Boosted Varying-Coefficient Regression Models for Product Demand Prediction

November 14, 2011

Jianqiang C. Wang\textsuperscript{1} Trevor Hastie\textsuperscript{2}

Abstract

Estimating the aggregated market demand for a product in a dynamic market is intrinsically important to manufacturers and retailers. Motivated by the need for a statistical demand prediction model to facilitate laptop pricing at Hewlett-Packard, we have developed a novel boosting-based varying-coefficient regression model. The developed model uses regression trees as the base learner. Our method is generally applicable to varying-coefficient models with a large number of mixed-type varying-coefficient variables, which proves to be challenging for conventional nonparametric smoothing methods. The propose approach works well in both predicting the response and estimating the varying-coefficient functions, based on a simulation study. Finally, we have applied this methodology to real-world mobile computer sales data for product demand prediction.

KEY WORDS: Boosting; gradient descent; tree-based regression; varying-coefficient model.

\textsuperscript{1}Hewlett-Packard Labs, Palo Alto, CA, 94304.
\textsuperscript{2}Department of Statistics, and Health, Research & Policy, Stanford University, CA, 94305.
1 Introduction

Product pricing in a dynamic market with competition is intrinsically important to manufacturers and retailers. The conventional practice of using business expertise to make decisions is subjective, irreproducible and difficult to scale up to a large number of products. The use of advanced analytic tools, including statistical analysis, operations research, and microeconomics, provides a scientifically sound approach to accurately price a large number of products while offering a reproducible and real-time solution. The flow chart in Figure 1 shows the process of an ongoing pricing optimization project at Hewlett-Packard, where the objective is to price the mobile computers the company offers. In this project, we obtain historical product sales data from a third-party marketing firm, estimate the aggregated market demand for each product, and seek to optimize certain business criterion (e.g., profit) under various constraints (e.g., constraints on market share, component availability, inventory and so on). A key input to the *pricing and optimization engine* is a demand prediction model that quantifies the demand under different price points for each product. This motivates the study of demand prediction models in this paper.

There is a huge body of literature on demand modeling in econometrics, and many working models can be broadly classified into two categories: models that target the demand units and models that target the consumer choices (choice-based models). We focus on models for the number of units in this paper, and will pursue choice models in the future. The market sales data we obtain contains information on the sales units, average selling price, product features and environmental variables like the time, geographic location and sales channel. Ideally, we would like to have a demand model for an arbitrary product in any period, location and offered from any channel. But the scarcity of data points prevents us from fitting individual models. Further, the concern about overfitting rises if such models are indeed built. So we need to link the individual models in some way to have sufficient observations for estimating the demand with precision. Conceptually, we would like to use relatively simple model structure for demand as functions of price (e.g., linear regression), but
allow the model parameters to vary with product features and environmental variables. This leads to our study of varying-coefficient models. The varying-coefficient models are readily interpretable and easily comprehended by non-statisticians. Specifically for this context, we can provide a standard interface between the statistical model and the downstream pricing optimization by reporting the fitted individual regression parameters.

The varying-coefficient regression model, initially introduced by Cleveland et al. (1991) and Hastie and Tibshirani (1993), is gaining its popularity in statistics literature in recent years. In practice, the varying-coefficient models often have solid scientific motivation and yet yield superior fits to the empirical data by allowing the parameters to vary as functions of some environmental variables. Very often in varying-coefficient models, the coefficients have unknown functional form, which is estimated nonparametrically. A seminal review of the varying-coefficient models given by Fan and Zhang (2008) discussed three estimation methods, including kernel smoothing, polynomial splines and smoothing splines. In the same paper, the authors talked about the generalized varying-coefficient models in which
the distribution of the response variable belongs to an exponential family. The authors have also discussed the applications of the varying-coefficients models in various contexts, including longitudinal and functional data, survival analysis, nonlinear time series and so on.

In prediction problems like the one we have, there is a large number of varying-coefficient variables with mixed types, including categorical, ordinal and continuous. Specifically, in predicting product demand, the varying-coefficient variables include various product features and environmental variables like the time and location. The regression coefficients are thus functions of high-dimensional variates, which need to be estimated based on data. Here, the interaction among product features is complex. It is unrealistic to assume that their effects are additive, and it is difficult to specify a functional form that characterizes their joint effects on the regression parameters. Given these practical constraints, we desire to have a data-driven approach for estimating the high-dimensional non-additive coefficient functions. The tree-based approach, initially proposed by Breiman et al. (1984) under the name of classification and regression trees (CART), has proved to be a successful learning method in a wide variety of problems, including but not limited to high-dimensional classification and regression. The tree-based methods handle the high-dimensional prediction problems in a scalable way, naturally incorporate complex interactions and are favored by practitioners for their interpretability. Unfortunately, the single-tree based learning methods are unstable, and a small perturbation to the data may lead to a dramatically changed model. An ensemble method called boosting has been introduced that greatly improves the predictive performance of tree models by combining multiple trees. Both the single-tree based model and boosting naturally handle variable interactions and automatically select the important variables. Thus in this paper, we study the tree-based varying-coefficient models and the boosted version, as an alternative to conventional nonparametric varying-coefficient models estimated by kernel smoothing or splines.

