Variable Selection and Sensitivity Analysis via Dynamic Trees with an Application to Computer Code Performance Tuning

Robert B. Gramacy & Matthew A. Taddy
The University of Chicago Booth School of Business

Stefan M. Wild
Argonne National Laboratory

ABSTRACT: We show how the newly developed dynamic tree model can support variable selection and a sensitivity analysis of inputs, two tasks usually requiring disparate model structure. To this end, we adapt methods used in conjunction with static tree models and Gaussian process models (GPs). Compared with static trees, this approach allows for dynamic (sequential) variable selection with fully Bayesian evidence and sensitivity indices not previously enjoyed. Compared with GPs, it facilitates sensitivity analysis and variable selection on problems that are several orders of magnitude larger in size than previously possible. Importantly, this allows a unifying approach to mixed data applications. The same computational techniques can be used, for example, in regression and classification applications with trivial modification. We illustrate our methodology on several instructive data sets and then analyze automatic computer code tuning that combines classification and regression elements, demanding both variable selection and input sensitivity analysis on large data sets.

KEYWORDS: Sensitivity analysis, variable selection, main effects, sensitivity indices, exploratory data analysis, data visualization, particle filtering, CART, computer experiment.

Gramacy is Assistant Professor and Kemper Foundation Faculty Scholar, and Taddy is Assistant Professor and Neubauer Family Faculty Fellow, both in the Econometrics and Statistics group at the University of Chicago Booth School of Business, 5807 S. Woodlawn Avenue, Chicago, IL 60637; [rbgramacy, taddy]@chicagobooth.edu, http://faculty.chicagobooth.edu/[robert.gramacy, matt.taddy]. Wild is an Assistant Computational Mathematician in the Mathematics and Computer Science Division at Argonne National Laboratory, 9700 S. Cass Avenue, Bldg. 240-1154, Argonne, IL 60439, and a Fellow in the Computation Institute at the University of Chicago; wild@mcs.anl.gov, http://www.mcs.anl.gov/∼wild.

1 Introduction

In any regression analysis, it is essential to quantify the influences on the response by individual candidate explanatory variables. This assessment should cover an array of information, attributing direction, strength, and evidence to covariate effects, both when acting independently and when interacting. For linear statistical models, various well-known tools are available for the task. In ordinary least-squares, for example, there are t and F tests for the effect of predictor(s), ANOVA to decompose variance contributions, and *leverages* to measure influence in the input space. Such tools are fundamental to applied linear regression analysis.

In contrast, analogous techniques tend to be unavailable for more complicated nonparametric regression methods, such as neural networks and other basis expansions, or Gaussian processes (GPs) and other stochastic process models. These are precisely the models used in the broad engineering problems that motivate this work — optimization under uncertainty and emulation of computer simulators — where regression analysis must allow for both nonlinearity and nonconstant variance. Unfortunately, because of the lack of easy-to-apply tools, it is common to treat the probability model as a black-box prediction machine and to neglect the analysis of covariance that is standard in applications of linear regression.

In this paper, we resolve the tension between flexibility and interpretability by detailing a framework that provides both, without need for compromise. We argue that *dynamic trees* (DTs), introduced in Taddy et al. (2011), are a uniquely appropriate platform for both predictive modeling and analysis of covariance in complex regression settings. Part of our argument holds generally for trees: partitioning of the covariate space, the same quality that is key to model flexibility, acts as a natural and interpretable foundation for attribution of variable influence. Distinct from most other tree methods, however, DTs are accompanied by an efficient approach to posterior simulation and provide full uncertainty quantification for each metric of covariance analysis, hence allowing for proper consideration of statistical evidence. Moreover, although the examples here involve batch data analysis, model estimation is inherently on-line and naturally suited to the analysis of sequential data.

We present two complementary analyses: *variable selection* (VS) and *sensitivity analysis* (SA). The first focuses on selecting the subset of covariates that should be included in the model, in that they lead to predictions of low variance and high accuracy. The second seeks to characterize how elements of this subset influence the response. As discussed in more detail in Section 1.2, it is most common to focus on only one of these two tasks: variable selection is common in additive models, where the structure for covariance is assumed rather than estimated, whereas in more complicated functional SA settings the set of covariates is taken as given. This methodological split is unfortunate, as VS and SA work best together, with SA as a higher-fidelity analysis that follows in-or-out decisions made in VS. Hence, we have found that the use of DT models as a platform for both VS and SA is a powerful tool in applied statistics, and this article's goal is to facilitate and illustrate application of these techniques.

Following Section 2's review of DT methods, Sections 3 and 4 contain details for our approaches to VS and SA, respectively. In both cases we take advantage of the unique construction of DT models to improve on standard methodology, and each section concludes with an illustration of the methods and a brief comparison study. These methodological contributions are directly motivated by applications in performance tuning of scientific codes, an area of research that is growing to cope with increasing heterogeneity and turnover in computer archi-

tecture (see, e.g., Vuduc et al., 2004). Section 5 details analysis and results for a large-scale experiment on the optimization of linear algebra kernels. Section 6 provides a concluding discussion, including a brief description of future work.

1.1 A Motivating Example

To achieve higher performance of a given code, annotation scripts [e.g., Orio (Hartono et al., 2009)] can apply code transformations such as loop reordering to the original source code to generate a modified version of the code. The output code can then be compiled in a variety of ways (i.e., by setting compiler optimization flags), resulting in an executable that runs more or less quickly on a particular machine depending on the nature of the transformation, compilation, machine architecture, and original source code. Because of the high variability and complexity of modern architectures, codes must increasingly be tuned specifically for a targeted machine. Hence, given a machine architecture, the goal is to study how the transformation and compilation options can be used to minimize code execution times under the constraint that the resulting code yields correct numerical output. As evidenced by the success of the ATLAS project (`http://math-atlas.sourceforge.net/`), even minor performance gains for basic computational kernels can be significant when called repeatedly.

In this paper, we focus on a dataset comprising various code transformations and their corresponding execution times from Balaprakash et al. (2011a). In the design of each experiment (the input source code), a subset of the possible transformation/compilation options (inputs) was thought to yield correct numerical outputs, and these were varied in full enumeration over the input space to obtain execution times. Some of the inputs are ordinal and some categorical. Our task is to measure the relative importance of each input for predicting/determining the execution time, to explore how each (relevant) input contributes to the execution time marginally and (to the extent possible) conditionally, and to check whether there is any predictable pattern in the constraint violations that arise for a small subset of the combination of inputs. For the kernels considered, the constraint violations are associated with unsuccessful compilation of the transformed code and/or segmentation faults for the resulting executable.

A second goal is to investigate, and possibly control for, the *cold cache effect*. This effect, due to compulsory cache misses sometimes arising from initial accesses to a cache block, is also referred to as *cold-start misses* (Patterson and Hennessy, 2007) and can cause the first execution instance to run slower than subsequent instances. Basically, the question is whether any of the inputs to the transformation/compilation program affect the size of the cold cache effect, whether it decays persistently beyond the second instance, and whether modeling it is important for understanding the effect of various transformations on execution times. While recent works (e.g., Balaprakash et al., 2011b) have focused on defining input spaces for performance tuning problems, formulating appropriate objectives in the presence of the cache effects and other operating system noise remains an unresolved issue.

1.2 Methodology Review

VS is often equated with setting coefficients to zero. Hence, the approaches are predicated on a specific, usually additive, form for the influence of covariates on response. In the analysis of computer experiments, for example, Reich et al. (2009) use the COSSO model of Lin and

Zhang (2006); Cantoni et al. (2011) use the nonnegative garrote (NNG, Breiman, 1995); and Huang et al. (2010) apply grouped LASSO. However, because of the complexity of the modeled processes and a need for high precision, researchers using statistical emulation for engineering processes are seldom content with a single additive regression structure for the entire input space. Moreover, the consideration of interaction terms in additive models can require huge, overcomplete bases, leading to burdensome computation. As a result, it is more common to rely on GP priors or other nonparametric regression techniques (e.g., Bayarri et al., 2009; Sansó et al., 2008). Since such modeling significantly complicates VS (e.g., Linkletter et al., 2006), it is rare to see principled covariate selection in application.

Instead of a dedicated VS procedure, engineering applications commonly employ some form of SA. In its classic treatment, as in examples from Saltelli et al. (2000, 2008), running the computer code to obtain a response is presumed to be cheap. When it is expensive, it becomes necessary to emulate the code with an estimated probability model (see Santner et al., 2003, for an overview). In turn, authors have proposed a variety of schemes for extension of classic SA to account for response surface uncertainty. Because of their role as the canonical choice for modeling computer experiments, GPs commonly are combined with SA in applications (e.g., Ziehn and Tomlin, 2009; Marrel et al., 2009). However, the associated methodology is usually based on restrictive stationarity and homoskedasticity assumptions needed to derive either empirical Bayes (Oakley and O'Hagan, 2004) or fully Bayesian (Morris et al., 2008; Farah and Kottas, 2011) estimates of sensitivity indices. For SA under flexible alternatives to the stationary GP, as in Storlie et al. (2009), authors resort to ad hoc estimation techniques for the metrics of interest. A notable exception to this is presented by Taddy et al. (2009), where the integration is embedded within posterior simulation to obtain samples from SA indices' posterior distribution. A similar idea sets the foundation for our generic SA framework in Section 4.

