNE** @depth 1: [2,0.5]
**GROW** @depth 1: [2,0.55], n=(51,11)
**PRUNE** @depth 1: [2,0.45]
**GROW** @depth 1: [2,0.45], n=(48,15)
**PRUNE** @depth 1: [2,0.45]
**GROW** @depth 1: [2,0.45], n=(48,14)
r=6000 d=[0] [0] [0]; n=(48,14,18) k=0.16681
**PRUNE** @depth 1: [2,0.45]
**GROW** @depth 1: [2,0.5], n=(50,13)
**PRUNE** @depth 1: [2,0.5]
**GROW** @depth 1: [2,0.5], n=(50,12)
**PRUNE** @depth 1: [1,0.55]
r=7000 d=[0] [0]; n=(62,18) k=0.129155

Sampling @ nn=0 pred locs:
**GROW** @depth 1: [1,0.45], n=(45,12)
**GROW** @depth 1: [1,0.3], n=(12,11)
**PRUNE** @depth 1: [1,0.3]
**PRUNE** @depth 1: [1,0.45]
**GROW** @depth 1: [2,0.3], n=(41,16)
r=1000 d=[0] [0] [0]; mh=3 n=(45,12,23) k=0.599484
**PRUNE** @depth 1: [2,0.35]
**GROW** @depth 1: [1,0.2], n=(23,34)
**PRUNE** @depth 1: [1,0.2]
**GROW** @depth 1: [2,0.35], n=(45,12)
**PRUNE** @depth 1: [2,0.35]
**GROW** @depth 1: [1,0.2], n=(23,34)
**PRUNE** @depth 1: [1,0.25]
**GROW** @depth 1: [1,0.45], n=(45,12)
r=2000 d=[0] [0] [0]; mh=3 n=(45,12,23) k=0.774264
**PRUNE** @depth 1: [1,0.45]
**GROW** @depth 1: [1,0.3], n=(12,11)
**PRUNE** @depth 1: [1,0.3]
**GROW** @depth 1: [1,0.45], n=(45,12)
r=3000 d=[0] [0] [0]; mh=3 n=(45,12,23) k=0.464159
**PRUNE** @depth 1: [1,0.45]
**GROW** @depth 1: [1,0.5], n=(51,11)
r=4000 d=[0] [0] [0]; mh=3 n=(48,15,17) k=0.464159
**GROW** @depth 2: [2,0.3], n=(36,12)
**PRUNE** @depth 2: [2,0.3]
```

```
r=5000 d=[0] [0] [0]; mh=3 n=(48,14,18) k=0.599484
Grow: 5.498%, Prune: 4.874%, Change: 36.27%, Swap: 27.54%
finished repetition 1 of 2

burn in:
**GROW** @depth 2: [2,0.3], n=(36,12)
**PRUNE** @depth 2: [2,0.3]
**CPRUNE** @depth 0: var=2, val=0.55->0.5, n=(60,20)
**GROW** @depth 1: [2,0.75], n=(11,12)
**PRUNE** @depth 1: [2,0.45]
r=1000 d=[0] [0]; mh=3 n=(57,23) k=0.16681
**GROW** @depth 1: [1,0.45], n=(45,12)
**PRUNE** @depth 1: [1,0.45]
r=2000 d=[0] [0]; mh=3 n=(57,23) k=0.16681

Sampling @ nn=0 pred locs:
**GROW** @depth 1: [1,0.3], n=(12,11)
**PRUNE** @depth 1: [1,0.3]
**GROW** @depth 1: [2,0.3], n=(41,16)
**PRUNE** @depth 1: [2,0.3]
**GROW** @depth 1: [2,0.3], n=(41,16)
**PRUNE** @depth 1: [2,0.35]
**GROW** @depth 1: [1,0.35], n=(38,19)
r=1000 d=[0] [0] [0]; mh=3 n=(45,12,23) k=0.359381
**PRUNE** @depth 1: [1,0.45]
**GROW** @depth 1: [1,0.4], n=(41,16)
r=2000 d=[0] [0] [0]; mh=3 n=(45,12,23) k=1
**GROW** @depth 2: [2,0.3], n=(34,11)
**PRUNE** @depth 2: [2,0.3]
**GROW** @depth 2: [1,0.1], n=(13,32)
**PRUNE** @depth 2: [1,0.1]
r=3000 d=[0] [0] [0]; mh=3 n=(45,12,23) k=0.359381
**PRUNE** @depth 1: [1,0.45]
**GROW** @depth 1: [1,0.25], n=(28,29)
**PRUNE** @depth 1: [1,0.25]
**GROW** @depth 1: [2,0.3], n=(41,16)
**GROW** @depth 1: [2,0.1], n=(18,23)
**PRUNE** @depth 2: [2,0.4]
r=4000 d=[0] [0] [0]; mh=3 n=(12,45,23) k=0.359381
**PRUNE** @depth 1: [2,0.05]
**GROW** @depth 1: [1,0.3], n=(12,11)
**PRUNE** @depth 1: [1,0.3]
**GROW** @depth 1: [2,0.35], n=(45,12)
**PRUNE** @depth 1: [2,0.35]
**GROW** @depth 1: [1,0.4], n=(41,16)
**GROW** @depth 2: [2,0.3], n=(34,11)
```

