

Acknowledgments

This work was partially supported by research subaward 08008-002-011-000 from the Universities Space Research Association and NASA, NASA/University Affiliated Research Center grant SC 2003028 NAS2-03144, Sandia National Laboratories grant 496420, and National Science Foundation grants DMS 0233710 and 0504851. I would like to thank Matt Taddy for his contributions to recent releases of the package. I am especially grateful to my thesis advisor, Herbie Lee, whose contributions and guidance in this project have been invaluable throughout. Finally, I would like to thank an anonymous referee whose many helpful comments improved the paper.

A Implementation notes

The treed GP model is coded in a mixture of `C` and `C++`: `C++` for the tree data structure (\mathcal{T}) and `C` for the GP at each leaf of \mathcal{T} . The code has been tested on Unix (`Solaris`, `Linux`, `FreeBSD`, `OSX`) and Windows (2000, XP) platforms.

It is useful to first translate and re-scale the input data (\mathbf{X}) so that it lies in an $\mathbb{R}^{m \times x}$ dimensional unit cube. This makes it easier to construct prior distributions for the width parameters to the correlation function $K(\cdot, \cdot)$. Proposals for all parameters which require MH sampling are taken from a uniform “sliding window” centered around the location of the last accepted setting. For example, a proposed a new nugget parameter g_ν to the correlation function $K(\cdot, \cdot)$ in region r_ν would go as

$$g_\nu^* \sim \text{Unif}\left(\frac{3}{4}g_\nu, \frac{4}{3}g_\nu\right).$$

Calculating the corresponding forward and backwards proposal probabilities for the MH acceptance ratio is straightforward.

For more details about the MCMC algorithm and proposals, etc., please see the original technical report on *Bayesian treed Gaussian process models* [15].

B Interfaces and features

The following subsections describe some of the ancillary features of the `tgp` package such as the gathering and summarizing of MCMC parameter traces, the progress meter, and an example of how to use the `predict.tgp` function in a collaborative setting.

B.1 Parameter traces

Traces of (almost) all parameters to the `tgp` model can be collected by supplying `trace=TRUE` to the `b*` functions. In the current version, traces for the linear prior correlation matrix (\mathbf{W}) are not provided. I shall illustrate the gathering and analyzing of traces through example. But first, a few notes and cautions.

Models which involve treed partitioning may have more than one base model (GP or LM). The process governing a particular input \mathbf{x} depends on the coordinates of \mathbf{x} . As such, `tgpr` records region-specific traces of parameters to GP (and linear) models at the locations enumerated in the `XX` argument. Even traces of single-parameter Markov chains can require hefty amounts of storage, so recording traces at each of the `XX` locations can be an enormous memory hog. A related warning will be given if the product of $|\mathbf{XX}|$, $(\text{BTE}[2] - \text{BTE}[1]) / \text{BTE}[3]$ and R is beyond a threshold. The easiest way to keep the storage requirements for traces down is to control the size of `XX` and the thinning level `BTE[3]`. Finally, traces for most of the parameters are stored in output files. The contents of the trace files are read into `R` and stored as `data.frame` objects, and the files are removed. The existence of partially written trace files in the current working directory (CWD)—while the `C` code is executing—means that not more than one `tgpr` run (with `trace = TRUE`) should be active in the CWD at one time.

Consider again the exponential data. For illustrative purposes I chose `XX` locations (where traces are gathered) to be (1) in the interior of the interesting region, (2) on/near the plausible intersection of partition boundaries, and (3) in the interior of the flat region. The hierarchical prior `bprior = "b0"` is used to leverage a (prior) belief the most of the input domain is uninteresting.

```
> exp2d.data <- exp2d.rand(n2 = 150, lh = 0, dopt = 10)
> X <- exp2d.data$X
> Z <- exp2d.data$Z
> XX <- rbind(c(0, 0), c(2, 2), c(4, 4))
```

We now fit a treed GP LLM and gather traces, and also gather EI and ALC statistics for the purposes of illustration. Prediction at the input locations `X` is turned off to be thrifty.

```
> out <- btgpllm(X = X, Z = Z, XX = XX, corr = "exp",
+   bprior = "b0", pred.n = FALSE, Ds2x = TRUE, R = 10,
+   trace = TRUE, verb = 0)
```

Figure 19 shows a dump of `out$trace` which is a `"tgptraces"`-class object. It depicts the full set of parameter traces broken down into the elements of a `list`: `$XX` with GP/LLM parameter traces for each `XX` location (the parameters are listed); `$hier` with traces for (non-input-dependent) hierarchical parameters (listed); `$linarea` recording proportions of the input space under the LLM; `$parts` with the boundaries of all partitions visited; `$post` containing (log) posterior probabilities; `preds` containing traces of samples from the posterior predictive distribution and adaptive sampling statistics.

Plots of traces are useful for assessing the mixing of the Markov chain. For example, Figure 20 plots traces of the range parameter (d) for each of the 3 predictive locations `XX`. It is easy to see which of the locations is in the same partition with others, and which have smaller range parameters than others.

