Treed Gaussian Process Models for Classification

Tamara Broderick and Robert B. Gramacy

Statistical Laboratory, University of Cambridge
{tb361,bobby}@statslab.cam.ac.uk

Summary. Recognizing the successes of treed Gaussian process (TGP) models as an interpretable and thrifty model for nonstationary regression, we seek to extend the model to classification. By combining Bayesian CART and the latent variable approach to classification via Gaussian processes (GPs), we develop a Bayesian model averaging scheme to traverse the full space of classification TGPs (CTGPs). We illustrate our method on synthetic and real data and thereby show how the combined approach is highly flexible, offers tractable inference, produces rules that are easy to interpret, and performs well out of sample.

Key words: treed models, Gaussian process, model averaging, latent variable

1 Introduction and Background

A Gaussian process (GP) [7] is a popular nonparametric model for regression and classification that specifies a prior over functions. For ease of computation, typical priors often confine the functions to stationarity. While stationarity is a reasonable assumption for many data sets, still many exhibit only local stationarity. A treed Gaussian process (TGP) [5] represents a thrifty alternative (for the regression problem) that takes a local divide-and-conquer approach to nonstationary modeling. It defines a treed partitioning process on the predictor space and fits separate stationary GPs to the regions at the leaves. The treed form of the partition makes the model particularly interpretable.

We seek to extend the TGP model to classification. Separately, both treed models (CART [2] and Bayesian CART [3]) and GPs [6] have already been been successfully applied to classification. The machinery of treed partitions for a nonstationarity process and latent variables for classification suggests two possible combinations. We argue that one of the two offers clear advantages in terms of faster mixing in the resulting trans-dimensional Markov chain. Furthermore, we explore schemes for efficiently sampling the latent variables, which is important to obtain good mixing in the (significantly) expanded

parameter space compared to the regression case. Before delving further into the details we shall review the GP model for regression and classification.

1.1 Gaussian Processes for Regression and Classification

For real-valued p-dimensional inputs, a Gaussian process (GP) is formally a prior on the space of functions $Z : \mathbb{R}^p \to \mathbb{R}$ such that the function values $Z(x)$ at any finite set of input points x have a joint Gaussian distribution [9]. A particular GP is defined by its mean and correlation functions. The mean function $\mu(x) = \mathbb{E}(Z(x))$ is often constant or linear in the explanatory variable coordinates: $\mu(x) = f(x)\beta$, where $f(x) = [1, x]^\top$. The correlation function is defined as $K(x, x') = \sigma^{-2}[Z(x) - \mu(x)][Z(x') - \mu(x')]^\top$. We follow [5] and further assume that the correlation function can be decomposed into two components: a underlying strict correlation function K^* and a noise term of constant and strictly positive size g that is i.i.d. at the predictor points: $K(x_i, x_j) = K^*(x_i, x_j) + g\delta_{i,j}$. Here, $\delta_{i,j}$ is the Kronecker delta function, and g is called the *nugget*. It represents a source of measurement error and can offer improved numerical stability. A popular choice for $K^*(x, x')$ is the anisotropic squared exponential correlation:

$$K^*(x, x') = \exp\left\{ -\sum_{p=1}^{P} \frac{(x_p - x'_p)^2}{d_p} \right\}.$$

The strictly positive parameters d_p describe the *range* (or *length-scale*) of the process in each direction. Further discussion of appropriate correlation structures for GPs is provided by, e.g., [9]. The GP model features some notable drawbacks, including stationarity and computational cost (requiring the $O(N^3)$ inversion of an $N \times N$ matrix).

We may extend the GP model for regression to classification by introducing latent variables [6]. Here, the data consist of predictors X and classes $C \in \{1, \dots M\}$. For each class, we define a set of latent variables $\{Z_m\}_{m=1}^M$. For a particular class m, the latent variable generative model is a GP as before: $Z_m \sim \mathcal{N}(\mu_m(X), K_m(X, X))$. The class probabilities are now obtained from the latent variables via a softmax function:

$$p(C(x) = m) \propto \exp(-Z_m(x)). \tag{1}$$

Finally, the classes are drawn from a categorical distribution with these probabilities. In practice, we eliminate redundancy by including only $M - 1$ GPs and then set the last set of latent variables to zero. Similar drawbacks to GPs apply in the classification context, with the added complexity of $O(MN)$ extra latent variables that need to be estimated. Many of these issues are addressed by partitioning.