Partition trees (e.g., CART: Breiman et al., 1984) provide a basis for regression that has both a simplicity amenable to VS and the flexibility required for modeling computer experiments. Furthermore, partition trees overcome some well-known drawbacks of the more commonly applied GP computer emulators: expensive $\mathcal{O}(n^3)$ matrix inversion, clumsy handling of categorical predictors, and difficulty in allowing for nonstationarity or heteroskedastic errors. Taddy et al. (2011) provide extensive background on general tree-based regression and argue for its wider adoption in engineering applications. In the context of this paper's goals, trees present a unique nonadditive foundation for VS. In their most simple form, with constant mean response at the tree leaves, VS is automatic: if a variable is never used to define a tree partition, it has been effectively removed from the regression. Indeed, this idea motivated some of the earliest work on the use of trees, as in Morgan and Sonquist (1963), for automatic interaction detection. In a more nuanced approach to VS, Breiman et al. (1984) introduced indices of variable importance that measure squared error reduction due to tree-splits defined on each covariate. Hastie et al. (HTF; 2009, Chap. 10) promote these indices for sensitivity analysis and describe how the approach can be extended for their boosted trees.

However, these techniques are purely algorithmic and lack a full probability model; hence, their use is especially problematic in analysis of computer experiments, where uncertainty quantification is often a primary objective. Moreover, the HTF importance indices are only point estimates of the underlying sensitivity metrics; thus, they preclude basing VS on posterior evidence and make it difficult to properly interpret the SA. Authors have attempted to

overcome some of these shortcomings through use of Bayesian inference, most recently in schemes that augment the tree model to allow for better control or flexibility. Chipman et al. (2010) describe a mixture model for regression trees and their `BayesTree` software includes a direct analogue of the HTF importance indices; and the method of Taddy et al. (2009) is implemented in `tgp` (see Gramacy and Taddy, 2010, Section 3).

Despite this significant existing work, the current article's contribution is the most complete treatment that we know of for both VS and SA in partition trees. Regarding VS, we argue for novel measures of variable importance that are derived from a full probability model and can be characterized through posterior sampling.These indices can also be interpreted as tools for SA, yielding a unified approach to covariance analysis with trees. For more detailed SA concerning each variable's main as well as total effect on the response, we show that the Monte Carlo routine of Gramacy and Taddy (2010) can be applied in generic posterior simulation settings. Everything is founded on the platform of dynamic trees, outlined below.

2 Dynamic Tree Models

Dynamic trees (DTs; Taddy et al., 2011) are the process–analog of Bayesian treed models (Chipman et al., 1998, 2002). The model specification and prior are defined sequentially for fast updating of predictive surfaces that exhibit organically increasing complexity as new data arrive. Here we review model specification and inference. Software is available in the `dynaTree` (Gramacy and Taddy, 2011) package, which has been extended to cover the techniques described in this paper.

2.1 Model and Prior Specification

Model specification starts with the tree prior from Chipman et al. (1998): a leaf node η may split with probability $p_{\text{split}}(\mathcal{T}, \eta) = \alpha(1 + D_\eta)^{-\beta}$, where $\alpha, \beta > 0$ and D_η is the depth of η in the tree $\mathcal{T}$. Partition locations are chosen uniformly over the data input locations $\mathbf{x}^\eta$ in η. The joint prior is then determined by the probability that the internal nodes $\mathcal{I}_\mathcal{T}$ have split and the leaves $\mathcal{L}_\mathcal{T}$ have not: $\pi(\mathcal{T}) \propto \prod_{\eta \in \mathcal{I}_\mathcal{T}} p_{\text{split}}(\mathcal{T}, \eta) \prod_{\eta \in L_\mathcal{T}} [1 - p_{\text{split}}(\mathcal{T}, \eta)]$. In the old, static version the model specification is completed with a likelihood comprising independent sampling models employed at the leaves: $p(y_1, \ldots, y_n | \mathcal{T}, \mathbf{x}_1, \ldots, \mathbf{x}_n) = \prod_{\eta \in \mathcal{L}_\mathcal{T}} p(y^\eta | \mathcal{T}, \mathbf{x}^\eta)$. The $p(y^\eta | \mathcal{T}, \mathbf{x}^\eta)$ are marginal likelihoods where parameters have been (analytically) integrated over their priors. Sampling from the posterior distribution proceeded by MCMC, via stochastic proposals for local changes to $\mathcal{T}$: *grow*, *prune*, *change*, and *swap* moves. Any data type/model may be used as long as the marginal likelihood calculation is analytic.

In DTs the moves are imbedded into the process, which describes how old trees mature into new trees when new data arrive. Suppose that $\mathcal{T}_{t-1}$ includes the recursive partitioning rules associated with $\mathbf{x}^{t-1}$, the set of covariates observed up to time $t-1$. The DT process is defined by the set of trees $\{\mathcal{T}_t\}$ which are reachable from $\mathcal{T}_{t-1}$ when $\mathbf{x}_t$ arrives. These are determined by three equally probable, local moves on the leaf node $\eta(\mathbf{x}_t) \in \mathcal{T}_{t-1}$: *stay*, *prune*, and *grow*. This has the effect of making the (prior) tree process data dependent (but only on $\mathbf{x}$, not y). While seemingly restrictive, it is this aspect that makes local "searches" efficient.

Stay moves keep the tree hierarchy unchanged, so that $\mathcal{T}_t = \mathcal{T}_{t-1}$. *Prune* moves remove $\eta(\mathbf{x}_t)$ as well as its sibling's subtree so that the parent of $\eta(\mathbf{x}_t)$ is a leaf in $\mathcal{T}_t$. *Grow* moves create

a new pair of leaf nodes in $\mathcal{T}_t$ by randomly proposing to split on one of the data locations in $\eta(\mathbf{x}_t)$. Choosing uniformly over these three moves completes the specification of $p(\mathcal{T}_t|\mathcal{T}_{t-1}, \mathbf{x}^t)$. The new observation, y_t, provides a stochastic rule for determining how the update $\mathcal{T}_{t-1} \rightarrow \mathcal{T}_t$ occurs, via $p(y^t|\mathcal{T}_t, \mathbf{x}^t)$ for each possible $\mathcal{T}_t$. As in the static tree model, any data type may be modeled in this way as long as the leaf marginal likelihood is analytic. However, efficient inference [see Section 2.2] and our SA extensions [Section 4] further require that the within-leaf posterior predictive distribution be of a standard form. Taddy et al. (2011) show that this is the case for three important leaf models with appropriate priors: *constant* and *linear* models for regression and *multinomial* models for classification.

2.2 Inference

Like with many pragmatic modeling choices, the DT specification is "tuned" to be efficient under particular inferential mechanics, in this case the sequential Monte Carlo method of *particle learning* (Carvalho et al., 2010). At each iteration t, the discrete approximation to the tree posterior $\{\mathcal{T}_{t-1}^{(i)}\}_{i=1}^N$, based on N particles, is updated to $\{\mathcal{T}_t^{(i)}\}_{i=1}^N$ by applying first a *resample* step and then a *propagate* step. The *resample* step first weights the particles according to the predictive probability of the next $\mathbf{x}$–y pair, $w_i = p(y_t|\mathcal{T}_{t-1}^{(i)}, \mathbf{x}_t)$, and then resamples them by replacement according to the weights. The *propagate* step updates the resampled particles following the process outlined in Section 2.1. The calculations involved in both steps are efficient because they are local (they apply only to the subtrees of the parent of each $\eta^{(i)}(\mathbf{x})$) and because the updates to the sufficient statistics (for updating marginal likelihood calculations) at those subtrees involve standard recursions.

As with all particle simulation methods, some Monte Carlo error will accumulate, and in practice one must be careful to assess its effect. Taddy et al. (2011) show that, despite inevitable particle degeneracy, DT out-of-sample performance is comparable to GPs and other highly non-parametric methods in both regression and classification contexts, but at a fraction of the computational cost. A nice byproduct of particle learning inference for DTs is reliable marginal likelihoods via the sequential factorization $p(y^T|\mathbf{x}^T) \approx \frac{1}{N}\sum_{i=1}^N \sum_{t=1}^T \log p(y_t|\mathbf{x}_t, \mathcal{T}_{t-1}^{(i)})$, i.e., using the probabilities calculated in the resample step. Taddy et al. (2011) used these to compare leaf model specifications (e.g., constant and linear) via Bayes factors (BFs). These come in handy as a backstop to heuristic VS schemes, as outlined below.

3 Variable Selection

As mentioned in the introduction, tree models engender basic VS through the estimation of split locations — any variable not split upon has been deselected. However, this binary determination does not provide any spectrum of variable importance, and the unavailable null distribution for tree splits can lead to inclusion of spurious variables. Hence, we need measures of covariate influence that are based on analysis of response variance and that will establish evidence for variable inclusion. Taking advantage of the full probability model characterized by DTs, fitted trees can be used to obtain probabilistic measures of variable importance.