The remainder of the paper proceeds as follows. Section 3 introduces the tree-based
varying-coefficient model and presents details on the greedy algorithm for binary tree partitioning. Section 4 formulates the boosting tree-based varying-coefficient model and talks about graphical representations of the boosting model. Section 5 presents numerical results comparing the two proposed procedures based on simulated data. Section 6 applies both the tree-based and boosting models to the mobile computer sales data obtained from a third-party marketing firm. Finally, some remarks are provided in section 7 that generalizes the scope of our proposed methods.

2 Semiparametric Varying-coefficient Linear Regression

We introduce the boosting tree-based varying-coefficient regression model, as an alternative to kernel smoothing and spline-based methods, for estimating the varying-coefficient surfaces. The boosting approach takes the tree regression as a base learner, in which we approximate the varying-coefficient functions by piecewise constant functions. We introduce the generic version of the semiparametric varying-coefficient regression in this section, and describe the tree and boosting methods in sections 3 and 4.

Let $y$ be the response variable, $\mathbf{x} \in \mathbb{R}^p$ denote the vector of predictors that a parametric relationship is available between $y$ and $\mathbf{x}$, for any given values of modifying predictor vector $\mathbf{s} \in \mathbb{R}^q$. The regression relationship between $y$ and $\mathbf{x}$ varies under different values of $\mathbf{s}$. Let $(\mathbf{s}_i', \mathbf{x}_i', y_i)$ denote the measurements on subject $i$, where $i = 1, \ldots, n$. Here, the varying-coefficient variable is $\mathbf{s}_i = (s_{i1}, s_{i2}, \ldots, s_{iq})'$ and the regression variable is $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{ip})'$, and we allow overlap between the two sets of variables. We work here with the following varying-coefficient linear regression model, and will discuss generalizations to nonlinear models or generalized linear models in section 7. The varying-coefficient linear model specifies that,

$$y_i = f(\mathbf{x}_i, \mathbf{s}_i) + \epsilon_i = \mathbf{x}_i'\beta(\mathbf{s}_i) + \epsilon_i,$$  

(1)
where the regression coefficients $\beta(s_i)$ are modeled as functions of $s$.

In model (1), our key interest is to estimate the multivariate coefficient surface $\beta(s_i)$. Fan and Zhang (2008) gave an excellent review of the varying-coefficient models and discussed three approaches in estimating the coefficient function $\beta(s_i)$, including kernel smoothing, polynomial splines and smoothing splines. We provide a novel estimation method here, which is readily scalable for a high-dimensional varying-coefficient vector $s_i$ with interactions not specified a priori. In the exposition of the proposed method, we will explain the base learner, which is a modified regression tree, in section 3, and then describe the boosting method in section 4.

3 Tree-based Varying-coefficient Regression

In this paper, we offer two views of the tree-based varying-coefficient regression: as a stand-alone model or as a base learner for boosting. When the tree method is used as a stand-alone approach, we need to tune the number of nodes to achieve optimal performance. When boosting the tree, we can fix the number of nodes at a small value, but tune the number of boosting iterations instead. We do not seek to draw distinctions between these two perspectives, but attempt to make our presentation as general as possible.

The tree-based method approximates $\beta(s_i)$ in (1) by a piecewise constant function. More specifically, the tree algorithm seeks to partition the space of $s$ according to certain information criterion and then approximate $\beta(s_i)$ in each partition by a constant vector. Here, we refer to the varying-coefficient variables as *partition variables* as we build partitions based on these variables, and the predictors in $x$ as *regression variables*. The idea of partitioning the space of varying coefficient variables $s$, and then imposing a parametric form familiar to the subject matter area within each partition conforms with the general notion of conditioning on the varying-coefficient variables.

Let $\{C_m\}_{m=1}^M$ denote a partition of the space $\mathbb{R}^q$ satisfying $C_m \cap C_{m'} = \emptyset$ for any $m \neq m'$, and $\bigcup_{m=1}^M C_m = \mathbb{R}^q$. Here, $M$ denotes the number of partitions. The proposed tree-based
varying-coefficient model is

\[ y_i = \sum_{m=1}^{M} x'_i \beta_m I(s_i \in C_m) + \epsilon_i, \]  

(2)

where \( I(\cdot) \) denotes the indicator function with \( I(c) = 1 \) if event \( c \) is true and zero otherwise. The error terms \( \epsilon_i \)'s are assumed to have zero mean and homogeneous variance \( \sigma^2 \). The implied varying-coefficient function is thus,

\[ \beta(s_i) \approx \sum_{m=1}^{M} \beta_m I(s_i \in C_m), \]

a piecewise constant function in \( \mathbb{R}^q \). In the terminology of recursive partitioning, the set \( C_m \) is referred to as a terminal node or leaf node, which defines the ultimate grouping of the observations. The proposed partitioned regression model (2) can be treated as an extension of regression trees which reduces to the ordinary regression tree if the vector of regression variables only includes the constant. Variations of our proposed partitioned regression model have been studied in literature, which were termed as piecewise linear model in Loh (1997), and as treed regression in Chipman et al. (2002), to name a few.

The number of terminal nodes \( M \) is unknown, as well as the partitions \( \{C_m\}_{m=1}^{M} \). In its fullest generality, the estimation of model (2) requires the estimation of \( M, C_m \) and \( \beta_m \) simultaneously. The number of components \( M \) is difficult to estimate and, in our case, could either be tuned via out-of-sample goodness-of-fit criteria or automatically determined by imposing certain stopping rules. When the trees are used as base learners in boosting, the standard practice is to fix the number of tree nodes and tune the model via the number of boosting iterations. So the determination of \( M \) is trivial in such cases.