2 Treed Gaussian Processes

Fitting different, independent models to the data in separate regions of the input space naturally implements a globally nonstationary model. Moreover, dividing up the space results in smaller local covariance matrices, which are more quickly inverted. Finally, partitions offer a natural and data-inspired blocking strategy for latent-variable sampling in classification.

2.1 TGP for Regression

Treed models provide a partition process that is recursive, so arbitrary axis-aligned regions in the p-dimensional predictor space may be defined. Conditional on a treed partition, models are fit in each of the leaf regions. In CART [2] the underlying models are "constant" in that only the mean and standard deviation of the real-valued outputs are inferred. The tree is "grown" according to one of many decision-theoretic heuristics and may be "pruned" using cross-validation methods. In Bayesian CART (BCART), these models may be either constant [3] or linear [4] and, by contrast with CART, the partitioning structure is determined by Monte Carlo inference on the joint posterior of the tree and the models used at the leaves. In regression TGP (hereafter RTGP), the leaf models are GPs, but otherwise the setup is identical to BCART. Note that the constant and linear model are just special cases of the GP model. Thus RTGPs encompass BCART for regression, and inference may proceed according to a nearly identical Monte Carlo method, described shortly.

The hierarchical model for the RTGP begins with the tree prior, following [3]. Let $r \in \{1, \ldots, R\}$ index the R non-overlapping regions partitioned by the tree $\mathcal{T}$ drawn from the tree-prior. In the regression problem, each region contains data $\{X_r, Z_r\}$. Let a be the number of columns and n_r the number of rows in F_r, extending the predictor matrix to include an intercept term: $F_r = (1, X_r)$. A "constant mean" may be obtained with $F_r = 1$; in this case, $a = 1$. The generative model for the GP in region r incorporates the multivariate normal ($\mathcal{N}$), inverse-gamma (IG), and Wishart (W) distributions:

$$
\begin{aligned}
Z_r | \beta_r, \sigma_r^2, K_r &\sim \mathcal{N}_{n_r}(F_r \beta_r, \sigma_r^2 K_r) & \sigma_r^2 &\sim IG(\alpha_\sigma/2, q_\sigma/2) \\
\beta_r | \sigma_r^2, \tau_r^2, W, \beta_0 &\sim \mathcal{N}_a(\beta_0, \sigma_r^2 \tau_r^2 W) & \beta_0 &\sim \mathcal{N}_a(\mu, B) \\
\tau_r^2 &\sim IG(\alpha_\tau/2, q_\tau/2) & W^{-1} &\sim W((\rho V)^{-1}, \rho).
\end{aligned}
\tag{2}
$$

The hyperparameters $\mu, B, V, \rho, \alpha_\sigma, q_\sigma, \alpha_\tau, q_\tau$ are constant in the model.

We sample from the joint distribution of the tree structure $\mathcal{T}$, the R sets of GP parameters θ_r ($r = 1, \ldots, R$) in each region defined by $\mathcal{T}$, and the GP hyperparameters θ_0 (those variables in Eq. (2) that are not treated as constant but also not indexed by r) by Markov Chain Monte Carlo (MCMC). We sequentially draw $\theta_0|$rest, $\theta_r|$rest for each $r = 1, \ldots, R$, and $\mathcal{T}|$rest. Conditional on $\mathcal{T}$, all parameters (θ_r, $r = 1, \ldots, R$) and hyperparameters of the GPs can

be sampled with Gibbs steps, with the exception of the covariance function parameters $\{d_r, g_r\}$. Expressions are provided by [5].

Monte Carlo integration over tree space, conditional on the GP parameters θ_r, $r = 1, \ldots, R$, is more involved since the new tree structure drawn from the distribution $\mathcal{T}|$rest may have a different number of leaf nodes than its predecessor. Changing the number of leaf nodes changes the dimension of $\theta = (\theta_1, \ldots, \theta_R)$, so simple MH draws are insufficient in this case. Instead, reversible jump Markov Chain Monte Carlo (RJ-MCMC) allows a principled transition between models of different sizes [8]. In this framework, we mostly use the same four moves (*grow, prune, change, swap*) as in BCART to explore the tree space. These moves and their application are described in more detail in [5]. From the hierarchical model in Eq. (2) we can solve for the predictive distribution of the outputs Z. These are expressed in closed form by [5] and are provided later in this section (Eq. (3)) in the particular context of sampling from the latent variables used for classification.