3.1 Measuring the Importance of Predictors

Following the basic logic of tree-based variable selection, variables contribute to reduction in predictive variance through each split location. We will label the leaf model-dependent uncertainty reduction for each node η as $\Delta(\eta)$. Grouping these by variable, we obtain the importance index for each covariate $k \in 1, \ldots, p$ as

$$J_k(\mathcal{T}) = \sum_{\eta \in \mathcal{I}_\mathcal{T}} \Delta(\eta) \mathbb{1}_{[v(\eta)=k]}, \tag{1}$$

where $v(\eta) \in \{1, \ldots, p\}$ is the splitting dimension of η. Through efficient storage of data and split rules, these indices are inexpensive to calculate for any given tree. Given a filtered set of trees, as described in Section 2, the implied sample of J_k indices provides a full posterior distribution of importance for each variable; this can form the basis for model-based VS.

For $\Delta(\eta)$ we consider the decrease in predictive uncertainty associated with the split in η. In regression, the natural choice is the average reduction in predictive variance,

$$\Delta(\eta) = \int_{A_\eta} \sigma^2_\eta(\mathbf{x})\, d\mathbf{x} - \int_{A_{\eta_\ell}} \sigma^2_{\eta_\ell}(\mathbf{x})\, d\mathbf{x} - \int_{A_{\eta_r}} \sigma^2_{\eta_r}(\mathbf{x})\, d\mathbf{x}, \tag{2}$$

where η_ℓ and η_r are η's children, $\sigma^2_\eta(\mathbf{x})$ is the predictive variance at $\mathbf{x}$ in the node η, and A_η is the bounding covariate rectangle for that node. Rectangles on the boundary of the tree are constrained to the observed variable support and, from recursive partitioning, $A_\eta = A_{\eta_\ell} \cup A_{\eta_r}$.

For constant leaf-node models, each integral in (2) is simply the area of the appropriate rectangle multiplied by that node's predictive variance. For classification, we replace the predictive variance at each node with the predictive entropy based on $\hat{p}_c$, the posterior predicted probability of each class c in node η. This leads to the entropy reductions $\Delta(\eta) = |A_\eta| H_\eta - |A_\ell| H_\ell - |A_r| H_r$, where $H_\eta = -\sum_c \hat{p}_c \log \hat{p}_c$. Since the rectangle area calculations involve high-dimensional recursive partitioning and can be both computationally expensive and numerically unstable, a Monte Carlo alternative is to replace $|A|$ with n, the number of data points in η. We find that this provides a fast and accurate approximation.

A regression tree with linear leaves presents a more complex setting, since the reduction in predictive variance is not constant over each partition. In the appendix, we show that the calculations in (2) remain available in closed form. However, since in this case covariates also affect response through the linear leaf model, (1) provides only a partial measure of variable importance. In Section 4 we describe alternative SA metrics whose interpretations do not depend on leaf model specification.

3.2 Selecting Variables

The posterior sample $\{J_k(\mathcal{T}_t^{(i)})_{i=1}^N\}$ can be used to assess the importance of each predictor $k = 1, \ldots, p$, through both graphical visualization and ranking of summary statistics. As a tool for VS, we advocate estimated relevance probability, $\mathbb{P}(J_k(\mathcal{T}) > 0) \approx \frac{1}{N} \sum_{i=1}^N \mathbb{I}_{\{J_k(\mathcal{T}_t^{(i)})>0\}}$.[1] A backwards stagewise procedure based on this criterion, illustrated in the examples below, is

[1]Note that $\Delta(\eta)$, and thus $J_k(\mathcal{T}_t^{(i)})$ for particle i, may be negative for some $\eta \in \mathcal{I}_{\mathcal{T}_t^{(i)}}$ because of the uncertainty inherent in our Monte Carlo posterior sample.

to repeatedly refit the trees after deselecting variables whose relevance probability is less than a certain threshold. The algorithm terminates when the mean sequential BF for the most recently fit model, easily available as described in Section 2.2, no longer indicates a strong preference for the simpler model. In practice, we find that this process converges in one or two iterations.

3.3 Examples

We present here examples using two datasets: synthetic data and spam data.

Simple synthetic data

We consider data first used by Friedman (1991) to illustrate multivariate adaptive regression splines (MARS) and then used by Taddy et al. (2011) to demonstrate the competitiveness of DTs relative to modern (batch) nonparametric models. The input space is 10-dimensional; however, the response, given by $10\sin(\pi x_1 x_2) + 20(x_3 - 0.5)^2 + 10x_4 + 5x_5$ with $\mathrm{N}(0,1)$ additive error, depends only on five of the predictors. We used $N = 10000$ particles and $T = 1000$ input-output pairs sampled uniformly in $[0,1]^{10}$. Following Taddy et al. (2011), we repeated the process ten times to understand the nature of the MC error on our VS procedure.

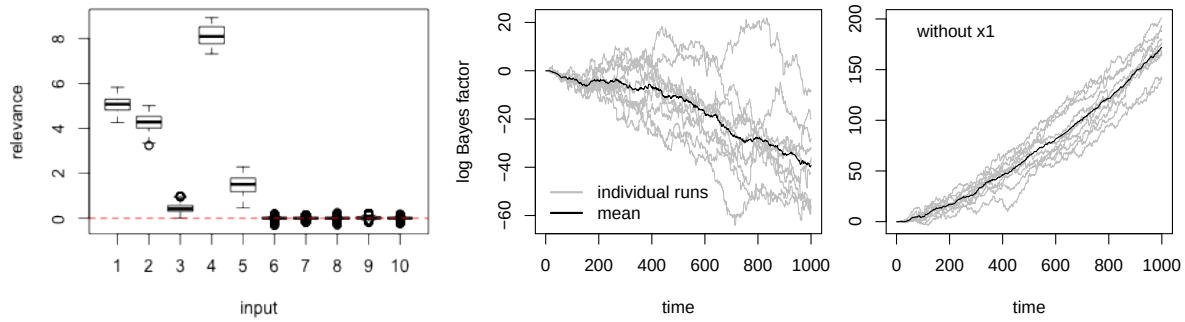

Figure 1: Variable selection in the Friedman data. The boxplots on the *left* show the posterior relevance. The *right* two plots show (log) BFs, first for the full predictor set versus the set reduced to the five relevant variables, and then when one of the relevant variables is removed.

The results are summarized in Figure 1. The boxplots on the left show the cumulative 100,000 samples of the tallied relevance statistics for each variable. The (latter) five useless variables are easily identified, since their relevance statistics tightly straddle zero. After removing these variables we re-ran the fitting procedure (again ten times) and calculated (log) BFs, treating the smaller model as the null (i.e., in the denominator). All ten (log) BF paths (*center* panel) clearly indicate that the larger model is not supported by the data. In fact, there is a decreasing trend in the BF suggesting that the smaller model is actually a better fit. So while deselecting irrelevant variables is not technically necessary, doing so becomes increasingly important as the data length grows relative to a fixed-sized particle cloud (i.e., in order to ward off particle depletion). The final, right panel in the figure shows the (log) BF calculation that would result had we further considered the first input for deselection (i.e., suggesting only inputs 2–5 were important). Clearly, the larger model (in the numerator) is strongly preferred.

Spam data

We turn now to the Spambase data set, from the UCI Machine Learning Repository (Asuncion and Newman, 2007). The aim is to illustrate our VS procedure in a classification context, but also to scale up to larger n and p with significant interaction effects. The data contains binary classifications of 4,601 emails based on 57 attributes (predictors). The left panel of Figure 2

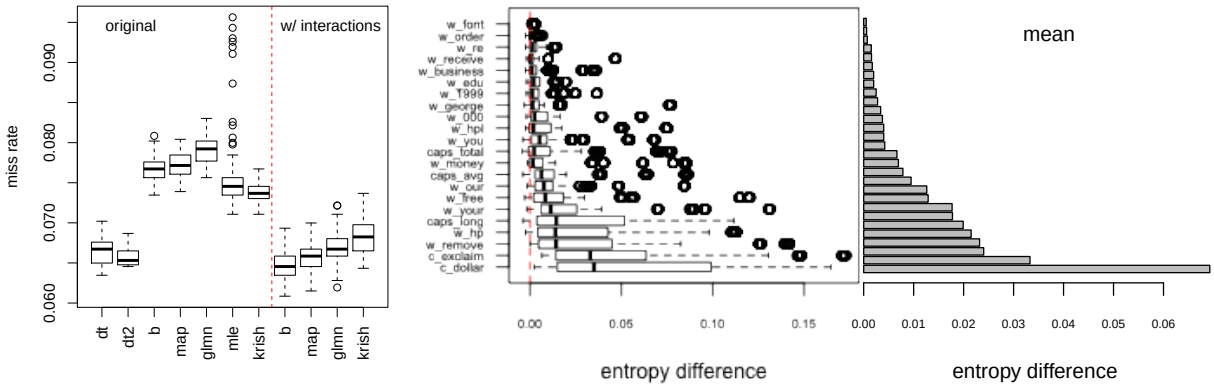

Figure 2: *(Left)* Boxplots of misclassification rates divided into two sections, depending on absence or presence of interaction terms in the design matrix. *(Right panels)* Posterior samples relevance statistics, and their means.

shows the results of a MC experiment based on misclassification rates obtained using random fivefold cross validation training/testing sets sets, repeated twenty times for 100 sets total. The comparators are modern regularized logistic regression models, including fully Bayesian and maximum a posteriori (MAP) estimators via Gibbs sampling (Gramacy and Polson, 2010), an estimator from the `glmnet` package (Friedman et al., 2010), and the EM-based method of (Krishnapuram et al., 2005). Results for these comparators on an interaction-expanded set of (≈ 1700) predictors are also provided. Expansion is crucial to realize good performance from the logistic models.