We postpone the discussion of the determination of \( M \) till later, and focus on the estimation of partition and regression coefficients for the moment. The least squares criterion for (2) leads to the following estimator of \( (C_m, \beta_m) \), as minimizers of sum of squared errors (SSE),

\[ (\hat{C}_m, \hat{\beta}_m) = \arg \min_{(C_m, \beta_m)} \sum_{i=1}^{n} \left( y_i - \sum_{m=1}^{M} x'_i \beta_m I(s_i \in C_m) \right)^2 = \arg \min_{(C_m, \beta_m)} \sum_{i=1}^{n} \sum_{m=1}^{M} (y_i - x'_i \beta_m)^2 I(s_i \in C_m), \]

(3)
In the above, the estimation of $\beta_m$ is nested in that of the partitions. We define $\hat{\beta}_m(C_m)$ as a consistent estimator of $\beta_m$ given the partitions. The estimator could be a least squares estimator, maximum likelihood estimator, or an estimator defined by estimating equations. Specifically, we take the following least squares estimator as an example,

$$\hat{\beta}_m(C_m) = \arg \min_{\beta_m} \sum_{i=1}^{n} (y_i - x_i' \beta_m)^2 I(s_i \in C_m),$$

in which the minimization criterion is essentially based on the observations in node $C_m$ only. Thus, we can “profile” out the regression parameters $\beta_m$ and have

$$\hat{C}_m = \arg \min_{C_m} \sum_{m=1}^{M} \text{SSE}(C_m) := \arg \min_{C_m} \sum_{i=1}^{n} \sum_{m=1}^{M} (y_i - x_i' \hat{\beta}_m(C_m))^2 I(s_i \in C_m),$$

(4)

where $\text{SSE}(C_m) := \arg \min_{C_m} \sum_{i=1}^{n} (y_i - x_i' \beta_m)^2 I(s_i \in C_m)$. In applications where the responses are counts or categorical variables, generalized linear models (McCullagh and Nelder 1999) are often used which assume a parametric distribution of the response variable (Poisson or multinomial distribution) and relate the mean response to predictors through a link function. One can replace the least squares criterion (3) by $-2\log$ likelihood, and proceed in a similar fashion. The extension to generalized linear model framework is straightforward, and omitted in the current paper.

**Computational details**

By definition, the sets $C_m$s comprise an optimal partition of the space expanded by the partitioning variables $s$, where the “optimality” is with respect to the least squares criterion. The search for the optimal partition is of combinatorial complexity, and it is of great challenge to find the globally optimal partition even for a moderate-sized dataset. The tree-based algorithm is an approximate solution to the optimal partitioning and scalable to large-scale datasets. We restrict our discussions to binary trees that employ “horizontal” or “vertical” partitions of the feature space and are stage-wise optimal. The greedy algorithm for growing the tree cycles through the space of partition variables and finds optimal binary splits of the
feature space at every iteration. The widely used CART algorithm employs a depth-first search algorithm to grow a large tree and then prunes back based on certain cost-complexity measure. The recursive partitioning based binary trees are implemented in a number of algorithms including THAID (Morgan and Sonquist 1963), CART (Breiman et al. 1984), and C4.5 (Quinlan 1993). Stochastic search algorithms have also been proposed to find better trees than CART, and Bayesian CART has been proposed by Chipman et al. (1998) and Denison et al. (1998) that uses Markov chain Monte Carlo (MCMC) to navigate through the tree space. Genetic Algorithms have also been employed for finding trees with good performance (Goldberg 1989, Papagelis and Kalles 2001, Fan and Gray 2005).

The recursive partitioning approach grows a larger-than-necessary tree first and then prunes back to obtain a tree with a prespecified number of nodes (say $M$). This can be slow for boosting where the desired number of nodes is fairly small. Here, we employ a different tree-growing algorithm in Algorithm 1, which adopts the breadth-first search. The breadth-first search cycles through all terminal nodes at each step to find the optimal split, and stops when the number of terminal nodes reaches the desired value $M$. We use the reduction of SSE as a criterion to decide which variable to split on. For a single tree, the stopping criterion is either the size of the resulting child node is smaller than the threshold $n_0$ or the number of terminal nodes reaches $M$. The minimum node size $n_0$ needs to be specified with respect to the complexity of the regression model, and should be large enough to ensure that the regression function in each node is estimable with high probability. The number of terminal nodes $M$, which is a measure of model complexity, controls the “bias-variance tradeoff”. In the present paper, we focus on using trees as base learners for boosting in which $M$ is prespecified, but still we will briefly discuss how to tune $M$ in section 5.

In Algorithm 1, we propose to cycle through the partition variables at each iteration and consider all possible binary splits based on each variable. The candidate split depends on the type of the variable. For an ordered or a continuous variable, we sort the distinct values of the variable, and place “cuts” between any two adjacent values to form partitions. Hence for
an ordered variable with $L$ distinct values, there are $L - 1$ possible splits, which can be huge for a continuous variable in a large-scale data. Thus we specify a threshold $L_{\text{cont}}$ (say 500, for instance), and only consider splits at the $L_{\text{cont}}$ equally spaced quantiles of the variable if the number of distinct values exceeds $L_{\text{cont}} + 1$. An alternative way of speeding up the calculation is to use an updating algorithm that “updates” the regression coefficients as we change the split point, which is computationally more efficient than having to recalculate the regression every time. Here, we adopt the former approach for its algorithmic simplicity.