2.2 TGP for Classification

One can envision at least two possible ways in which the latent variable approach and the treed approach to classification may be combined. The first method starts with an RTGP; recall that the RTGP tree partitions the predictor space into regions, each of which is assigned a stationary regression model (the GP). Analogously in the classification case, we could partition the predictor space into regions with a single tree and assign a stationary classification model (the CGP) to each region. Since this model contains just *one* tree, we call it the OTGP. The second method starts from the CGP; recall that the CGP uses $M - 1$ sets of real-valued latent variables generated from $M - 1$ stationary regression models (GPs). To introduce nonstationarity, instead consider $M - 1$ sets of real-valued latent variables generated from $M - 1$ nonstationary regression models (RTGPs). Since this model contains *multiple* trees, we call it the MTGP.

The MTGP has a variety of advantages over the OTGP. Indeed, the OTGP is a special case of the MTGP where the parameters, hyperparameters, tree structure, and hence region partitions are fixed across all trees. Thus, the MTGP represents natural splits in the data more easily and more interpretably. The MTGP requires fewer splits per tree than the OTGP. The MTGP also enjoys a higher Monte Carlo acceptance rate in tree space since, compared to OTGP moves, its moves are more local. In OTGP the data across all classes contribute to the acceptance probability, whereas in MTGP the acceptance of moves depends on how the particular class involved may be distinguished from all others. These last two considerations combine to ensure better mixing for the MTGP. Finally, from a practical standpoint, the MTGP is more directly implemented from the RTGP as it essentially amalgamates $M - 1$ of these models. Thus, we focus on the MTGP in what follows and

refer to it as the CTGP (TGP for classification) in analogy to the acronym RTGP.

The hierarchical model for the CTGP is straightforward. Given data (X, C), we introduce latent variables $\{Z_m\}_{m=1}^{M-1}$. Each of the corresponding trees $\{T_m\}_{m=1}^{M-1}$ divides the space into an independent region set of cardinality R_m. Each tree has its own, independent, RTGP prior where the hyperparameters, parameters, and latent variable values—for fixed class index m—are generated as in Eq. (2). It is most sensible to use a constant (rather than linear) mean in each of the leaves for the RTGP latent variables in the classification context. To approximate the joint distribution of the $M-1$ TGPs, we sample with RJ-MCMC much as in Section 2.1. Sampling is accomplished by visiting each tree in turn. For the mth class, we sequentially draw $\theta_{m,0}|\text{rest}$, $\theta_{m,r}|\text{rest}$ for each region r of R_m, $T_m|\text{rest}$, and finally the latent variables $Z_{m,r}|\text{rest}$ for each r. The first three draws are the same as for the RTGP. Drawing $Z_{m,r}|\text{rest}$ is the step unique to the CTGP.

While we cannot sample directly from $Z_{m,r}|\text{rest}$ to obtain a Gibbs sampling draw, we can factorize the full conditional for some subset of $Z_{m,r}$ into the distribution of the class given the latent variables at its predictor(s) $p(C(x_I)|\{Z_{m,r}(x_I)\}_{m=1}^{M-1})$ and the distribution of the latent variable(s) given the current GP together with the other latent variables in its region $p(Z_{m,r}(x_I)|X_r, \theta, T, Z_{m,r}\backslash Z_{m,r}(x_I))$. Then we can use MH and propose from the latter distribution. Here r labels the region within a particular class, and I is an index set over some of the predictors x in region r. To condense notation in Eq. (3), let $Z_I = Z_{m,r}(x_I)$ (similarly for F), and let $K_{I,I'} = K_{m,r}(x_I, x_{I'})$. Finally, $-I$ is the index set of points in region r of class m that are not in I. Then we have $Z_I|X_r, \theta, T, Z_{-I} \sim \mathcal{N}_{|I|}(\hat{z}, \hat{\sigma}^2)$ with

$$\hat{z} = F_I\tilde{\beta}_{-I} + K_{I,-I}K_{-I,-I}^{-1}(Z_{-I} - F_{-I}\tilde{\beta}_{-I})$$
$$\hat{\sigma}^2 = \sigma_r^2(\kappa_{I,I} - \kappa_{I,-I}\kappa_{-I,-I}^{-1}\kappa_{-I,I}), \quad \kappa_{I,I'} = K_{I,I'} + \tau_r^2 F_I W F_{I'}^\top, \quad (3)$$

where $\beta|X_I, Z_I, \theta, T \sim \mathcal{N}_a(\tilde{\beta}_I, V_I)$ using

$$V_I^{-1} = F_I^\top K_{I,I}^{-1}F_I + W^{-1}/\tau_r^2 \qquad \tilde{\beta}_I = V(F_I^\top K_{I,I}^{-1}Z_I + W^{-1}\beta_0/\tau_r^2).$$