Our DT contributions are `dt` and `dt2`, each using $N = 1000$ particles and 30 repeats, which took about half the execution time of the interaction-expanded logistic comparators. The `dt2` estimator is the result of a single iteration of the VS procedure outlined above, leveraging the $\{J_k\}$ obtained from the initial `dt` run. This usually resulted in 25 (of 57) deselections. The subsequent BF calculation(s) indicated a preference for the small model in every case considered. Notable results include the following. The DT-based estimators perform as well as the interaction-expanded linear model estimators, without explicitly using the expanded predictor set. That is, the trees naturally exploit interaction, which ports over into the VS procedure without any extra effort. Hence, the resulting estimator (`dt2`) is just as good as the former (`dt`, using the entire set of predictors) but with lower MC error. In fact, based on the worst cases in the MC experiment, `dt2` is the best estimator in this study. We also found that marginal improvements can be obtained with further deselection stages.

The right panels of Figure 2 show the posterior samples of the entropy difference tallies for predictors whose median relevance was greater than zero, and the corresponding posterior means (by which they have been ordered). A similar plot is given for random forests in HTF (HTF, Figure 10.6). Our ordering of importance is similar, but importance drops off more

quickly as our single-tree model is more parsimonious than the additive trees of random forests. Perhaps even more notable, our posterior analysis emits uncertainty estimates from which we can readily observe a high degree of multi covariability. Apparently, even the most marginally relevant predictors can be masked when conditioning on others.

4 Sensitivity Analysis

The importance indices of Section 3 provide a computationally efficient measure of a covariate's first order effect — variance reduction directly attributed to splits on that variable. These indices are not, however, appropriate for all applications of sensitivity analysis. First, with non-constant leaf prediction models, such as for linear trees, focusing only on variance reduction through splits ignores potential influence in the leaf model (for example, a covariate effect that is perfectly linear will lead to J_k near zero if fit with linear trees). Second, the importance metrics depend on the entire sample and cannot easily be focused on local input regions, say for optimization. Third, the importance indices provide a measure that is clearly interpretable in the context of tree models but does not correspond to any of the generic covariance decompositions in standard SA. In this section, we describe a technique for Monte Carlo estimation of these decompositions, referred to as *sensitivity indices* in the literature, that is model-free and can be constrained to subsets of the input space.

4.1 Sensitivity Indices

The classic paradigm of SA investigates analysis of response variability in terms of its conditional and marginal variance. This occurs in relation to a given *uncertainty distribution* on the inputs, labeled $U(\mathbf{x})$, and the integrals involved in variance decomposition are all measured with respect to this distribution. From the practitioner's perspective, U can represent uncertainty about future values of $\mathbf{x}$ or the relative amount of research interest in various areas of the input space (see Taddy et al., 2009). In application, U is commonly set as a uniform distribution over a bounded input region. Although it is possible to adapt the type of sampling described herein to account for correlated inputs in U (e.g., Saltelli and Tarantola, 2002), we treat only the standard and computationally convenient independent specification, $U(\mathbf{x}) = \prod_{k=1}^{p} u_k(x_k)$.

The sensitivity index for a set of covariates measures the variance, with respect to U, in conditional expectation given those variables. For example, the two most commonly reported indices concern *first order* and *total* sensitivity,

$$S_j = \frac{\mathbb{V}\mathrm{ar}\{\mathbb{E}\{y|x_j\}\}}{\mathbb{V}\mathrm{ar}\{y\}} \qquad \text{and} \qquad T_j = \frac{\mathbb{E}\{\mathbb{V}\mathrm{ar}\{y|\mathbf{x}_{-j}\}\}}{\mathbb{V}\mathrm{ar}\{y\}}, \qquad j = 1, \ldots, p, \tag{3}$$

respectively. The first order index represents response sensitivity to variable main effects and is closest in spirit to the importance metrics of Section 3. From the identity $\mathbb{E}\{\mathbb{V}\mathrm{ar}\{y|\mathbf{x}_{-j}\}\} = \mathbb{V}\mathrm{ar}\{y\} - \mathbb{V}\mathrm{ar}\{\mathbb{E}\{y|\mathbf{x}_{-j}\}\}$, T_j measures residual variance in conditional expectation and thus represents all influence connected to a given variable. Hence, $T_j - S_j$ measures the variability in y due to the interaction between input j and the other inputs, and a large difference can trigger additional local SA to determine its functional form. Note that all moments in (3) are with

respect to the uncertainty distribution U, but in our setting there is an additional layer of integration over the response uncertainty; for example, $\mathbb{E}\{y|x_j\} = \int_{\mathbb{R}^{p-1}} \mathbb{E}\{y(\mathbf{x})\} \prod_{k \neq j} u_k(x_k)\, d\mathbf{x}_{-j}$, where the inner expectation is the mean of the posterior predictive.

Our proposed approach is to use Monte Carlo integration, based on draws from the fitted response surface for a given tree, to estimate SA indices for each particle in a sample of DTs and thus provide an approximate posterior for any SA index of interest. The scheme is based on integral approximations presented in Saltelli (2002), with steps taken to account for an unknown response surface: the true response is replaced with statistically predicted values, and integration is repeated across a set of potential fitted trees. Although we focus on first-order and total sensitivity, indices for any covariate subset are available through an analogous adaptation of the appropriate routines in Saltelli (2002).

In the remainder of this section, calculations are presented for given individual tree particles; we suppress particle set indexing. Also, in our derivations we assume that the conditional response $y(\mathbf{x})$ has been integrated over the relevant posterior uncertainty. We begin to integrate the common $\mathbb{E}^2\{y\}$ terms by recognizing that

$$S_j = \frac{\mathbb{E}\{\mathbb{E}^2\{y|x_j\}\} - \mathbb{E}^2\{y\}}{\mathbb{V}\mathrm{ar}\{y\}} \qquad \text{and} \qquad T_j = 1 - \frac{\mathbb{E}\{\mathbb{E}^2\{y|x_{-j}\}\} - \mathbb{E}^2\{y\}}{\mathbb{V}\mathrm{ar}\{y\}}. \tag{4}$$

Assuming uncorrelated inputs, an approximation is facilitated by taking two equal-sized latin hypercube designs (LHDs) with respect to U. Specifically, we create LHDs M and M' of size m, assembled as matrices comprising p-length row-vectors $\mathbf{s}_k$ and $\mathbf{s}'_k$, for $k = 1, \ldots, m$, respectively. The unconditional quantities use M:

$$\widehat{\mathbb{E}\{y\}} = \frac{1}{m}\sum_{k=1}^{m} \mathbb{E}\{y|\mathbf{s}_k\} \quad \text{and} \quad \widehat{\mathbb{V}\mathrm{ar}\{y\}} = \frac{1}{m}\mathbb{E}\{y|M\}^\top \mathbb{E}\{y|M\} - \widehat{\mathbb{E}\{y\}}\widehat{\mathbb{E}\{y\}}, \tag{5}$$

where $\mathbb{E}\{y|M\}$ is the column vector $[\mathbb{E}\{y|\mathbf{s}_1\}, \cdots, \mathbb{E}\{y|\mathbf{s}_m\}]^\top$. From the former we can obtain $\widehat{\mathbb{E}^2\{y\}} = \widehat{\mathbb{E}\{y\}}\widehat{\mathbb{E}\{y\}}$.

Approximating the remaining components in (4) involves mixing columns of M' and M, which is where the independence assumption is crucial. Let M'_j be M' with the jth column replaced by the jth column of M, and likewise let M_j be M with the jth column of M'. The conditional second moments are then

$$\begin{aligned} \mathbb{E}\{\widehat{\mathbb{E}^2\{y|x_j\}}\} &= \tfrac{1}{m-1}\mathbb{E}\{y|M\}^\top \mathbb{E}\{y|M'_j\}, \\ \mathbb{E}\{\widehat{\mathbb{E}^2\{y|x_{-j}\}}\} &= \tfrac{1}{m-1}\mathbb{E}\{y|M'\}^\top \mathbb{E}\{y|M_j\} \approx \tfrac{1}{m-1}\mathbb{E}\{y|M\}^\top \mathbb{E}\{y|M'_j\}, \end{aligned} \tag{6}$$

the latter approximation saving us the effort of predicting at the locations in M_j.

In total, the set of input locations requiring evaluation under the predictive equations is the union of M, M', and $\{M'_j\}_{j=1}^p$. For LHDs of size m this is $m(p+2)$ locations for each of N particles. Together m and N determine the accuracy of the approximation. Usually N is fixed by other, more computationally expensive, sequential Monte Carlo particle updating considerations. The result of a particle-wide application of the above approximations is a sample from the (full) posterior distribution for $\mathbf{S}$ and $\mathbf{T}$.

4.2 Main Effect Visualization

A byproduct of the above procedure is information that can be used to estimate main effects. For each particle and input direction j, we apply a simple one-dimensional smoothing of the scatterplot of $[s_{1j}, \ldots, s_{mj}, s'_{1j}, \ldots, s'_{mj}]$ versus $[\mathbb{E}\{y|M\}, \mathbb{E}\{y|M'\}]$. This provides a realization of $\mathbb{E}\{y|x_j\}$ over a grid of x_j values, and therefore a draw from the posterior of the main effect curve. Average and quantile curves from each particle can be used to visualize the posterior uncertainty for the effect of each input direction as a function of its value. We find that the resulting curves are largely insensitive to the choice of smoother. We therefore prefer a simple moving average since the combined LHDs of size $2m$ represent a very large sample in one dimension.