Splitting on an unordered categorical variable is challenging, especially when there are many categories. We consider the following three methods, including exhaustive search, category ordering and gradient descent:

1. **Exhaustive search.** We consider all possible partitions of the factor levels into two disjoint sets. For a categorical variable with $L$ categories, an exhaustive procedure will attempt $2^{L-1} - 1$ possible splits.

2. **Category ordering.** The exhaustive search is computationally intensive for a categorical variable with a large number of categories, thus we propose to order the categories in a way similar to CART to alleviate the computational burden. In a piecewise constant model like CART, Hastie et al. (2009) reviewed the idea of ordering the categories by the mean response in each category, and then treating the categorical variable as if it were an ordinal variable. This reduces the computation from exponential complexity to linear. The simplification was justified by Fisher (1958) in optimal splitting setup, and is exact for a continuous response regression problem where we model the mean. In the partitioned regression context, let $\hat{\beta}_l$ denote the least squares estimate of $\beta$ based on observations in the $l$-th category. The fitted model in the $l$-th category is denoted as $x'\hat{\beta}_l$. A strict ordering of the $x'\hat{\beta}_l$s as functions of $x$ may not exist, thus we suggest an approximate solution. We propose to use $\bar{x}'\hat{\beta}_l$ to order the $L$ categories where $\bar{x}$ is the mean vector of $x_i$s in the current node, and then treat the categorical
variable as ordinal. This approximation works well when the fitted models are clearly separated, but is not guaranteed to provide an optimal split at the current stage.

3. **Gradient descent.** The idea of ordering the categories ignores any partitions that do not conform with the current ordering, and is not guaranteed to reach a stage-wise optimal partition. A third algorithm borrows the idea of gradient descent on an integer space. In this algorithm, we start with a random partition of the $L$ categories into two nonempty and nonoverlapping groups, then cycle through all the categories and flip the group membership of each category. The $L$ group assignments resulting from flipping each individual category are compared in terms of the reduction in SSE. We then choose the grouping that maximizes the reduction in SSE as the current assignment and iterate until the algorithm converges. This algorithm performs a gradient descent on the space of possible assignments, where any two assignments are considered adjacent or reachable if they differ only by one category. The gradient descent algorithm is guaranteed to converge to a local optimum, thus we can choose multiple random starting points in the hope of reaching the global optimal. If the criterion is locally convex near the initial assignment, then this search algorithm has polynomial complexity in the number of categories.

In our numerical work, we adopt the following two strategies, the default algorithm which combines the exhaustive search, gradient descent and category ordering, and an ordering approach that always orders the categories:

**Default.** In the default tree growing algorithm, we specify a lower and an upper bound on the number of categories, namely $L_{\text{min}}$ and $L_{\text{max}}$. When the number of categories is less than or equal to the lower bound, we perform an exhaustive search; when $L_{\text{min}} < L \leq L_{\text{max}}$, we perform gradient descent with a random starting point; and when the number of categories is beyond $L_{\text{max}}$, we order the categories and treat the variable as ordinal. Without further notice, we use this tree growing algorithm with
\[ L_{\min} = 5 \text{ and } L_{\max} = 40. \]

**Ordering.** In the ordering approach, we order the categorical variable irrespective of the number of categories (i.e., \( L_{\max} = 2 \)).

The ordering approach is much faster than the default algorithm. This approach usually loses in prediction power for a single tree but performs comparably to the default method when combined with boosting.

At every stage of the tree, the algorithm cycles through the partition variables to find the optimal splitting variable. The number of possible splits can differ dramatically for different types of variables and splitting methods. For continuous and ordinal variables, the number of possible splits depends on the number of distinct values, capped by \( L_{\text{cont}} \); while for categorical variables, this number is exponential in the number of categories under exhaustive search, and linear if the variable is ordered. The number of attempted splits vary from one variable to another, which introduces bias in the selection of which variable to split on. Usually, variables that afford more splits, especially categorical variables with many categories, are favored by the algorithm. Note that we can alleviate this issue by category ordering, which reduces the possible splits on the variable to be linear. For classification and regression trees, various other algorithms have been proposed to reduce the selection bias, e.g., the QUEST and CRUISE algorithms (Loh and Shih 1997, Loh and Kim 2001, Loh 1997), which use hypothesis testing to select the split variable. Their proposed ideas can be extended to our setting, which leaves room for further research on this topic.

### 4 Boosting Tree-based Varying-coefficient Regression

Boosting was originally proposed by Freund and Schapire (1995, 1996, 1997) for classification problems, and is now widely recognized as a powerful statistical learning method for building predictive models. Boosting is competitive in high-dimensional classification and regression problems (Bühlmann and Hothorn 2007, Hastie et al. 2009). Boosting is usually
Algorithm 1 Iterative “PartReg” Algorithm (Breadth-first search).

Require: $n_0$ – the minimum number of observations in a terminal node and $M$ – the desired number of terminal nodes.

1. Initialize the current number of terminal nodes $l = 1$ and $C_m = \mathbb{R}^q$.

2. While $l < M$, loop:

   (a) For $m = 1$ to $l$ and $j = 1$ to $q$, repeat:

      i. Consider all partitions of $C_m$ into $C_{m,L}$ and $C_{m,R}$ based on the $j$-th variable. The maximum reduction in SSE is,

      \[
      \Delta \text{SSE}_{m,j} = \max \{ \text{SSE}(C_m) - \text{SSE}(C_{m,L}) - \text{SSE}(C_{m,R}) \},
      \]

      where the maximum is taken over all possible partitions based on the $j$-th variable such that $\min \{ \#C_{m,L}, \#C_{m,R} \} \geq n_0$ and $\#C$ denotes the cardinality of set $C$.

      ii. Let $\Delta \text{SSE}_l = \max_m \max_j \Delta \text{SSE}_{m,j}$, namely the maximum reduction in the sum of squared error among all candidate splits in all terminal nodes at the current stage.