In this setup the prior for Z cancels with the proposal probability in the acceptance ratio. The newly proposed Z may be accepted with probability equal to the likelihood ratio:

$$A = \prod_{i \in I} \frac{\exp(-Z'_{C(x_i),r}(x_i))}{\sum_{m=1}^M \exp(-Z'_{m,r}(x_i))} \times \frac{\sum_{m=1}^M \exp(-Z_{m,r}(x_i))}{\exp(-Z_{C(x_i),r}(x_i))}.$$

We may employ a blocking scheme to increase mixing in the marginal latent Z process; however there will natural be a trade-off in block size. Proposing all components of Z_m at once leads to a small acceptance ratio and poor mixing. But proposing each component of Z_m individually may result in only small,

incremental changes. An advantage of the treed partition is that it yields a natural blocking scheme for updating the latent variables. While we may block further within a leaf, this existing treed partition is a step forward from the CGP.

3 Illustrations and Empirical Results

We illustrate CTGP and compare it to CGP on real and synthetic data by making timings and calculating misclassification rates. Since the most likely class label at a particular predictor value corresponds to the largest latent variable at that predictor (via Eq. (1)), we may predict the class labels by first keeping a record of the predicted class labels at each round of the Monte Carlo run and then taking a majority vote upon completion.

3.1 2d Exponential Data

Consider the synthetic 2d exponential regression data, where the input space is $[-2, 6] \times [-2, 6]$, and the true response is given by the 2d exponential function $z(x) = x_1 \exp(-x_1^2 - x_2^2)$. To convert the real-valued outputs to classification labels we calculate the Hessian H. Then, for a particular input (x_1, x_2) we assign a class label based on the sign of the sum of the eigenvalues of $H(x_1, x_2)$, indicating the direction of concavity at that point. A function like the 2d exponential whose concavity changes more quickly in one region of the input space than in another (and is therefore well fit by an RTGP model) will similarly have class labels that change more quickly in one region than in another. The *left-hand* side of Figure 1 shows the resulting class labels. Overlaid on the
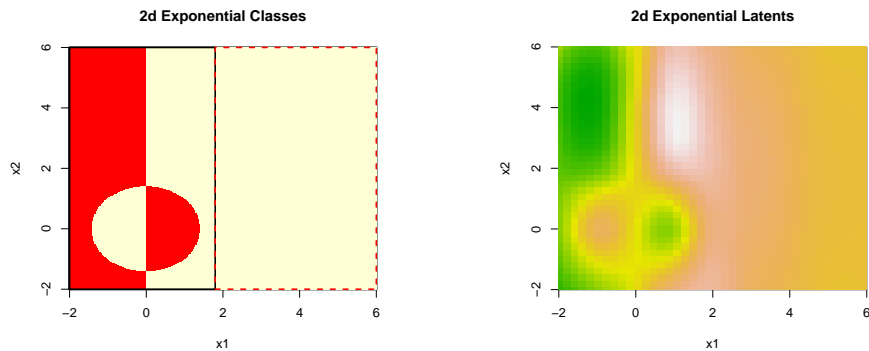

Fig. 1. 2d exponential data (*left*) and mean latent variables (*right*).

plot is the maximum *a posteriori* tree encountered in the trans-dimensional Markov chain sampling from the CTGP posterior. We trained the classifier(s)

on (X, C) data obtained by a maximum entropy design of size $N = 400$ sub-sampled from a dense grid of 10000 points and calculated the misclassification rate on the remaining 9600 locations. The rate was 3.3% for CGP and 1.7% for CTGP, showing a relative improvement of roughly 50%. CTGP wins here because the relationship between response (class labels) and predictors is clearly nonstationary. The speed improvements obtained by partitioning were even more dramatic. CGP took 21.5 hours to execute 15000 RJ-MCMC rounds, whereas CTGP took 2.0 hours, an over 10-fold improvement. The *right-hand* plot in Figure 1 shows the posterior mean of the latent variables under the CTGP model.