4.3 Examples

Consider again the Friedman data from Section 3.3, using the first six inputs. Ordinarily we would recommend an initial VS procedure before undertaking SA to eliminate all irrelevant variables, but we keep one irrelevant input for illustrative purposes. Figure 3 summarizes the SA under constant and linear DTs (DTC and DTL, respectively), and under a GP (fit using `tgp`) for comparison. In all three cases the number of particles (or MCMC samples for the GP) and LHD sizes was the same: $N = 10000$ and $m = 1000$, without repeats. The main effects for DTL and GP are essentially identical. Apparently, DTC struggles to capture the marginal behavior of every input; x_3 is particularly off. These observations carry over to the $\mathbf{S}$ and $\mathbf{T}$ indices. DTL displays the same average values as the GP, but with greater uncertainty. DTC is again in less agreement and with greater uncertainty. It seems that whereas DTC works well for VS, DTL is better for SA.

With DTL and GP SA results so similar it is fair to ask why one should bother with DTL. The answer rests in the computational expense of the two procedures. The DT fit and SA stages both take a few minutes, separately. The GP version, even using a multithreaded version of `tgp`, takes about six hours on the same machine and requires that the two stages occur simultaneously. Hence, if new $\mathbf{x}$-y pairs are added or a new U specified, the entire analysis must be rerun from scratch. With DTs, the fit can be updated in a matter of seconds, and only the SA stage must be rerun, leading to even greater savings. In sum, the DT SA can give similar results to GPs but is hundreds of times faster.

GPs also are good at (but even slower) classification (GPC). Perhaps this is why we could not find GPC software providing SA output. Figure 4 shows the results of an SA for a 3-class/2d data set [see Gramacy and Polson (2011) for details and GPC references]. Fitting a GPC model from 200 $\mathbf{x}$-y pairs takes about an hour, for example, with the `plgp` package. By contrast, fitting a DT with multinomial leaves using $N = 10000$ particles takes a few seconds; and the SA postprocessing steps, which must proceed separately for each class, take a couple of minutes. The MAP class labels and predictive entropy shown on the left panel give an indication of the nature of the surface. Notice that the entropy is high near the misclassified points (red dots). The smooth transitions are difficult to capture with axis-aligned splits.

The plots in the right panels show the main effects and $\mathbf{S}$ and $\mathbf{T}$ indices for each class. All three sets of plots indicate a dominant x_1 influence, which conforms to intuition since that axis spans three labels whereas x_2 spans only two. Lower S and T values for x_2 provide further

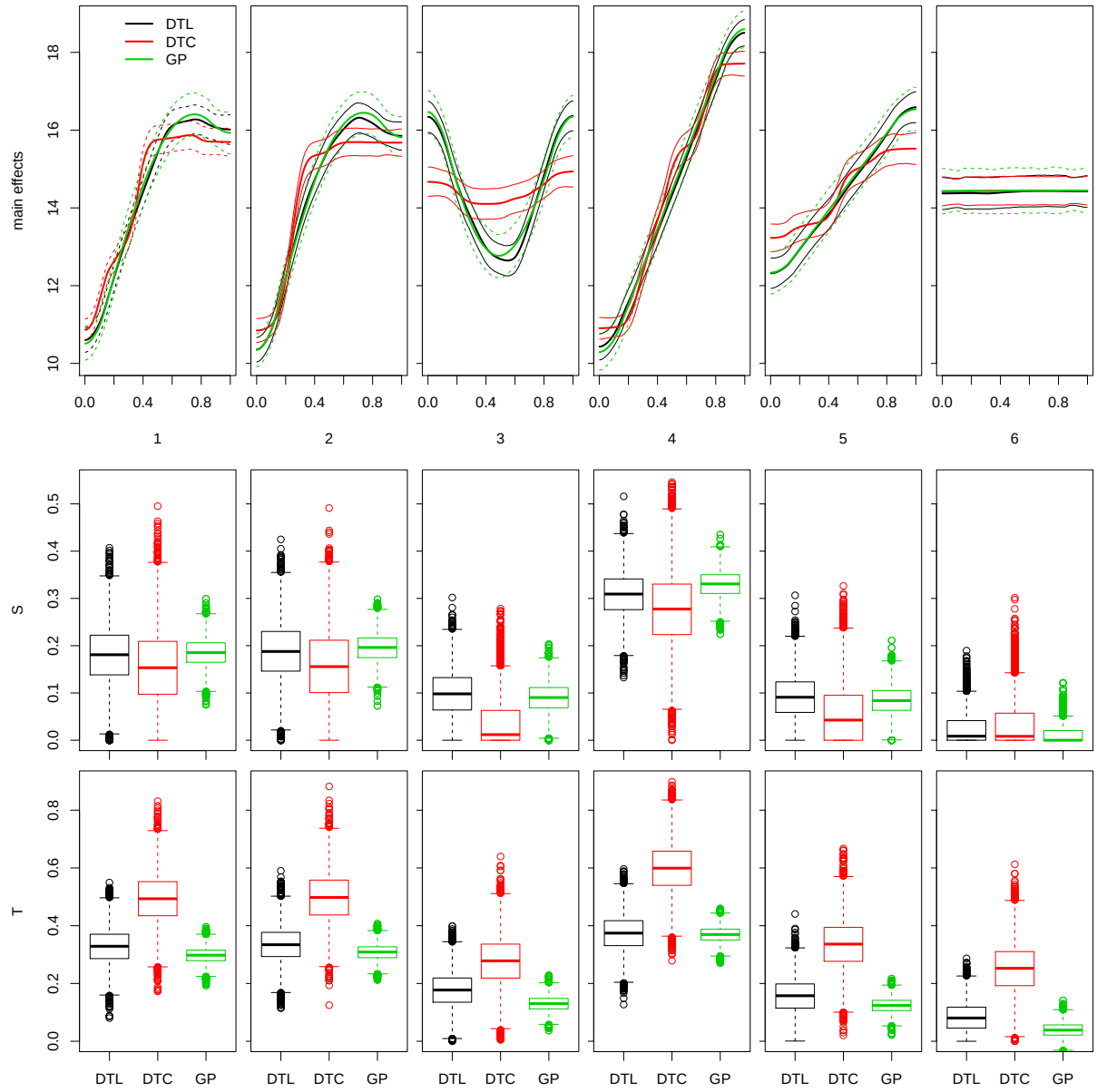

Figure 3: Main effects *(first row)*, $\mathbf{S}$ *(second row)* and $\mathbf{T}$ *(third row)* indices for the Friedman data using dynamic trees and GPs.

evidence that its contribution to the variance is partly coupled with that of x_1.

5 A Computer Experiment: Optimizing Linear Algebra Kernels

We now examine the data generated by linear algebra kernels from (Balaprakash et al., 2011a), focusing on the GESUMMV experiment. The results obtained for the other two kernels (MATMUL and TENSOR) we examined are similar, and are therefore omitted because of space constraints. We first examine the extent of the cold cache effect with VS techniques and ulti-