   (b) Let $\Delta \text{SSE}_{m^*,j^*} = \Delta \text{SSE}_l$, namely the $j^*$-th variable on the $m^*$-th terminal node provides the optimal partition. Split the $m^*$-th terminal node according to the optimal partitioning criterion and increase $l$ by 1.
equipped with a “base learner”, for example a simple tree in tree boosting or a component-wise regression in $L_2$-boosting (Bühlmann and Yu 2003, Bühlmann 2006). We iteratively apply the base learner to reweighted data to fit multiple models, and combine these ensemble models to obtain the final prediction model. It has been pointed out by Breiman (1998, 1999) and Friedman (2001) that boosting can be viewed as a gradient descent algorithm in functional space, which places the boosting algorithm in the framework of statistical estimation. Asymptotic consistency of various boosting examples was established by Jiang (2004), Logosi and Vayatis (2004), Zhang and Yu (2005) and Bartlett and Traskin (2007), among others.

The tree model results in discontinuous estimates of $\beta(s_i)$, and we can obtain “smoother” estimates by combining multiple trees via boosting. We work under the varying-coefficient model (1): $y_i = f(x_i, s_i) + \epsilon_i = x_i' \beta(s_i) + \epsilon_i$, and estimate this model by gradient boosting using $M$-node tree as base learner. Friedman (2001) and Bühlmann and Hothorn (2007) provided comprehensive overviews of the generic gradient boosting algorithm. Here, our target is to estimate a real-valued function $f(x, s) = x' \beta(s)$ by minimizing the expected risk with respect to some loss function. Given response variable $y$ and the fitted model $f$, the loss function is denoted as $\phi(y, f)$, which is usually differentiable and convex with respect to $f$. Examples of loss functions $\phi(y, f)$ include the squared error loss $\phi(y, f) = (y - f)^2/2$ for least squares regression and absolute error loss $\phi(y, f) = |y - f|$ for $L_1$-regression, to name a few. Note that the model fit $f$ is a function of both the partition variables and regression variables. We desire to estimate the model $f$ by minimizing the empirical risk measure,

$$\hat{f} = \arg\min_{f \in \mathcal{F}_n} \frac{1}{n} \sum_{i=1}^{n} \phi(y_i, f(x_i, s_i)), \quad (6)$$

where $\mathcal{F}_n = \{f(x, s) \mid f(x, s) = x' \beta(s)\}$ denotes the constrained functional space. The constrained space $\mathcal{F}_n$ forces the resulting estimates to be linear in the regression variables, but does not restrict the functional form of the varying-coefficient variables.

We discuss how to construct the boosted tree model here. Assume that we are in the $b$-th iteration and have model fits $\hat{f}_{b-1}(x_i, s_i)$. The goal is to find an incremental model,
denoted as $T(x_i, s_i)$, that minimizes the empirical risk,

$$\hat{T} = \arg\min_{T \in \mathcal{F}_n} \frac{1}{n} \sum_{i=1}^{n} \phi(y_i, \hat{f}^{(b-1)}(x_i, s_i) + T(x_i, s_i)),$$

where $\mathcal{F}_n$ is defined previously. If we use $M$-node trees as base learners, we approximate the increment $T(x_i, s_i)$ by a piecewise regression function $x_i' \sum_{m=1}^{M} \beta_m^{(b)} I(s_i \in \hat{C}_m^{(b)})$, and try to solve the following minimization problem,

$$(\hat{C}_m^{(b)}, \hat{\beta}_m^{(b)}) = \arg\min_{(C_m^{(b)}, \beta_m^{(b)})} \frac{1}{n} \sum_{i=1}^{n} \phi \left( y_i, \hat{f}^{(b-1)}(x_i, s_i) + x_i' \sum_{m=1}^{M} \beta_m^{(b)} I(s_i \in \hat{C}_m^{(b)}) \right).$$

Our algorithm 1 can be adopted to find an approximate solution to the above minimization problem. In cases of $L_2$ loss, we can compute the residuals $y_i - \hat{f}^{(b)}(x_i, s_i)$ and fit the residuals on $(x_i, s_i)$ using the partition algorithm.

The gradient boosting algorithm is described in Algorithm 2. Here, we start with a simple fit $\hat{f}^{(0)}$ and then iteratively update the estimate by adding the incremental model fitted on “residuals”. At each boosting step, we fit an $M$-node tree using Algorithm 1, and use the fitted tree to produce pseudo observations $u_i = -\frac{\partial}{\partial f} \phi(y_i, f) \bigg|_{f=\hat{f}}$, which will be used as response in the next iteration. The value of pseudo observations depends on the loss function and the current model estimate. In case of $L_2$ loss, the pseudo residuals are simply regression residuals under the current model. At the end we combine these multiple trees to obtain the final boosting model,

$$\hat{f}^{(B)} = \hat{f}^{(0)} + \nu \sum_{b=1}^{B} \sum_{m=1}^{M} x_i' \beta_m^{(b)} I(s_i \in \hat{C}_m^{(b)})$$

which is again a tree model. The boosting model is based on a more granular partition of the space expanded by $s$. Here, $B$ is the number of boosting steps and $0 < \nu \leq 1$ is regularization parameter that controls the learning rate. These two tuning parameters will be further explained shortly. Algorithm 2 uses the sample mean $\bar{y}_n$ as initial fit $\hat{f}^{(0)}$. The final model $\hat{f}^{(B)}(x_i, s_i)$ can be represented as $x_i' \hat{\beta}(s_i)$ if an intercept is contained in the regression, otherwise we can either use zero as the initial fit $\hat{f}^{(0)}$ or model the centered
Algorithm 2 Boosting “PartReg” Algorithm.