```
**PRUNE** @depth 2: [2,0.3]
**PRUNE** @depth 1: [1,0.45]
**GROW** @depth 1: [1,0.3], n=(12,11)
**PRUNE** @depth 1: [1,0.3]
**GROW** @depth 1: [2,0.3], n=(41,16)
r=5000 d=[0] [0] [0]; mh=3 n=(41,16,23) k=0.774264
Grow: 5.49%, Prune: 5.098%, Change: 29.5%, Swap: 27.6%
finished repetition 2 of 2

effective sample sizes:
0: itemp=1, len=1084, ess=1084
1: itemp=0.774264, len=934, ess=72.1928
2: itemp=0.599484, len=712, ess=1.62342
3: itemp=0.464159, len=796, ess=4.5198
4: itemp=0.359381, len=402, ess=4.43881
5: itemp=0.278256, len=310, ess=1.59663
6: itemp=0.215443, len=282, ess=0.996501
7: itemp=0.16681, len=206, ess=2.04431
8: itemp=0.129155, len=134, ess=1.09449
9: itemp=0.1, len=140, ess=0.99675
total: len=5000, ess.sum=1173.5, ess(w)=1173.45
lambda-combined ess=1173.45
```

Notice how the MCMC inference procedure starts with $B + T = 9000$ rounds of stochastic approximation (initial adjustment of the pseudo–prior) in place of typical (default) the $B = 2000$ burn–in rounds. Then, the first round of sampling from the posterior commences, over $T = 2$ rounds, during which the observation counts in each temperature are tallied. The progress meter shows the current temperature the chain is in, say `k=0.629961`, after each of 1000 sampling rounds. The first repeat starts with a pseudo–prior that has been adjusted by the observation counts, which continue to be accumulated throughout the entire procedure (i.e., they are never reset). Any subsequent repeats begin after a similar (re-)adjustment.

Before finishing, the routine summarizes the sample size and effective sample sizes in each rung of the temperature ladder. The number of samples is given by `len`, the ESS by `ess`. These quantities can also be recovered via `traces`, as shown in the examples to follow. The ESS of the optimal combined IT sample is the last quantity printed. This, along with the ESS and total numbers of samples in each temperature, can also be obtained via the `tgp`-class output object.

```
> exp.btlm$ess

$combined
[1] 1173.448

$each
```

```
           k count          ess
1  1.0000000  1084 1084.0000000
2  0.7742637   934   72.1927837
3  0.5994843   712    1.6234211
4  0.4641589   796    4.5197963
5  0.3593814   402    4.4388131
6  0.2782559   310    1.5966258
7  0.2154435   282    0.9965006
8  0.1668101   206    2.0443095
9  0.1291550   134    1.0944920
10 0.1000000   140    0.9967501
```

5.2.1 Motorcycle accident data

Recall the motorcycle accident data of Section 3.4 of the first tgp vignette [9]. Consider using IT to sample from the posterior distribution of the treed GP LLM model using the geometric temperature ladder.

```
> library(MASS)
> moto.it <- btgpllm(X = mcycle[, 1], Z = mcycle[,
+     2], BTE = c(2000, 52000, 10), m0r1 = TRUE, bprior = "b0",
+     R = 3, itemps = geo, trace = TRUE, pred.n = FALSE,
+     verb = 0)
```

Out of a total of 15600 from the joint chain, the resulting (optimally combined) ESS was:

```
> moto.it$ess$combined
```

```
[1] 1243.051
```

Alternatively, $\mathbf{w}^{\lambda^*}$ can be extracted from the traces, and used to make the ESS calculation directly.