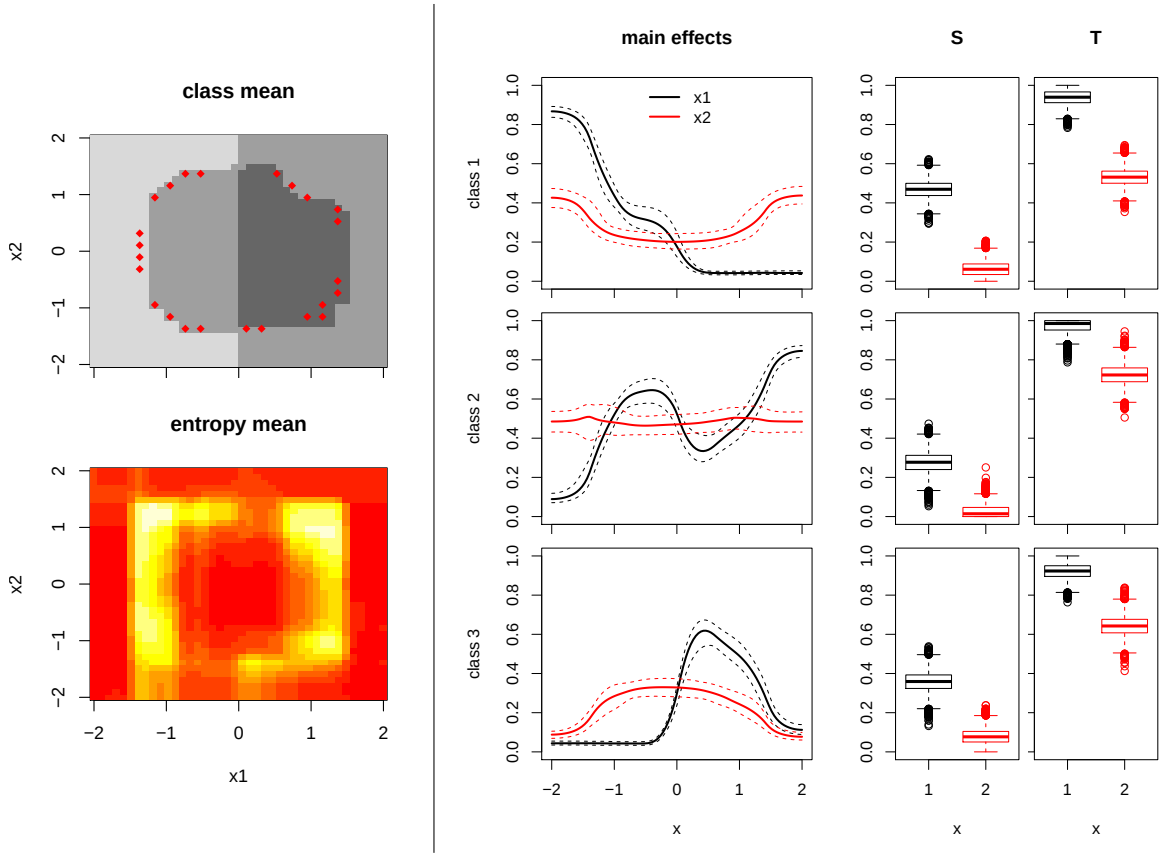

Figure 4: *(Left)* Posterior predictive mean and entropy; misclassified points are shown as red dots. *(Right panels)* SA main effects and $\mathbf{S}$ and $\mathbf{T}$ indices for each class.

mately use evidence from BFs to conclude that, while clearly present, acknowledging it in the model does not necessarily lead to a better fit. Then we turn to a full analysis of the sensitivity to inputs and we explore the extent to which one can learn about, and avoid, constraint violations.

As described in (Balaprakash et al., 2011a), the data for these experiments was obtained on Fusion, a 320-node cluster at Argonne National Laboratory, each compute node consisting of 2.6 GHz Pentium Xeon 8-core processors with 36 GB of RAM. Although the size of the resulting data are well out of reach of standard approaches based on GPs, these problems are considered to be relatively small [several orders of magnitude larger problems are considered in (Balaprakash et al., 2011b)] from a performance tuning perspective. Hence, a complete enumeration of the design space was performed. In order to address the stochasticity of the collected runtimes due to operating system noise, 35 trials of each design point were run in succession on the same dedicated node.

The GESUMMV kernel from the updated BLAS library (Blackford et al., 2002) carries out a sum of dense matrix-vector multiplies. The design variables consist of two loop-unrolling parameters taking integer values in $\{1, \cdots, 30\}$ and three binary parameters associated with performing scalar replacement, loop parallelization, and loop vectorization, respectively. Of the $2^3 30^2 = 7,200$ total design points, 199 were found to result in a compilation error or an improper memory access and thus were deemed to violate a constraint on correctness. The

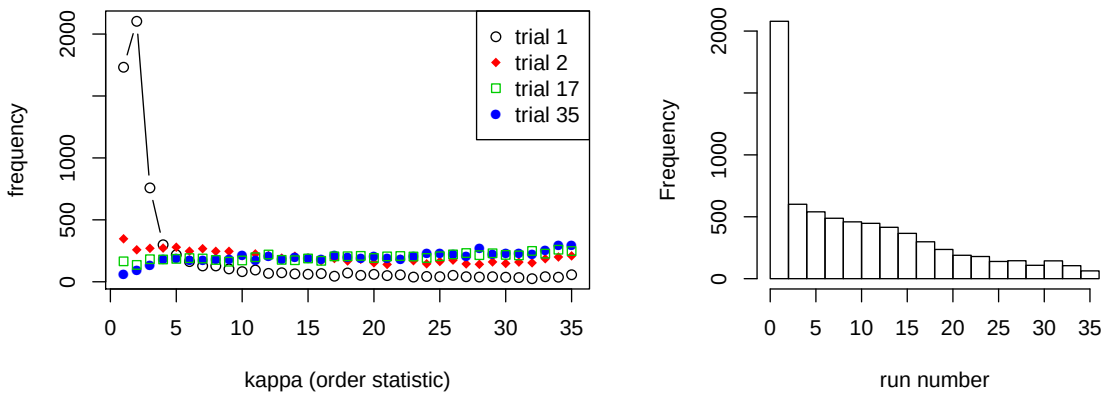

Figure 5: *(Left)* Histograms for 4 particular trials with respect to the order statistics on decreasing runtimes; *(Right)* Frequency of trial number that yielded the maximum of the 35 runtimes.

resulting 245,035 (successful) runtimes were between 0.15 and 0.68 seconds, the mean and median both being 0.22 seconds.

5.1 Cold Cache Effect and Variable Selection

Figure 5 shows an illustration of the cold cache effect. The left plot shows four "histograms" counting the number (out of the 7,001 input locations that did not result in a constraint violation) of times the first, second, 17th, and last of the trials resulted in the κ^{th} largest runtime of the 35 trials performed. While the first trial stands out as the slowest, results for the three other trials indicate that this effect does not persist for later trials. That is, the second (17th or 35th) does not tend to have the second (17th- or 35th-) largest runtime. Viewing things another way tells a slightly different story, however. The right histogram in the figure shows that lower trial numbers tend to yield the maximum execution time more frequently than higher ones.

Interest is in exploring these cursory observations statistically to help decide whether the first trial requires, or all trials require, special handling in order to obtain more reliable predictions, or to accurately tease out the effects of the inputs on the variability of execution times. Since the absolute average distance from the max to the median (among 35 trials for each of the 7,001 input configurations) is about 0.01, compared with the full difference between the maximum and minimum execution in the entire data set at 0.53, it is not immediately clear that accounting for the effect is important.

Consider the following experiment on a subset of the data comprising the first trial and the last trial for every input configuration — 14,002 runs in total. The five inputs were augmented with a sixth comprising of an indicator which is zero for those from the first trial and one for those from the last. Our thinking is that if the cold cache effect is statistically significant, then this experiment would reveal so. Figure 6 summarizes the results of the experiment, which used the constant leaf model, 1000 particles, and 30 repeats. The left panel shows the posterior relevance samples; and the focus, for now, is on the relevance of the sixth input, which is small. The posterior probability that the relevance is greater than zero is 0.83; however, the mean relevance is 3.3×10^{-8}, so the scale of the y-axis (tailored to the other, more relevant inputs) is somewhat unfair visually. Thus, the relevance statistic detects a positive relevance for the

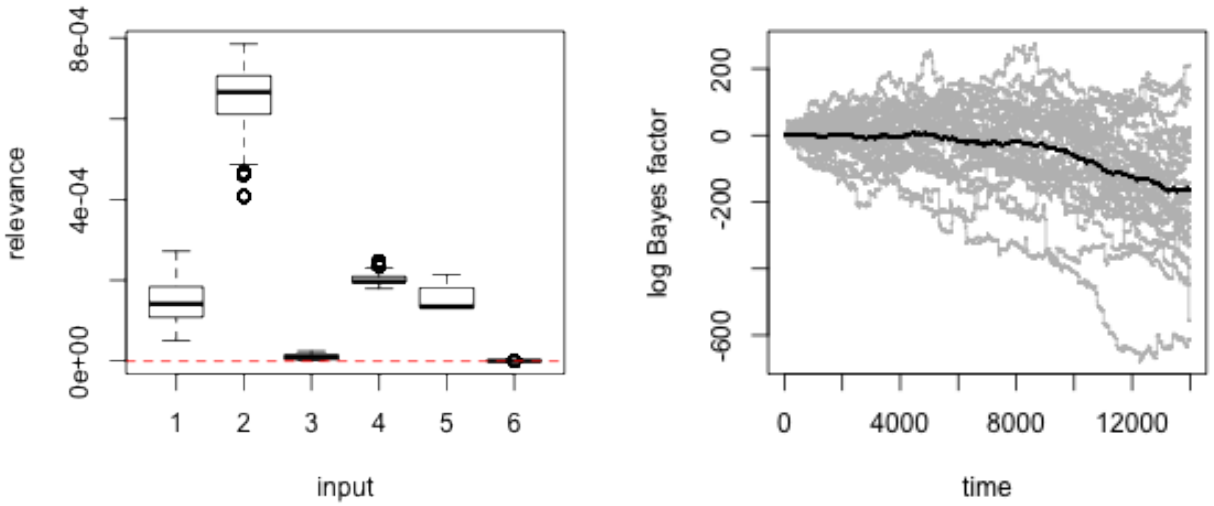

Figure 6: *(Left)* Relevance statistics for the five inputs, plus the cold cache indicator (sixth input). *(Right)* BF comparing the model with the sixth input to the one without.

cold cache effect, which is reassuring given that we know it exists (Figure 5). However, it also indicates that it may not be statistically relevant (a 17% chance), and therefore not useful for prediction. For a second opinion we consult the BF. The right panel of the figure shows a decreasing trend, which indicates that the extra predictor is not helpful.

Before turning to SA (ignoring the cold cache effect) we observe that input three also shows low relevance. In contrast to the cold cache indicator, however, all the posterior relevance samples were positive. Still, a similar BF calculation (not shown) indicated that it too could be dropped from the model. The remaining four inputs have much greater relevance; BF calculations (also not shown) reinforce that these predictors are important to obtain a good fit.