Require: \( B \) – the number of boosting steps, \( M \) – the number of terminal nodes in each base learner, \( n_0 \) – the minimum node size for each base tree, and \( \nu \) – the regularization parameter in boosting.

1. Start with \( \hat{f}(0) = \arg \min_c \frac{1}{n} \sum_{i=1}^n \phi(y_i, c) \). An alternative is to fit an ordinary linear model as the initial model, namely, \( \hat{f}(0) = \arg \min_f \frac{1}{n} \sum_{i=1}^n \phi(y_i, f(x_i, s_i)) \), where \( f(x_i, s_i) = x_i' \beta \).

2. For \( b = 1, \cdots, B \), repeat:

   (a) Compute the negative gradient evaluated at the current fit \( u_i = \left. -\frac{\partial}{\partial f} \phi(y_i, f) \right|_{f=\hat{f}(b-1)} \), for \( i = 1, \cdots, n \).

   (b) Fit \( u_i \) on \( s_i \) and \( x_i \) using the iterative “PartReg” algorithm (Algorithm 1) to obtain

   \[ \hat{u}_i = \sum_{m=1}^M x_i' \beta_m^{(b)} I(s_i \in \hat{C}_m^{(b)}) \].

   (c) Update the fitted regression function by \( \hat{f}(b) = \hat{f}(b-1) + \nu \sum_{m=1}^M x_i' \beta_m^{(b)} I(s_i \in \hat{C}_m^{(b)}) \).

3. Output the fitted model \( \hat{f} = \hat{f}(B) \).
response variable. An alternative initial fit is \( \hat{f}^{(0)} = \mathbf{x}' \hat{\beta} \) with constant coefficient \( \hat{\beta} \) that minimizes the empirical risk (6).

The boosting algorithm involves four tuning parameters: the number of boosting iterations \( B \), the learning rate \( \nu \), the minimum node size \( n_0 \) and the size of each base learner \( M \). The unregularized model uses a learning rate of \( \nu = 1 \), and empirical evidence has shown that smaller values of \( \nu \) lead to superior predictive performance on test data. The parameters \( \nu \) and \( B \) heavily depend on each other, and having a smaller value of \( \nu \) often requires a larger number of iterations to obtain comparable out-of-sample prediction power. The learning rate \( \nu \) is fixed at 0.01 hereafter. The minimum node size \( n_0 \) should be large enough so that we have sufficient observations to estimate the regression model. The size of each tree controls the levels of interactions among the partition variables, and having \( M \) terminal nodes allows up to order \( M - 1 \) interactions. In our numerical studies, we fix all the tuning parameters except \( B \), and use the test error to choose \( B \).

**Interpretation of the boosting model**

The boosting model is often hard to visualize given the high-dimensionality of the partition variables, so in this section, we provide two methods for interpreting and visualizing the estimated model: the measure of relative importance and partial dependence plot. The measure of variable importance pertains to the partition variables only, as the regression variables \( \mathbf{x}_i \) are often key predictor variables thus it is not necessary to test the importance of components of \( \mathbf{x}_i \). The partial dependence plot depicts the partial (not marginal) dependence of \( \hat{f} \) on either the regression or partition variables, as will be illustrated later.

Friedman (2001) and Hastie et al. (2009) discussed a measure of variable importance for boosting regression trees, which is related to the reduction of the squared error loss contributed by the splitting variable. Here, we first define the importance of variable \( s_j \) in a
single tree $T$ as,

$$I_j^2(T) = \sum_{l=1}^{M-1} \Delta \text{SSE}_l I_{v(l) = s_j},$$

where $v(l)$ denotes the variable chosen for splitting in the $l$-th step, and the improvement in model fit, $\Delta \text{SSE}_l$, is defined in (5). Thus the importance of the $j$-th variable $s_j$ is defined as the sum of $\Delta \text{SSE}_l$s when the variable is chosen for splitting the tree. In boosting, we denote $b$-th tree as $T_b$, and compute the relative importance of $s_j$ as the average of $I_j^2(T_b)$ among all iterations, namely,

$$I_j^2 = \frac{1}{B} \sum_{b=1}^{B} I_j^2(T_b).$$

We can then rescale the $I_j^2$s so that the largest importance index is 100, and then represent the variable importance metrics via barplots as in Figure 6.