```
> p <- moto.it$trace$post
> ESS(p$wlambda)
```

```
[1] 1243.052
```

The unadjusted weights $\mathbf{w}$ are also available from in the trace. We can see that the naïve choice of $\lambda_i = W_i/W$, leading to the estimator in (5), has a clearly inferior effective sample size.

```
> ESS(p$w)
```

```
[1] 8.720477
```

To see the benefit of IT over ST we can simply count the number of samples obtained when $k^{(t)} = 1$. This can be accomplished in several ways: either via the traces or through the output object.

```
> as.vector(c(sum(p$itemp == 2), moto.it$ess$each[1,
+     2:3]))

[[1]]
[1] 0

$count
[1] 466

$ess
[1] 466
```

That is, (optimal) IT gives effectively 2.67 times more samples. The naïve combination, leading to the estimator in (5), yields an estimator with effective sample size that is 2% of the number of samples obtained under ST.

Now, we should like to compare to the MCMC samples obtained under the same model, without IT.

```
> moto.reg <- btgpllm(X = mcycle[, 1], Z = mcycle[,
+     2], BTE = c(2000, 52000, 10), R = 3, m0r1 = TRUE,
+     bprior = "b0", trace = TRUE, pred.n = FALSE,
+     verb = 0)
```

The easiest comparison to make is to look at the heights explored under the three chains: the regular one, the chain of heights visited at all temperatures (combined), and those obtained after applying IT via re-weighting under the optimal combination λ^*.

```
> L <- length(p$height)
> hw <- suppressWarnings(sample(p$height, L, prob = p$wlambda,
+     replace = TRUE))
> b <- hist2bar(cbind(moto.reg$trace$post$height, p$height,
+     hw))
```

Figure 6 shows barplots indicating the count of the number of times the Markov chains were in trees of various heights after burn–in. Notice how the tempered chain (denoted “All Temps” in the figure) frequently visits trees of height one, whereas the non–tempered chain (denoted “reg MCMC”) never does. The result is that the non–tempered chain underestimates the probability of height two trees and produces a corresponding overestimate of height four trees—which are clearly not supported by the data—even visiting trees of height five. The IT estimator appropriately down–weights height one trees and provides correspondingly more realistic estimates of height two and four trees.

The improved mixing of the ST/IT chain is also evident in the increased rate of accepted tree operations. These rates can be accessed from the `tgp`–class objects as follows.

```
> moto.it$gpcs
> barplot(b, beside = TRUE, col = 1:3, xlab = "tree height",
+     ylab = "counts", main = "tree heights encountered")
> legend("topright", c("reg MCMC", "All Temps", "IT"),
+     fill = 1:3)
```

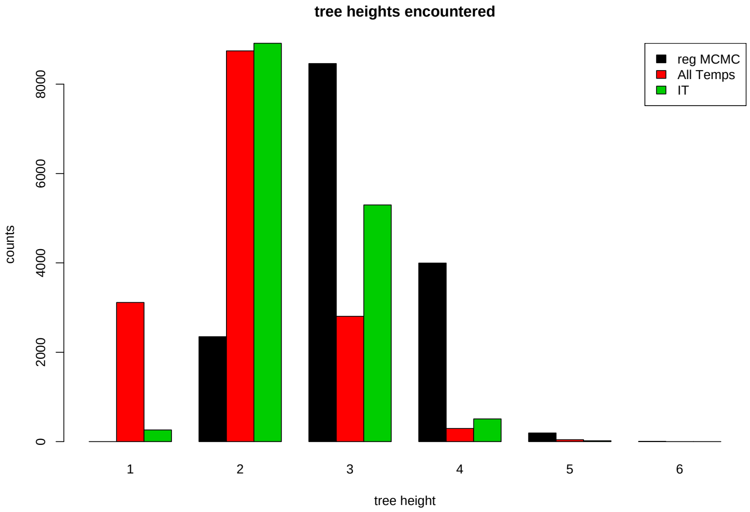

Figure 6: Barplots indicating the counts of the number of times the Markov chains (for regular MCMC, combining all temperatures in the inverse temperature ladder, and those re-weighted via IT) were in trees of various heights for the motorcycle data.