3.2 Classification TGP on Real Data

Consider the Credit Approval data set that may be obtained from the UCI Machine Learning database [1]. The set consists of 690 instances grouped into two classes: credit card application approval $(+)$ and application failure $(-)$. The names and values of the fifteen predictors for each instance are confidential. However, aspects of these attributes relevant to our classification task are available. E.g., we know that six inputs are continuous, and nine are categorical. Among the categorical predictors, the number of distinct categories ranges from two to fourteen. After binarization, we have a data set of 6 continuous and 41 binary predictors. The CGP treats these all as continuous attributes. We restrict the CTGP to form GPs only over the six continuous attributes and to apply the treed partition process on (and only on) the 41 binary attributes.

Our comparison consists of ten separate 10-fold cross-validations for a total of 100 folds. The average misclassification rate of the CGP across these folds was 14.6% (4.0%). The CTGP offers a slight improvement with a rate of 14.2% (3.6%). More impressive is the speed-up offered by CTGP. The average CPU time per fold used by the CGP method was 5.52 hours; with an average CPU time per fold of 1.62 hours, the CTGP showed a more than three-fold improvement.
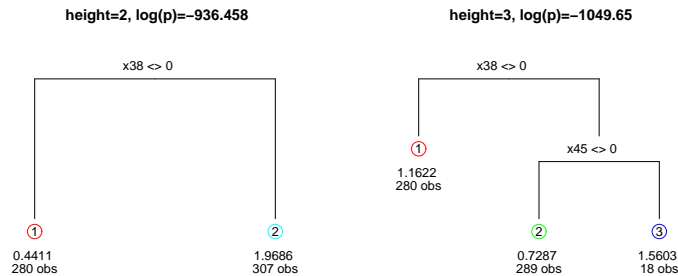

Fig. 2. Trees from CTGP on the Credit Approval data.

Finally, the interpretative aspect of the CTGP is worth highlighting. For a particular run of the algorithm on the Credit Approval data, the MAP trees of different heights are shown in Figure 2. These trees, and those for other runs, feature principal splits on the 38th binary predictor, which corresponds to the 9th two-valued categorical predictor. Therefore, the CTGP indicates, without additional work, the significance of this variable in predicting the success of a credit card application. To extract similar information from the CGP, one would have to devise and run some additional tests—no small feat given the running time of single CGP execution.

4 Conclusion

In this paper we have illustrated how many of the benefits of the regression TGP model extend to classification. The components of TGP, i.e. treed models and GPs, have separately long enjoyed success in application to classification problems. In the case of the GP, $M - 1$ processes are used as a prior for latent variables which encode the classes via a softmax function. While this is a powerful method which typically offers improvements over simpler approaches (including treed models), drawbacks include an implicit assumption of stationarity and slow evaluation due to repeated large matrix decompositions. In contrast, the treed methods provide a thrifty divide-and-conquer approach. The combined tree and GP approach provides a classification model that is speedy, interpretable, and highly accurate, combining the strengths of GP and treed models for classification.

References

1. Asuncion, A. and Newman, D. (2007). UCI Machine Learning Repository.
2. Breiman, L., Friedman, J. H., Olshen, R., and Stone, C. (1984). *Classification and Regression Trees*. Belmont, CA: Wadsworth.
3. Chipman, H., George, E., and McCulloch, R. (1998). Bayesian CART model search (with discussion). *J. of the American Stat. Assoc.*, 93, 935–960.
4. — (2002). Bayesian treed models. *Machine Learning*, 48, 303–324.
5. Gramacy, R. B. and Lee, H. K. H. (2008). Bayesian treed Gaussian process models with an application to computer modeling. *Journal of the American Statistical Association*, 103, 1119–1130.
6. Neal, R. M. (1998). Regression and classification using Gaussian process priors (with discussion). In *Bayesian Statistics 6*, ed. e. a. J. M. Bernardo, 476–501. Oxford University Press.
7. Rasmussen, C. E. and Williams, C. K. I. (2006). *Gaussian Processes for Machine Learning*. The MIT Press.
8. Richardson, S. and Green, P. J. (1997). On Bayesian analysis of mixtures with an unknown number of components. *Journal of the Royal Statistical Society, Series B, Methodological*, 59, 731–758.
9. Stein, M. L. (1999). *Interpolation of Spatial Data*. New York, NY: Springer.