5.2 Sensitivity Analysis

To further inform an optimization of the automatic code tuning process, we perform an SA. Figure 7 summarizes main effects and $\mathbf{S}$ and $\mathbf{T}$ indices for the four remaining variables. The full data (all 35 trials) were used — 245,035 input-output pairs total, ignoring the cold cache effect. Results for both constant and linear leaf models are shown. In contrast to our illustrative results for the Friedman data, the differences between linear and constant leaves are negligible. Since both treat the binary predictors in the same way, this result is perhaps not surprising.

The $\mathbf{S}$ and $\mathbf{T}$ indices on the remaining predictors tell a similar, but richer, story compared with the relevance statistics. Input two has the largest effect, and input one the smallest, but we also see that the effect of the inputs, marginally, is small (since the Ss are low and the Ts are high). This result would lead us to doubt that a rule of thumb for optimizing the codes based on the main effects alone would bear fruit, namely, that inputs close to $\langle 5, 12, 0, 0\rangle$ are most promising. Although this may be a sensible place to start, intricate interactions among the variables, as suggested by the $\mathbf{T}$ indices in particular, mean that a search for optimal tuning parameters would benefit from a methodical iterative approach, say with an expected improvement criterion (Jones et al., 1998) or other optimization routine. This is beyond the scope of this

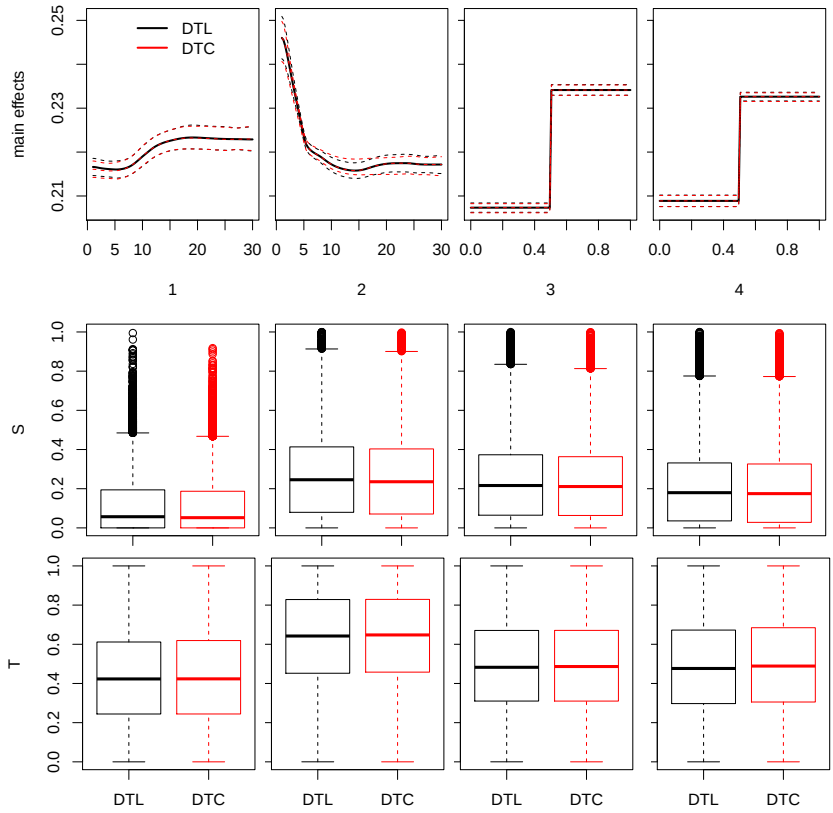

Figure 7: Main effects *(first row)*, $\mathbf{S}$ *(second row)*, and $\mathbf{T}$ *(third row)* indices for GESUMMV.

paper; however, a rule of thumb is certainly attainable via a more localized sensitivity analysis.

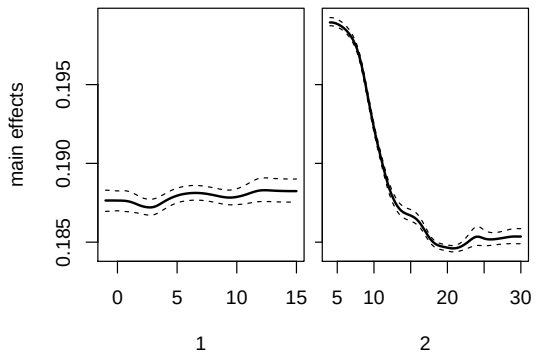

Figure 8: Close-up of (constrained) main effects. The $\mathbf{S}$ and $\mathbf{T}$ indices are similar to those in Figure 7.

Figure 8 shows the main effects from a SA (using DTC) whose uncertainty distribution $U(\mathbf{x})$ is constrained so that the first input is ≤ 15, the second is ≥ 5, and the third and fourth are fixed to zero but otherwise unchanged. Note that only $U(\mathbf{x})$ is restricted, not the actual input-output pairs. The full data is used, and the model-fitting does not need to be rerun. In contrast to the conclusion drawn from Figure 7, the function would actually appear to be minimized near $x_2 = 20$ rather than 12. Input x_1, as long as it is restricted to be less than

15, does not seem to matter. A broad-stroke characterization like this may be more valuable than an expensive iterative search, or, at the very least, yield a considerably smaller space for a subsequent search.

5.3 Checking for Constraint Violation Patterns

The original 7,200 design points, with (two) classification labels indicating `NA` values or positive real numbers (times), were used to fit a DT model with multinomial leaves. Otherwise the setup was similar to our earlier examples. Note that if one of the 35 trials was `NA`, then they all were. The resulting posterior distribution of the importance indices were decidedly null. The posterior probability of being relevant (i.e., of having a positive index) was $0.003, 0.026, 0.000, 0.000$, and 0.000 for the five inputs, respectively. We interpret this as meaning that the DT model detects no spatial pattern in the 2.7% of code failures compared with the successful runs. This conclusion was backed up by a simple BF calculation where the null model disallowed any partitioning. These results are reassuring because the input space was designed to limit the number of correctness violations; if there were known a priori relationships between the inputs and these violations, the design space would be adjusted accordingly to prevent failures at compile or run time.

6 Discussion

The advent of fast and cheap computers defined a statistical era in the late 20^{th} century, especially for Bayesian inference. Modestly sized data sets, enormous computing power, and clever algorithms allowed the entertainment of extremely flexible and nonparametric models. For computer experiments and other spatial data, GP models typify the state of the art from that era, with many successful applications. In classification problems, latent variables were key to exploiting computation for modeling flexibility. Today, further technological advance is defining a new era, that of massive data generation and collection where computer and physical observables are gathered at breakneck pace.

These huge datasets are testing the limits of the popular models and implementations. GPs are buckling under the weight of enormous matrix inverses, and latent variable models suffer from mixing (MCMC) problems. Thrifty nonparametric models are sorely needed. Recently popular remedies include partition trees, a throwback to the 1980s and earlier, before the MCMC revolution in statistics. This paper showed how the dynamic tree model can perform many of the same functions as GP and latent variable models at a fraction of the computational cost. By borrowing relevance statistics from classical trees and sensitivity indices from GPs, the end product is an exploratory data analysis tool that can facilitate variable selection, dimension reduction, and visualization. An open source implementation is provided in a recent update of the `dynaTree` package for R.

Our illustrations included instructive datasets from the recent literature and a new computer experiment on automatic code generation that is likely to be a hot application area for statistics and other disciplines as heterogeneous computing environments become more commonplace. Ultimately, the goal is to optimize code for the architecture "just in time", when it arrives at the computing node. In order to be realistically achievable, that goal will require both rules of

thumb, as facilitated by VS and SA procedures like the ones outlined in this paper, and iterative optimization steps that will surely feature in future work.

Acknowledgments

We are grateful to Prasanna Balaprakash for providing the data from (Balaprakash et al., 2011a). The work of Wild was supported by the Office of Advanced Scientific Computing Research, Office of Science, U.S. Department of Energy, under Contract DE-AC02-06CH11357.

Appendix

Here, we derive the variance integrals from (2) for a model with linear leaves. Dropping the node subscript (η, η_ℓ, or η_r), we have

$$\begin{aligned}\int_A \sigma^2(\mathbf{x})\,d\mathbf{x} &= \int_A \frac{s^2-\mathcal{R}}{n-p-1}\left(1+\frac{1}{n}+\mathbf{x}^\top\mathcal{G}^{-1}\mathbf{x}\right)d\mathbf{x} \\ &= \frac{s^2-\mathcal{R}}{n-p-1}\left(|A|\left(1+\frac{1}{n}\right)+\int_A \mathbf{x}^\top\mathcal{G}^{-1}\mathbf{x}\,d\mathbf{x}\right),\end{aligned} \tag{7}$$

where s^2 is the sum of squares, $\mathcal{R}$ is the regression sum of squares, $n \equiv |\eta|$ is the number of $(\mathbf{x}, y)$ pairs, $\mathcal{G}$ is the Gram matrix, and $|A|$ is the area of the rectangle. The remaining integral is just a sum of polynomials: with the intervals outlining the rectangle given by $(a_1, b_1), \ldots, (a_p, b_p)$ and (g_{ij}) the components of $\mathcal{G}^{-1}$,

$$\begin{aligned}\int_A \mathbf{x}^\top\mathcal{G}^{-1}\mathbf{x}\,d\mathbf{x} &= \int_{a_1}^{b_1}\cdots\int_{a_p}^{b_p}\sum_{i=1}^{p}\sum_{j=1}^{p} x_i x_j g_{ij}\,dx_i \\ &= \sum_{i=1}^{p}\frac{g_{ii}}{3}(b_i^3-a_i^3)\prod_{k\neq i}(b_k-a_k) + 2\sum_{i=1}^{p}\sum_{j>i}\frac{g_{ij}}{4}(b_i^2-a_i^2)(b_j^2-a_j^2)\prod_{k\neq i,j}(b_k-a_k) \\ &= |A|\left(\sum_{i=1}^{p}\frac{g_{ii}(b_i^3-a_i^3)}{3(b_i-a_i)} + \sum_{i=1}^{p}\sum_{j>i}\frac{g_{ij}(b_i+a_i)(b_j+a_j)}{2}\right).\end{aligned} \tag{8}$$

References

Asuncion, A. and Newman, D. (2007). "UCI Machine Learning Repository." `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

Balaprakash, P., Wild, S. M., and Hovland, P. D. (2011a). "Can Search Algorithms Save Large-scale Automatic Performance Tuning?" *Procedia Computer Science*, 4, 2136–2145. Proceedings of the International Conference on Computational Science, ICCS 2011.