```
        grow       prune    change      swap
1 0.03702610 0.04660486 0.5242553 0.9740578
```

```
> moto.reg$gpcs
```

```
       grow     prune    change      swap
1 0.0345094 0.0330873 0.4072958 0.8864704
```

The increased rate of *prune* operations explains how the tempered distributions helped the chain escape the local modes of deep trees.

Whenever introducing another parameter into the model, like the inverse temperature k, it is important to check that the marginal posterior chain for that parameter is mixing well. For ST it is crucial that the chain makes rapid excursions between the cold temperature, the hottest temperatures, and to visit each temperature setting roughly the same number of times.

Figure 7 shows a trace of the posterior samples for k in the motorcycle experiment. Arguably, the mixing in k–space leaves something to be desired. Since it can be very difficult to tune the pseudo–prior and MH proposal mechanism to get good mixing in k–space, it is fortunate that the IT methodology does not rely on the same mixing properties as ST does. Since samples can be obtained from the posterior distribution of the parameters of interest by re-weighting samples obtained when $k < 1$ it is only important that the chain frequently visit low

```
> plot(log(moto.it$trace$post$itemp), type = "l", ylab = "log(k)",
+     xlab = "samples", main = "trace of log(k)")
```

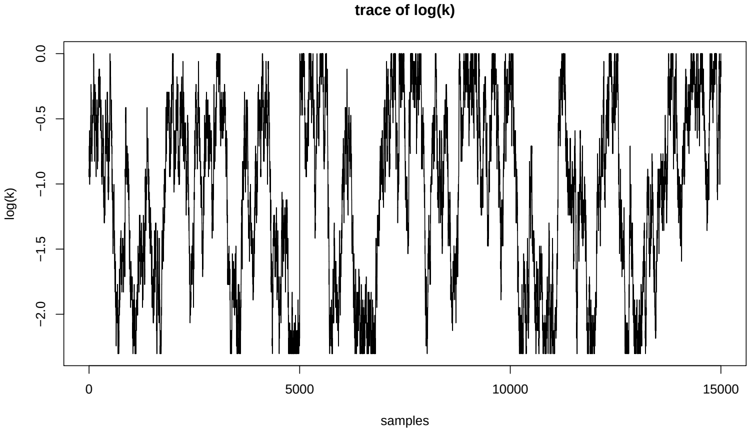

Figure 7: A trace of the MCMC samples from the marginal posterior distribution of the inverse temperature parameter, k, in the motorcycle experiment

temperatures to obtain good sampling, and high temperatures to obtain good mixing. The actual time spent in specific temperatures, i.e., $k = 1$ is less important. Figure 8 shows the histogram of the inverse temperatures visited in the Markov chain for the motorcycle experiment. Also plotted is a histogram of the *observation counts* in each temperature. The two histograms should similar shape but different totals. Observation counts are tallied during every MCMC sample after burn–in, whereas the posterior samples of k are thinned (at a rate specified in `BTE[3]`). When the default `trace = FALSE` argument is used only the observation counts will be available in the `tgp`–class object, and these can be used as a surrogate for a trace of k.

The compromise IT approach obtained using the sigmoidal ladder can yield an increase in ESS.

```
> moto.it.sig <- btgpllm(X = mcycle[, 1], Z = mcycle[,
+     2], BTE = c(2000, 52000, 10), R = 3, m0r1 = TRUE,
+     bprior = "b0", krige = FALSE, itemps = sig, verb = 0)
```

Compare the resulting ESS to the one given for the geometric ladder above.

```
> moto.it.sig$ess$combined

[1] 5333.829
```

Plots of the resulting predictive surface is shown in Figure 9 for comparison with those in Section 3.4 of the first `tgp` vignette [9].

```
> b <- itemps.barplot(moto.it, plot.it = FALSE)
> barplot(t(cbind(moto.it$itemps$counts, b)), col = 1:2,
+     beside = TRUE, ylab = "counts", xlab = "itemps",
+     main = "inv-temp observation counts")
> legend("topright", c("observation counts", "posterior samples"),
+     fill = 1:2)
```

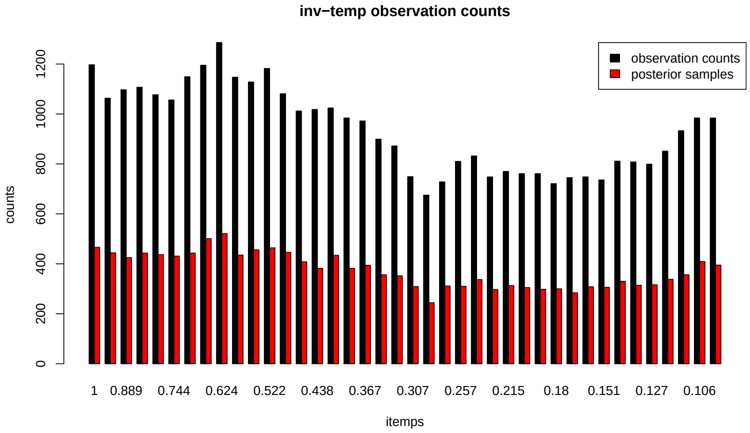

Figure 8: Comparing (thinned) samples from the posterior distribution for the inverse temperature parameter, k, (posterior samples), to the observation counts used to update the pseudo–prior, in the motorcycle experiment