The mean area under the LLM can be calculated as

```
> linarea <- mean(out$trace$linarea$la)
> linarea
```

```
> out$trace
```

This 'tgptraces'-class object contains traces of the parameters to a tgp model. Access is as a list:

- 1.) \$XX contains the traces of GP parameters for 3 predictive locations

Each of \$XX[[1]] ... \$XX[[3]] is a data frame with the columns representing GP parameters:

```
[1] index  lambda s2      tau2   beta0  beta1  beta2  nug
[9] d       b       ldetK
```

- 2.) \$hier has a trace of the hierarchical params:

```
[1] s2.a0  s2.g0  tau2.a0 tau2.g0 beta0  beta1  beta2
[8] d.a0   d.g0   d.a1   d.g1   nug.a0 nug.g0 nug.a1
[15] nug.g1
```

- 3.) \$linarea has a trace of areas under the LLM. It is a data frame with columns:

```
count: number of booleans b=0, indicating LLM
la: area of domain under LLM
ba: area of domain under LLM weighed by dim
```

- 4.) \$parts contains all of the partitions visited. Use the tgp.plot.parts.[1d,2d] functions for visuals

- 5.) \$post is a data frame with columns showing the following: log posterior (\$lpost), tree height (\$height), IS weights (\$w), tempered log posterior (\$tlpost), inv-temp (\$itemp), and weights adjusted for ESS (\$wess)

- 6.) \$preds is a list containing data.frames for samples from the posterior predictive distributions data (X) locations (if pred.n=TRUE: \$Zp, \$Zp.km, \$Zp.ks2) and (XX) locations (if XX != NULL: \$ZZ, \$ZZ.km, \$ZZ.ks2), with \$Ds2x when input argument ds2x=TRUE, and \$improv when improv=TRUE

Figure 19: Listing the contents of "tgptraces"-class objects.

```

> trXX <- out$trace$XX
> ltrXX <- length(trXX)
> y <- trXX[[1]]$d
> for (i in 2:ltrXX) y <- c(y, trXX[[i]]$d)
> plot(log(trXX[[1]]$d), type = "l", ylim = range(log(y)),
+      ylab = "log(d)", main = "range (d) parameter traces")
> names <- "XX[1,]"
> for (i in 2:ltrXX) {
+   lines(log(trXX[[i]]$d), col = i, lty = i)
+   names <- c(names, paste("XX[", i, "]", sep = ""))
+ }
> legend("bottomleft", names, col = 1:ltrXX, lty = 1:ltrXX)

```

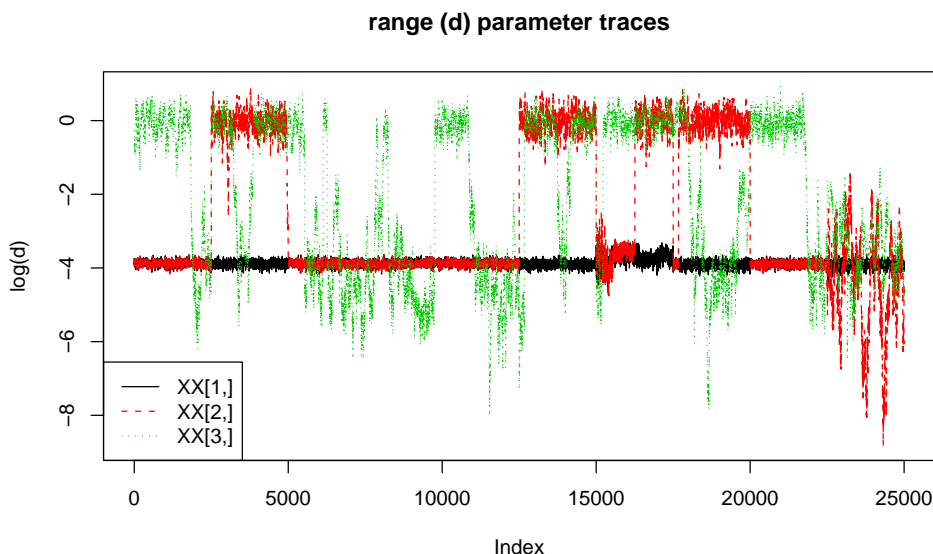


Figure 20: Traces of the (log of the) first range parameter for each of the three XX locations

```
[1] 0.530641
```

This means that the expected proportion of the input domain under the full LLM is 0.531. Figure 21 shows a histogram of areas under the LLM. The clumps near 0, 0.25, 0.5, and 0.75 can be thought of as representing quadrants (none, one, two, and tree) under the LLM. Similarly, we can calculate the probability that each of the XX locations is governed by the LLM.

```

> m <- matrix(0, nrow = length(trXX), ncol = 3)
> for (i in 1:length(trXX)) m[i, ] <- as.double(c(out$XX[i,
+ ], mean(trXX[[i]]$b)))
> m <- data.frame(cbind(m, 1 - m[, 3]))
> names(m) = c("XX1", "XX2", "b", "p1lm")
> m

```