Boosted Varying-Coefficient Regression Models for Product Demand Prediction

December 5, 2012

Jianqiang C. Wang[1] Trevor Hastie[2]

Abstract

Estimating the aggregated market demand for a product in a dynamic market is critical to manufacturers and retailers. Motivated by the need for a statistical demand prediction model for 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, and 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 proposed method works well in both predicting the response and estimating the coefficient surface, based on a simulation study. Finally, we have applied this methodology to real-world mobile computer sales data, and demonstrated its superiority by comparing with elastic net and kernel regression based varying-coefficient model.

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

[1]Hewlett-Packard Labs, Palo Alto, CA, 94304.

[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 offered mobile computers. 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. 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) pioneered by McFadden (1974). 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, concerns about overfitting arise 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

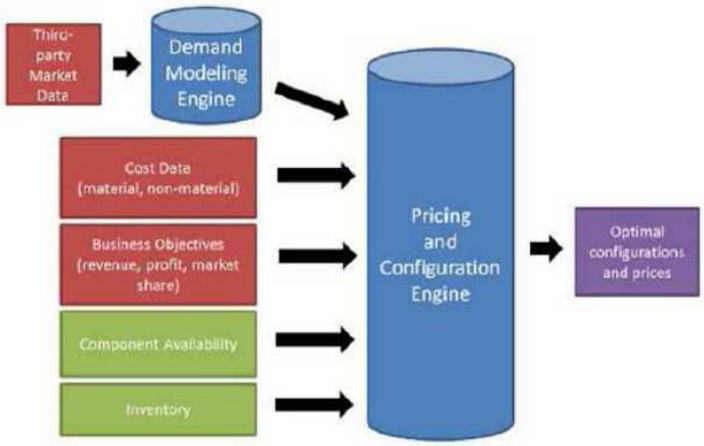

Figure 1: The flow chart of the pricing optimization project launched by HP; the items in blue represent the developed modules, items in red represent available input data, and the items in green mean information that could potentially be used.

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 and discussed

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 demand prediction, the varying-coefficient variables include various product features and environmental variables like the time and region. The regression coefficients are thus functions of high-dimensional variates, which need to be estimated based on data. 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 wish to have a data-driven approach for estimating the high-dimensional non-additive coefficient functions. The tree-based approach, initally proposed by Breiman et al. (1984) under the name of classification and regression trees (CART), has proven 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 method iteratively partitions the predictor space and fits a constant to each partition. Conventional tree-growing algorithms employ exhaustive search, for example in CART (Breiman et al. 1984) and C4.5 (Quinlan 1993), or significance test adjusted search as in THAID (Morgan and Sonquist 1963), for choosing the split variable and splitt rule. Other search algorithms have also been proposed to find better trees than CART, including Bayesian CART (Chipman et al. 1998 and Denison et al. 1998) that uses Markov chain Monte Carlo (MCMC) to navigate through the space of trees and Genetic Algorithms (Goldberg 1989, Papagelis and Kalles 2001, Fan and Gray 2005), among others.

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 pertubation to the data may lead to a dramatically changed model (Hastie et al. 2009, chapter 9). An ensemble method called *boosting* has been introduced to improve the

predictive performance of tree models by combining multiple trees. Boosting was originally proposed by Freund and Schapire (1995, 1996, 1997) for classification, and is now widely recognized as a powerful statistical learning method. Boosting is competitive in high-dimensional classification and regression problems (Bühlmann and Hothorn 2007, Hastie et al. 2009). Boosting is usually 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). The base learner is iteratively applied to the reweighted data to fit multiple models, which are combined 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 provides a statistical interpretation of the learning algorithm. 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. Both the single-tree based model and boosting naturally handle variable interactions and automatically select the important variables.

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 introduces the boosted tree-based varying-coefficient regression model, as an alternative to kernel smoothing and spline-based methods, for estimating the varying-coefficient surfaces. 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, and compares the empirical performance of the proposed method with existing methods, including elastic net based approach and kernel smoothing. 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 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, $\boldsymbol{x} \in \mathbb{R}^p$ denote the vector of predictors that a parametric relationship is available between y and $\boldsymbol{x}$, for any given values of *modifying* predictor vector $\boldsymbol{s} \in \mathbb{R}^q$. The regression relationship between y and $\boldsymbol{x}$ varies under different values of $\boldsymbol{s}$. Let $(\boldsymbol{s}_i', \boldsymbol{x}_i', y_i)$ denote the measurements on subject i, where $i = 1, \cdots, n$. Here, the varying-coefficient variable is $\boldsymbol{s}_i = (s_{i1}, s_{i2}, \cdots, s_{iq})'$ and the regression variable is $\boldsymbol{x}_i = (x_{i1}, x_{i2}, \cdots, x_{ip})'$, and we allow overlap between the two sets of variables. The first element of $\boldsymbol{x}_i$ is set to be 1 if we allow for an intercept term. 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,

$$y_i = f(\boldsymbol{x}_i, \boldsymbol{s}_i) + \epsilon_i = \boldsymbol{x}_i' \boldsymbol{\beta}(\boldsymbol{s}_i) + \epsilon_i, \tag{1}$$

where the regression coefficients $\boldsymbol{\beta}(\boldsymbol{s}_i)$ are modeled as functions of $\boldsymbol{s}$.