```
> plot(moto.it.sig)
```

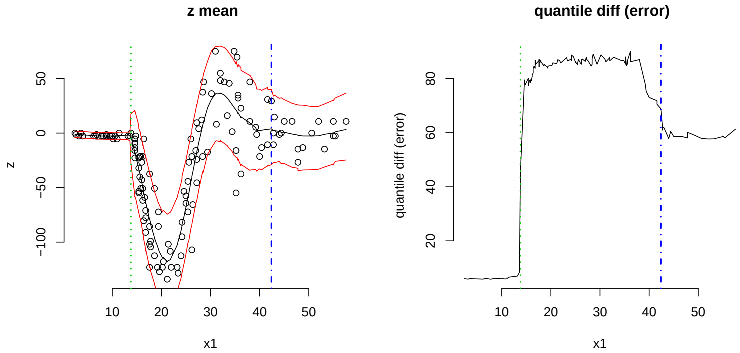

Figure 9: Posterior predictive surface for the motorcycle data, with 90% quantile errorbars, obtained under IT with the sigmoidal ladder.

5.2.2 Synthetic 2–d Exponential Data

Recall the synthetic 2–d exponential data of Section 3.4 of the tgp vignette [9], where the true response is given by

$$z(\mathbf{x}) = x_1 \exp(x_1^2 - x_2^2).$$

Here, we will take $\mathbf{x} \in [-6, 6] \times [-6, 6]$ with a D–optimal design

```
> Xcand <- lhs(10000, rbind(c(-6, 6), c(-6, 6)))
> X <- dopt.gp(400, X = NULL, Xcand)$XX
> Z <- exp2d.Z(X)$Z
```

Consider a treed GP LLM model fit to this data using the standard RJ–MCMC.

```
> exp.reg <- btgpllm(X = X, Z = Z, BTE = c(2000, 52000,
+     10), bprior = "b0", trace = TRUE, krige = FALSE,
+     R = 10, verb = 0)
```

```
> plot(exp.reg)
```

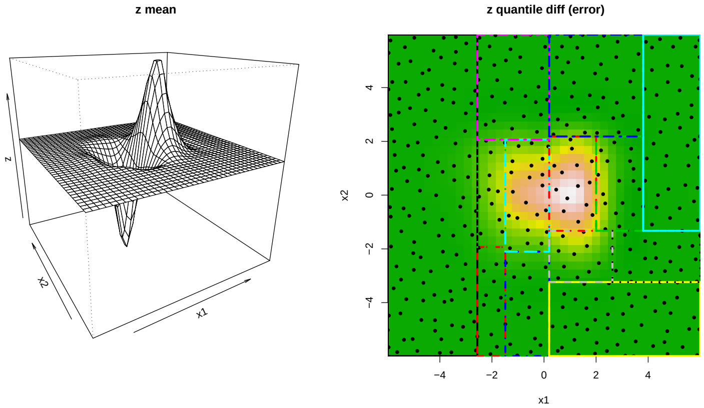

Figure 10: Posterior predictive surface for the 2–d exponential data: mean surface *(left)* and 90% quantile difference *(right)*

Figure 10 shows the resulting posterior predictive surface. The maximum *a' posteriori* (MAP) tree is drawn over the error surface in the *right–hand* plot. The height of this tree can be obtained from the `tgp`-class object.