In partial dependence plot, we split the predictors into two mutually exclusive sets, one set containing variables whose effects on response are of interest and the other set containing variables over which we average. In varying-coefficient models, we are most interested in the relationship between $y$ and $x$, and how this relationship varies at different values of $s$. The partial dependence of $y$ on $x$, with all partition variables being averaged over, is a linear regression with coefficients defined as,

$$\hat{\beta}(s_i) = \frac{1}{n} \sum_{i=1}^{n} \hat{\beta}(s_i).$$

The partial dependence function is generally different than the linear model estimates. The function $x'\hat{\beta}(s_i)$ is equivalent to the ordinary linear regression function if squared error loss is used and the regression variables are independent of the partition variables. When $x_i$ and $s_i$ are dependent, $x'\hat{\beta}(s_i)$ can be quite different than ordinary regression since $x'\hat{\beta}(s_i)$ computes the expectation of $\hat{f}(x, s_i)$ over the marginal distribution of $s_i$, not the conditional distribution $[s_i \mid x]$. Another interest is to see how the regression coefficients change under different values of $s$. For ease of presentation, we vary one partition variable $s_j$ only. Let $s_{i,(-j)}$ denote the subvector of $s_i$ with the $j$-th element dropped. We define

$$\hat{\beta}(s_j) = \frac{1}{n} \sum_{i=1}^{n} \hat{\beta}(s_j, s_{i,(-j)}).$$
where the expectation is taken over the marginal distribution of $s_{i,(-j)}$. The marginal function remains to be a linear function of $x$. We can specify different values of $s_j$, and examine how $x'\hat{\beta}(s_j)$ changes as a function of $x$ when we vary $s_j$.

5 Simulations

In this section, we present simulation results on the performance of the tree and boosting varying-coefficient models. We are interested in comparing the predictive performance of the two models, as well as the efficiency of reconstructed varying-coefficient functions. We consider varying-coefficient simple linear regression, where both the intercept and slope are functions of a 10-dim vector $s$. The model is $y_i = \beta_0(s_i) + \beta_1(s_i)x_i + \epsilon_i$, $i = 1, \cdots, n$, where $x_i$ follows a standard normal distribution, and $s_i$s are random samples from a uniform distribution on $[0, 1]^{10}$. The model errors $\epsilon_i$ follow a normal distribution with mean zero and variance $0.5^2$. The intercept and slope are both additive functions of $s_1$ and $s_2$ only to reflect sparsity, with $\beta_0(s_i) = 2\sin^2(2\pi s_{1i}) + \exp(2s_{2i} - 1)$, and $\beta_1(s_i) = 2\cos^2(2\pi s_{1i}) + 8s_{2i}(1 - s_{2i})$. The sample size $n = 2000$, and we leave out 20% of the observations as test data. Here, we consider the tree-based regression model with minimum node size $n_0 = 10$ and boosting 4-node trees with regularization parameter of $\nu = 0.01$. The default algorithm for growing the tree is applied in boosting. The number of terminal nodes $M$ in the tree model and the number of boosting iterations $B$ are both tuning parameters and chosen by data-driven methods.

In the tree model, the number of ultimate partitions $M$ is unknown, and we suggest choosing $M$ by examining the risk measure on a test sample. Let $(s'_i, x'_i, y_i), i = n+1, \cdots, N$ denote the test observations, and $(\hat{\beta}_m, \hat{C}_m)$ denote the estimated regression coefficients and partitions based on training sample and $M$ denote the range of tree sizes. The number of terminal nodes $M$ is chosen by minimizing the out-of-sample prediction error,

$$
\hat{M} = \arg\min_{M \in \mathcal{M}} \sum_{i=n+1}^{N} \left( y_i - \sum_{m=1}^{M} x'_i \hat{\beta}_m I(s_i \in \hat{C}_m) \right)^2.
$$

(7)
The above procedure is employed for tuning the tree model in both the simulation and application to real-world data.

In Figure 2, we plot the ratio of the mean squared error loss to the true residual variance against the tuning parameter. The $L_2$ loss for the tree becomes constant when $M \geq 118$, since the terminal nodes are so small that no further splits can be made hereafter that satisfy the node size requirement. For both models, overfitting causes the test error curve to become (nearly) flat beyond a certain threshold and then increase slightly. The overfitting of the tree model has been alleviated by specifying the minimum node size; and boosting does not have serious issue of overfitting either. Overall, boosting has smaller training and test error than tree-based method, which is true for a number of other simulation scenarios as well. For both methods, we choose the tuning parameters by minimizing the $L_2$ loss on test set, in order to automate the choice of tuning parameters.

![Figure 2: $L_2$ risk for tree-based and boosting varying-coefficient models with default tree growing algorithm; the solid line represents the $L_2$ risk on training data and the dashed line represents the $L_2$ risk on test data; the vertical axis shows the ratio of the squared error loss to the true error variance.](image)

To examine the estimation of $\beta_0$ and $\beta_1$ as functions of $s_1$ and $s_2$, we repeated the simulation 20 times and plotted the estimated varying-coefficient functions along with the
true functions in Figure 3. The true functions are $\beta_0$ and $\beta_1$ as marginal functions of $s_1$ and $s_2$, where we have taken expectations with respect to $s_{(-1)}$ and $s_{(-2)}$. The estimated curves based on either tree-based or boosting varying-coefficient regression are plotted alongside as dashed curves. In both methods, the nonlinear coefficient functions can be approximated reasonably well. We can see that the boundary and the high-curvature region are harder to estimate for both methods, indicated by larger bias and higher variance. This is frequently seen in nonparametric estimation, for example kernel smoothing and spline regressions. Boosting produces superior curve estimates than a single tree, in terms of both bias and variance. Further, the boosting estimates are “smoother” than the tree estimates.