In model (1), our key interest is to estimate the multivariate coefficient surface $\boldsymbol{\beta}(\boldsymbol{s}_i)$. Fan and Zhang (2008) gave an excellent review of the varying-coefficient models and discussed three approaches in estimating the coefficient function $\boldsymbol{\beta}(\boldsymbol{s}_i)$: 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 $\boldsymbol{s}_i$ 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 appropriately to avoid modeling overfitting. When boosting the tree, we can fix the number of nodes at a small value, but tune the number of boosting iterations instead (Hastie et al. 2009, p. 364). We do not seek to draw distinctions between these two perspectives, but attempt to integrate the two views during our presentation.

The tree-based method approximates $\boldsymbol{\beta}(\boldsymbol{s}_i)$ in (1) by a piecewise constant function. More specifically, the tree algorithm seeks to partition the space of $\boldsymbol{s}$ according to certain information criterion and then approximate $\boldsymbol{\beta}(\boldsymbol{s}_i)$ 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 $\boldsymbol{x}$ as *regression variables*. The idea of partitioning the space of varying coefficient variables $\boldsymbol{s}$, 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 $\cup_{m=1}^{M} C_m = \mathbb{R}^q$. Here, M denotes the number of partitions. The proposed tree-based varying-coefficient model is

$$y_i = \sum_{m=1}^{M} \boldsymbol{x}_i' \boldsymbol{\beta}_m \mathrm{I}_{(\boldsymbol{s}_i \in C_m)} + \epsilon_i, \tag{2}$$

where $\mathrm{I}_{(\cdot)}$ denotes the indicator function with $\mathrm{I}_{(c)} = 1$ if event c is true and zero otherwise. The error terms ϵ_is are assumed to have zero mean and homogeneous variance σ^2. The implied varying-coefficient function is thus,

$$\boldsymbol{\beta}(\boldsymbol{s}_i) \approx \sum_{m=1}^{M} \boldsymbol{\beta}_m \mathrm{I}_{(\boldsymbol{s}_i \in C_m)},$$

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 $\boldsymbol{\beta}_m$ 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 like sum of squared prediction error (Hastie et al. 2009, section 9.2). Alternatively, we can automatically determined M by imposing certain stopping rules like the minimum sample size of a terminal node. 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 $(C_m, \boldsymbol{\beta}_m)$, as minimizers of sum of squared errors (SSE),

$$(\widehat{C}_m, \hat{\boldsymbol{\beta}}_m) = \underset{(C_m, \boldsymbol{\beta}_m)}{\arg\min} \sum_{i=1}^{n} \left(y_i - \sum_{m=1}^{M} \boldsymbol{x}_i' \boldsymbol{\beta}_m \mathrm{I}_{(\boldsymbol{s}_i \in C_m)} \right)^2 = \underset{(C_m, \boldsymbol{\beta}_m)}{\arg\min} \sum_{i=1}^{n} \sum_{m=1}^{M} \left(y_i - \boldsymbol{x}_i' \boldsymbol{\beta}_m \right)^2 \mathrm{I}_{(\boldsymbol{s}_i \in C_m)}. \tag{3}$$

In the above, the estimation of $\boldsymbol{\beta}_m$ is nested in that of the partitions. We define $\hat{\boldsymbol{\beta}}_m(C_m)$ as a consistent estimator of $\boldsymbol{\beta}_m$ 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,

$$\hat{\boldsymbol{\beta}}_m(C_m) = \underset{\boldsymbol{\beta}_m}{\arg\min} \sum_{i=1}^{n} \left(y_i - \boldsymbol{x}_i' \boldsymbol{\beta}_m \right)^2 \mathrm{I}_{(\boldsymbol{s}_i \in C_m)},$$

in which the minimization criterion is essentially based on the observations in node C_m only.

Thus, we can "profile" out the regression parameters $\boldsymbol{\beta}_m$ and have

$$\widehat{C}_m = \arg\min_{C_m} \sum_{m=1}^{M} \mathrm{SSE}(C_m) := \arg\min_{C_m} \sum_{i=1}^{n} \sum_{m=1}^{M} \left(y_i - \boldsymbol{x}_i' \hat{\boldsymbol{\beta}}_m(C_m) \right)^2 \mathrm{I}_{(\boldsymbol{s}_i \in C_m)}, \tag{4}$$

where $\mathrm{SSE}(C_m) := \arg\min_{C_m} \sum_{i=1}^{n} (y_i - \boldsymbol{x}_i' \boldsymbol{\beta}_m)^2 \mathrm{I}_{(\boldsymbol{s}_i \in C_m)}$. 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 -2log likelihood, and proceed in a similar fashion. The extension to generalized linear models is presented in section **??** of the paper.