```
> h <- exp.reg$post$height[which.max(exp.reg$posts$lpost)]
> h
```

```
[1] 6
```

It is easy to see that many fewer partitions are actually necessary to separate the interesting, central, region from the surrounding flat region. Figure 11 shows a diagrammatic representation of the MAP tree. Given the apparent over–partitioning in this height 6 tree it would be surprising to find much posterior support for trees of greater height. One might indeed suspect that there are trees with fewer partitions which would have higher posterior probability, and thus guess that the Markov chain for the trees plotted in these figures possibly

```
> tgp.trees(exp.reg, "map")
```

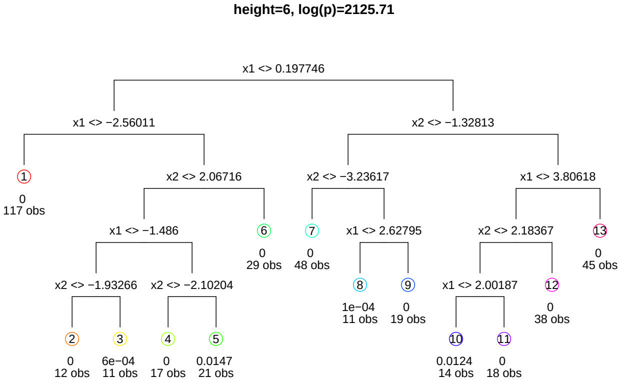

Figure 11: Diagrammatic depiction of the maximum *a' posteriori* (MAP) tree for the 2–d exponential data under standard RJ–MCMC sampling

became stuck in a local mode of tree space while on an excursion into deeper trees.

Now consider using IT. It will be important in this case to have a k_{m} small enough to ensure that the tree occasionally prunes back to the root. We shall therefore use a smaller k_{m}.Generally speaking, some pilot tuning may be necessary to choose an appropriate k_{m} and number of rungs m, though the defaults should give adequate performance in most cases.

```
> its <- default.itemps(k.min = 0.02)
> exp.it <- btgpllm(X = X, Z = Z, BTE = c(2000, 52000,
+     10), bprior = "b0", trace = TRUE, krige = FALSE,
+     itemps = its, R = 10, verb = 0)
```

As expected, the tempered chain moves more rapidly throughout tree space by accepting more tree proposals.

```
> exp.it$gpcs

        grow      prune    change      swap
1 0.07903683 0.07878659 0.8149473 0.6708709
```

```
> exp.reg$gpcs

        grow       prune    change      swap
1 0.01150490 0.008335254 0.6975074 0.3366485
```

We can quickly compare the effective sample sizes of the three possible estimators: ST, naïve IT, and optimal IT.

```
> p <- exp.it$trace$post
> data.frame(ST = sum(p$itemp == 1), nIT = ESS(p$w),
+     oIT = exp.it$ess$combined)
```

```
   ST      nIT      oIT
1 239 44.96045 384.8271
```

Due to the thinning in the Markov chain (`BTE[3] = 10`) and the traversal between $m = 10$ temperatures in the ladder, we can be reasonably certain that the 385 samples obtained via IT from the total of 50000 samples obtained from the posterior are far less correlated than the ones obtained via standard RJ–MCMC.

As with the motorcycle data, we can compare the tree heights visited by the two chains.

```
> L <- length(p$height)
> hw <- suppressWarnings(sample(p$height, L, prob = p$wlambda,
+     replace = TRUE))
> b <- hist2bar(cbind(exp.reg$trace$post$height, p$height,
+     hw))

> barplot(b, beside = TRUE, col = 1:3, xlab = "tree height",
+     ylab = "counts", main = "tree heights encountered")
> legend("topright", c("reg MCMC", "All Temps", "IT"),
+     fill = 1:3)
```

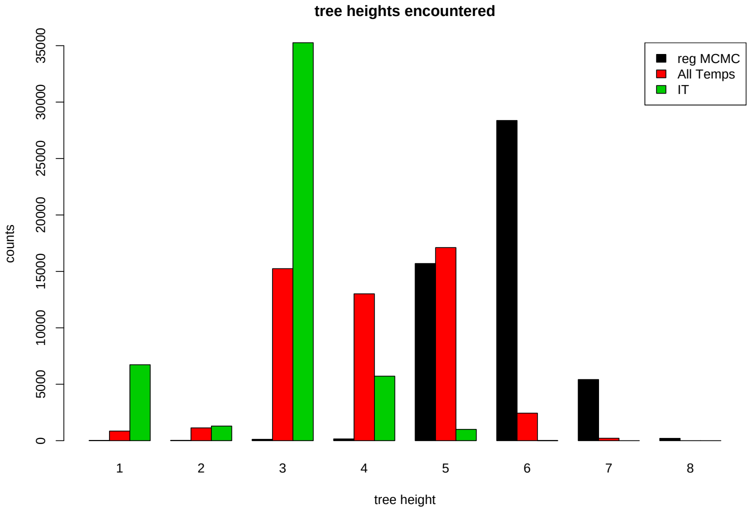

Figure 12: Barplots indicating the counts of the number of times the Markov chains (for regular MCMC, combining all temperatures in the inverse temperature ladder, and those re-weighted via IT) were in trees of various heights for the 2–d exponential data.

Figure 12 shows a barplot of `b`, which illustrates that the tempered chain frequently visited trees shallow trees. IT with the optimal weights shows that the standard RJ–MCMC chain missed many trees of height three and four with consider posterior support.

To more directly compare the mixing in tree space between the ST and tempered chains, consider the trace plots of the heights of the trees explored by