Figure 3: Reconstructed varying-coefficient surfaces by tree and boosting; the left panel shows the tree estimates and right panel shows the boosting estimates; 20 simulations are conducted under each method; the solid curves show the true functions and the dashed curves show our estimates.
Figure 4: The left panel plots the sales units against the average selling price and the right panel shows the log-transformed sales units against the price. Various product configurations are represented by different colors and have shown to have varying demand functions, which motivates the use of varying-coefficient models.

6 Application to Mobile Computer Sales Data

The proposed varying-coefficient models have been applied to the aggregated monthly mobile computer sales data in Australia, obtained from a third-party marketing firm. The dataset contains the sales information of various categories of computer models, including laptops, netbooks, hybrid tablets, ultra-mobile personal computers and so on. The monthly sales data goes from October 2010 to March 2011. Each row of the data set contains detailed configurations of the respective product, the number of units being sold, the revenue generated from selling the product in certain month and state. The average selling price is derived by taking the ratio of the revenue to the number of units being sold. We have screened the products that sell very few, and the screening leaves us with 164 distinct configurations and around 4,500 observations.

We use a varying-coefficient linear model to predict the demand. The sales units and log-transformed sales units are plotted against price in Table 4. The product price ranges
from nearly 200 to over 2500 U.S. dollars. We also observe that the distribution of the untransformed sales is highly skewed while the marginal distribution of the log-transformed sales is more symmetric. Thus we use the log-transformed variable as our modeling target. Let $y_i$ denote the number of units sold, $x_i$ denote the average selling price and $s_i$ denote the vector of varying-coefficient variables, including the month, state, sales channel and laptop features. The model we fit is

$$\log(y_i) = \beta_0(s_i) + \beta_1(s_i) + \epsilon_i,$$

which is estimated via the tree-based method as well as two variations of boosting: the default boosting algorithm and boosting a tree that orders the categorical variables. The minimum node size in the tree model is fixed at $n_0 = 10$. In boosting, we use 4-node trees as the base learner with a regularization parameter of $\nu = 0.05$. The tuning parameters, namely $M$ in a single tree and $B$ in boosting, are chosen by minimizing the squared error loss on a test sample. The $L_2$ loss on training and test sample is plotted on Figure 5 for both single tree and boosting. We can clearly observe the improvement in prediction performance by boosting the tree from Figure 5. The comparison between the default and the ordered boosting algorithms is also interesting, and it takes fewer iterations, but more computation time for the default boosting to achieve the same out-sample performance as the ordered boosting algorithm.

Figure 6 provides a graphical display of variable importance under the two boosting models. The left and right panels show the importance index under the default boosting and ordered boosting models, respectively. The variables with an importance measure less than 10 are suppressed from this plot. The two variations of boosting select similar sets of important variables, and the environmental variables including the state (REG), month (MONTH) and sales channels (REPORT) all have non-trivial effects on the demand function. Variables like the GPU model (GPU.MO), processor (PROCESS) and clock speed (CSP.MH) are also important in determining the price-demand relationship. The difference between the default and ordered boosting algorithms is that the former favors categorical variables with a
Figure 5: $L_2$ risk for tree-based varying-coefficient model and two variations of boosting varying-coefficient models; the middle panel shows boosting with default search algorithm when splitting on a categorical variable and the right panel shows boosting that orders the categories in growing each base tree; the solid and dashed lines represent the $L_2$ risk on training and test data, respectively. It took the ordered boosting algorithm twice as many iterations but $1/6$ of computation time, compared to the default boosting algorithm, to achieve the same predictive performance.

lot of categories. Figure 7 shows the partial dependence plot on brand, where in the context of varying-coefficient models, the key interest is how the regression functions change with the “control variable”. Both the variable importance and partial dependence plots have been presented to the pricing analysts to gain some business insights.

7 Discussion

In this paper, we have proposed a nonparametric varying-coefficient model based on statistical learning methods, motivated by the necessity of predicting the market demand for laptops. In this application, we have a large number of mixed-type varying-coefficient variables, which poses great challenges for conventional nonparametric smoothing methods due to the “curse of dimensionality”. The proposed method starts with a tree model that partitions the space of varying-coefficient variables and uses boosting to improve the prediction
Figure 6: The left panel shows the variable importance plot from default boosting algorithm and the right panel shows that from ordered boosting algorithm. Note that certain categorical variables (e.g., GPU.MO) with many categories have been downweighted in the latter approach.

performance as well as smoothing out the coefficient estimates. We have examined the performance of the proposed approach in a simulation and with an application to the marketing data. Our proposed model is a novel application of statistical learning algorithms to varying-coefficient models, and at the same time, can be regarded as a structured learning method that has realistic implications.

Our exposition primarily focuses on varying-coefficient linear regression estimated with a least squares criterion. But the methodology is readily generalized to nonlinear and generalized linear models, with a wide range of loss functions. One can use more robust loss functions, or use likelihood-based criteria for non-Gaussian data. And our proposed varying-coefficient model is applicable to classification problems with little modification. The estimation algorithms can be easily adapted to these more general situations.
Figure 7: The partial dependence plot of demand on product price and brand. The actual laptop brands are suppressed and denoted by alphabetical letters. The figure shows how the volume (level or intercept) and sensitivity (slope) of the demand vary between different brands.
Acknowledgements

The first author gratefully acknowledges the constructive comments from Kay-Yut Chen, Enis Kayis, Jose Luis Beltran, Ruxian Wang and Shailendra Jain in Hewlett-Packard Labs, and Guillermo Gallego at Columbia University.

References