Balaprakash, P., Wild, S. M., and Norris, B. (2011b). "SPAPT: Search Problems in Automatic Performance Tuning." Preprint ANL/MCS-P1872-0411, Mathematics and Computer Science Division, Argonne National Laboratory.

Bayarri, M. J., Berger, J. O., Kennedy, M. C., Kottas, A., Paulo, R., Sacks, J., Cafeo, J. A., Lin, C.-H., and Tu, J. (2009). "Predicting Vehicle Crashworthiness: Validation of Computer Models for Functional and Hierarchical Data." *Journal of the American Statistical Association*, 104, 929–943.

Blackford, L. S., Demmel, J., Dongarra, J., Duff, I., Hammarling, S., Henry, G., Heroux, M., Kaufman, L., Lumsdaine, A., Petitet, A., Pozo, R., Remington, K., and Whaley, R. C. (2002). "An updated set of basic linear algebra subprograms (BLAS)." *ACM Trans. Math. Softw.*, 28, 135–151.

Breiman, L. (1995). "Better Subset Regression Using the Nonnegative Garrote." *Technometrics*, 51, 373–384.

Breiman, L., Friedman, J. H., Olshen, R., and Stone, C. (1984). *Classification and Regression Trees*. Belmont, CA: Wadsworth.

Cantoni, E., Flemming, J., and E., R. (2011). "Variable Selection in Additive Models by Nonnegative Garrote." *Statistical Modelling*, *to appear*.

Carvalho, C. M., Johannes, M., Lopes, H. F., and Polson, N. G. (2010). "Particle Learning and Smoothing." *Statistical Science*, 25, 88–106.

Chipman, H., George, E., and McCulloch, R. (1998). "Bayesian CART Model Search (with discussion)." *Journal of the American Statistical Association*, 93, 935–960.

— (2002). "Bayesian Treed Models." *Machine Learning*, 48, 303–324.

Chipman, H. A., George, E. I., and McCulloch, R. E. (2010). "BART: Bayesian Additive Regression Trees." *The Annals of Applied Statistics*, 4, 266–298.

Farah, M. and Kottas, A. (2011). "Bayesian Inference for Sensitivity Analysis of Computer Simulators, with an Application to Radiative Transfer Models." Tech. Rep. UCSC-SOE-10-15, UC Santa Cruz.

Friedman, J. H. (1991). "Multivariate Adaptive Regression Splines." *Annals of Statistics*, 19, No. 1, 1–67.

Friedman, J. H., Hastie, T., and Tibshirani, R. (2010). "Regularization Paths for Generalized Linear Models via Coordinate Descent." *Journal of Statistical Software*, 33, 1, 1–22.

Gramacy, R. and Polson, N. (2010). "Simulation-based regularized logistic regression." Tech. Rep. arXiv:1005.3430, University of Chicago.

— (2011). "Particle Learning of Gaussian Process Models for Sequential Design and Optimization." *Journal of Computational and Graphical Statistics*, 20, 1, 102–118.

Gramacy, R. B. and Taddy, M. A. (2010). "Categorical Inputs, Sensitivity Analysis, Optimization and Importance Tempering with `tgp` Version 2, an R Package for Treed Gaussian Process Models." *Journal of Statistical Software*, 33, 6, 1–48.

— (2011). `dynaTree`*: Dynamic Trees for Learning and Design*. R package version 2.0.

Hartono, A., Norris, B., and Sadayappan, P. (2009). "Annotation-based Empirical Performance Tuning Using Orio." In *IEEE Symposium on Parallel Distributed Processing*, 1 –11.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. 2nd ed. Springer.

Huang, J., Horowitz, J., and Wei, F. (2010). "Variable Selection in Nonparametric Additive Models." *Annals of Statistics*, 38, 2282–2313.

Jones, D., Schonlau, M., and Welch, W. J. (1998). "Efficient Global Optimization of Expensive Black Box Functions." *Journal of Global Optimization*, 13, 455–492.

Krishnapuram, B., Carin, L., Figueiredo, M., and Hartemink, A. (2005). "Sparse Multinomial Logistic Regression: Fast Algorithms and Generalization Bounds." *IEEE Pattern Analysis and Machine Intellegence*, 27, 6, 957–969.

Lin, Y. and Zhang, H. (2006). "Component Selection and Smoothing in Smoothing Spline Analysis of Variance Models." *Annals of Statistics*, 34, 2272–2297.

Linkletter, C., Bingham, D., Hengartner, N., Higdon, D., and Ye, K. Q. (2006). "Variable Selection for Gaussian Process Models in Computer Experiments." *Technometrics*, 48, 478–490.

Marrel, A., Iooss, B., Laurent, B., and Roustant, O. (2009). "Calculations of Sobol Indices for the Gaussian Process Metamodel." *Reliability Engineering and System Safety*, 94, 742–751.

Morgan, J. N. and Sonquist, J. A. (1963). "Problems in the Analysis of Survey Data, and a Proposal." *Journal of the American Statistical Association*, 58, 415–434.

Morris, R. D., Kottas, A., Taddy, M., Furfaro, R., and Ganapol, B. (2008). "A Statistical Framework for the Sensitivity Analysis of Radiative Transfer Models." *IEEE Transactions on Geoscience and Remote Sensing*, 12, 4062–4074.

Oakley, J. and O'Hagan, A. (2004). "Probabilistic Sensitivity of Complex Models: a Bayesian Approach." *Journal of the Royal Statistical Society, Series B*, 66, 751–769.

Patterson, D. A. and Hennessy, J. L. (2007). *Computer Organization and Design - the Hardware / Software Interface*. 3rd ed. Morgan Kaufmann.

Reich, B., Storlie, C., and Bondell, H. (2009). "Variable Selection in Bayesian Smoothing Spline ANOVA Models: Application to Deterministic Computer Codes." *Technometrics*, 51, 2, 110–120.

Saltelli, A. (2002). "Making Best Use of Model Evaluations to Compute Sensitivity Indices." *Computer Physics Communications*, 145, 280–297.

Saltelli, A., Chan, K., and Scott, E., eds. (2000). *Sensitivity Analysis*. John Wiley and Sons.

Saltelli, A., Ratto, M., Andres, T., Campolongo, F., Cariboni, J., Gatelli, D., Saisana, M., and Tarantola, S. (2008). *Global Sensitivity Analysis: The Primer*. John Wiley & Sons.

Saltelli, A. and Tarantola, S. (2002). "On the Relative Importance of Input Factors in Mathematical Models: Safety Assessment for Nuclear Waste Disposal." *Journal of the American Statistical Association*, 97, 702–709.

Sansó, B., Lee, H. K. H., Zhou, W., and Higdon, D. (2008). "Inference for a Proton Accelerator Using Convolution Models." *Journal of the American Statistical Association*, 103, 604–613.

Santner, T. J., Williams, B. J., and Notz, W. I. (2003). *The Design and Analysis of Computer Experiments*. New York, NY: Springer-Verlag.

Storlie, C. B., Swiler, L. P., Helton, J. C., and Sallaberry, C. J. (2009). "Implementation and Evaluation of Nonparametric Regression Procedures for Sensitivity Analysis of Computationally Demanding Models." *Reliability Engineering & System Safety*, 94, 1735–1763.

Taddy, M., Gramacy, R., and Polson, N. (2011). "Dynamic Trees for Learning and Design." *Journal of the American Statistical Association*, 106, 493, 109–123.

Taddy, M. A., Lee, H. K. H., Gray, G. A., and Griffin, J. D. (2009). "Bayesian Guided Pattern Search for Robust Local Optimization." *Technometrics*, 51, 389–401.

Vuduc, R., Demmel, J. W., and Bilmes, J. A. (2004). "Statistical Models for Empirical Search-Based Performance Tuning." *International Journal of High Performance Computing Applications*, 18, 1, 65–94.

Ziehn, T. and Tomlin, A. (2009). "`GUI-HDMR` - a Software Tool for Global Sensitivity Analysis of Complex Models." *Environmental Modelling and Software*, 24, 775–785.