```
> ylim <- range(p$height, exp.reg$trace$post$height)
> plot(p$height, type = "l", main = "trace of tree heights",
+     xlab = "t", ylab = "height", ylim = ylim)
> lines(exp.reg$trace$post$height, col = 2)
> legend("topright", c("tempered", "reg MCMC"), lty = c(1,
+     1), col = 1:2)
```

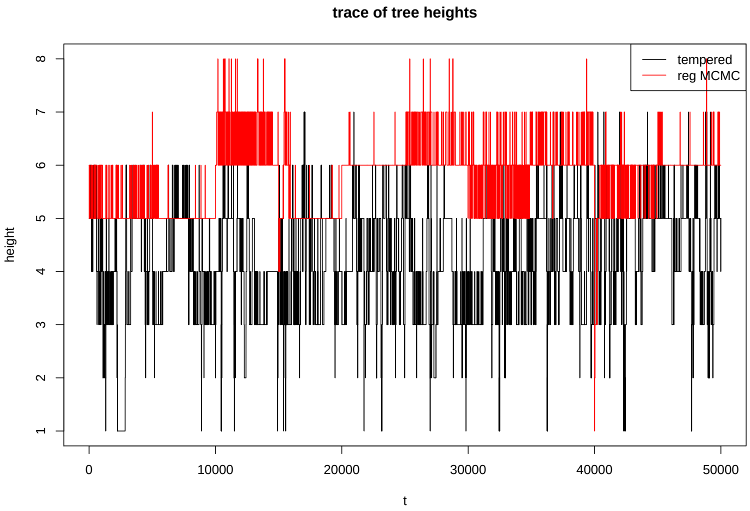

Figure 13: Traces of the tree heights obtained under the two Markov chains Markov chains (for regular MCMC, combining all temperatures in the inverse temperature ladder) on the 2–d exponential data.

the chains shown in Figure 13. Despite being restarted 10 times, the regular RJ–MCMC chain (almost) never visits trees of height less than five after burn–in and instead makes rather lengthy excursions into deeper trees, exploring a local mode in the posterior. In contrast, the tempered chain frequently prunes back to the tree root, and consequently discovers posterior modes in tree heights three and four.

To conclude, a plot of the posterior predictive surface is given in Figure 14, where the MAP tree is shown both graphically and diagrammatically.

References

[1] L. Breiman, J. H. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Wadsworth, Belmont, CA, 1984.

[2] H. A. Chipman, E. I. George, and R. E. McCulloch. Bayesian CART model search (with discussion). *Journal of the American Statistical Association*, 93:935–960, 1998.

```
> plot(exp.it)
> tgp.trees(exp.it, "map")
```

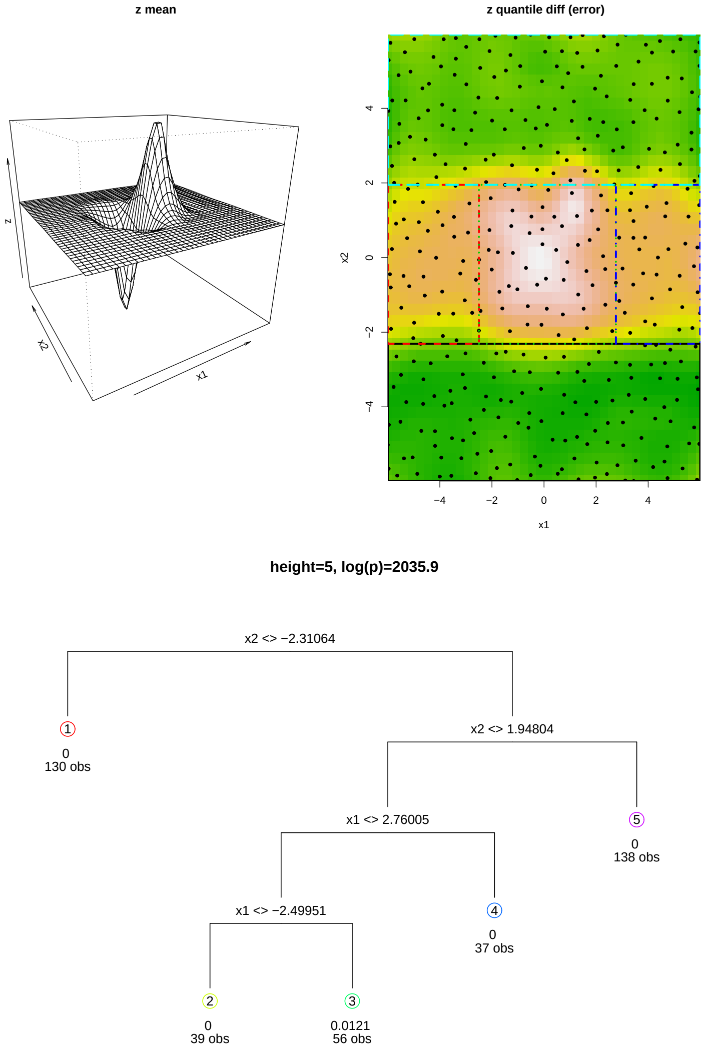

Figure 14: 2–d exponential data fit with IT. *Top:* Posterior predictive mean surface for the 2d–exponential, with the MAP tree overlayed. *Bottom:* diagrammatic representation of the MAP tree.

[3] H. A. Chipman, E. I. George, and R. E. McCulloch. Bayesian treed models. *Machine Learning*, 48:303–324, 2002.

[4] R. Douc, A. Guillin, J.-M. Marin, and C.P. Robert. Minimum variance importance sampling via population monte carlo. Technical report, CERE-

MADE, Université Paris Dauphine, and CREST, INSEE, Paris, 2007.

[5] J. H. Friedman. Multivariate adaptive regression splines. *Annals of Statistics*, 19, No. 1:1–67, March 1991.

[6] S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:721–741, 1984.

[7] C.J. Geyer. Markov chain Monte Carlo maximum likelihood. In *Computing Science and Statistics: Proceedings of the 23rd Symposium on the Interface*, pages 156–163, 1991.

[8] C.J. Geyer and E.A. Thompson. Annealing Markov chain Monte Carlo with applications to ancenstral inference. *Journal of the American Statistical Association*, 90:909–920, 1995.

[9] Robert B. Gramacy. `tgp`: An R package for Bayesian nonstationary, semiparametric nonlinear regression and design by treed gaussian process models. *Journal of Statistical Software*, 19(9), 6 2007.

[10] Robert B. Gramacy, Richard J. Samworth, and Ruth King. Importance tempering. Technical Report 0707.4242, ArXiv, 2007.

[11] W.K. Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57:97–109, 1970.

[12] C. Jennison. Discussion on the meeting on the gibbs sampler and other Markov chain Monte Carlo methods. *Journal of the Royal Statistical Society, Series B*, 55:54–56, 1993.

[13] Robert E. Kass, Bradley P. Carlin, Andrew Gelman, and Radford M. Neal. Markov chain monte carlo in practice: A roundtable discussion. *The American Statistician*, 52(2):93–100, May 1998.

[14] J. S. Liu. *Monte Carlo Strategies in Scientific Computing.* Springer, New York, 2001.

[15] N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, and R. Teller. Equations of state calculations by fast computing machine. *Journal of Chemical Physics*, 21:1087–1091, 1953.

[16] Radford M. Neal. Sampling from multimodal distributions using tempered transition. *Statistics and Computing*, 6:353–366, 1996.

[17] Radford M. Neal. Annealed importance sampling. *Statistics and Computing*, 11:125–129, 2001.

[18] Art Owen and Yi Zhou. Safe and effective importance sampling. *Journal of the American Statstical Association*, 95(449):135–143, March 2000.

[19] Eric Veach and Leonidas J. Guibas. Optimally combining sampling techniques for monte carlo rendering. In *SIGGRAPH '95 Conference Proceedings*, pages 419–428, Reading, MA, 1995. Addison–Wesley.