Computational details

By definition, the sets $\{C_m\}_{m=1}^{M}$ comprise an optimal partition of the space expanded by the partitioning variables $\boldsymbol{s}$, where the "optimality" is with respect to the least squares criterion. The search for the optimal partition is of combinatorial complexity, and it is challenging 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 tree-growing algorithm cycles through the space of partition variables and searches for 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 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. Hence, we employ a different tree-growing algorithm shown 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 being 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 will still 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 variable type. For an ordinal or a continuous variable, we sort the distinct values of the variable, and place "cutoffs" between any two adjacent values to form partitions. Hence, for an ordinal variable with L distinct values, there are $L-1$ possible splits, which can be huge for a continous variable in a large-scale data. To speed up, we specify a threshold value L_{cont} (say 500, for instance), and only consider splits at the L_{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 procedure for its algorithmic simplicity.

Splitting on an unordered categorical variable

Splitting based on an unordered categorical variable is challenging, especially when there are many categories. In our research, we have considered exhaustive search, gradient descent in the space of all possible binary splits and a third approach by ordering the categories. The

exhaustive search and gradient descent are both computational intensive, while the ordering approach is much faster. The ordering approach usually loses prediction power for a single tree but performs comparably to the more complex search algorithms when combined with boosting. For the sake of brevity, we limit our discussion to the ordering-based tree-growing algorithm.

Our category ordering approach is similar to that of CART (Breiman et al. 1984). In a piecewise constant model like CART, the categories can be ordered based on the mean response in each category, and then treated as ordinal variables (Hastie et al. 2009). This reduces the computation complexity from exponential to linear for binary splits. Fisher (1958) justified the simplification to be exact for a continuous-response regression problem where the mean is the modeling target. In partitioned regression, let $\hat{\boldsymbol{\beta}}_l$ denote the least squares estimator of $\boldsymbol{\beta}$ based on observations in the l-th category. The fitted model in the l-th category is denoted as $\boldsymbol{x}'\hat{\boldsymbol{\beta}}_l$. A strict ordering of the hyperplanes $\boldsymbol{x}'\hat{\boldsymbol{\beta}}_l$ may not exist, thus we resort to an approximate solution. We propose to order the L categories using $\bar{\boldsymbol{x}}'\hat{\boldsymbol{\beta}}_l$, where $\bar{\boldsymbol{x}}$ is the mean vector of $\boldsymbol{x}_i$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 a stage-wise optimal split.

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_{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 potential splits vary from one variable to another, which introduces bias to the selection of which variable to split on. Another factor that contributes to "unfair" split selection is the category ordering procedure. For an ordinal and an unordered categorical variable with the same number of categories, the

number of possible splits is the same. But the unordered variable is more likely to be chosen as the splitting variable, when neither of the two variables has any effect on the response. The reason is that the category ordering procedure is "response-driven", which has used the response information in ordering the categories. In spite of this, the category ordering is still favored than exhaustive search, since the latter is more biased towards the unordered categorical variables. 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 splitting 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

The tree model results in discontinuous estimates of $\boldsymbol{\beta}(\boldsymbol{s}_i)$. We can obtain "smoother" estimates by combining multiple trees via boosting. We work under the varying-coefficient model (1): $y_i = f(\boldsymbol{x}_i, \boldsymbol{s}_i) + \epsilon_i = \boldsymbol{x}_i'\boldsymbol{\beta}(\boldsymbol{s}_i) + \epsilon_i$, 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 $f(\boldsymbol{x}, \boldsymbol{s}) = \boldsymbol{x}'\boldsymbol{\beta}(\boldsymbol{s})$ 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 in our case. We wish to estimate the model f by minimizing the empirical risk measure,

$$\hat{f} = \arg\min_{f \in \mathcal{F}_n} \frac{1}{n} \sum_{i=1}^{n} \phi(y_i, f(\boldsymbol{x}_i, \boldsymbol{s}_i)), \tag{6}$$

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,

$$\Delta\text{SSE}_{m,j} = \max\{\text{SSE}(C_m) - \text{SSE}(C_{m,L}) - \text{SSE}(C_{m,R})\}, \tag{5}$$

      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) Define $(m^*, j^*) = \text{argmax}_{m,j}\, \Delta SSE_{m,j}$ where $m \in \{1, \cdots, l\}$ and $j \in \{1, \cdots, q\}$, namely the j^*-th variable on the m^*-th terminal node provides the optimal partition. Then split the m^*-th terminal node according to the optimal partitioning rule, denote $\Delta\text{SSE}_l = \Delta\text{SSE}_{m^*,j^*}$, and increase l by 1.

where $\mathcal{F}_n = \{f(\boldsymbol{x}, \boldsymbol{s}) \mid f(\boldsymbol{x}, \boldsymbol{s}) = \boldsymbol{x}'\boldsymbol{\beta}(\boldsymbol{s})\}$ 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-coefficients.

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

$$\widehat{T} = \underset{T \in \mathcal{F}_n}{\arg\min} \frac{1}{n} \sum_{i=1}^{n} \phi(y_i, \hat{f}^{(b-1)}(\boldsymbol{x}_i, \boldsymbol{s}_i) + T(\boldsymbol{x}_i, \boldsymbol{s}_i)),$$

where $\mathcal{F}_n$ is defined previously. If we use M-node trees as base learners, we approximate the increment $T(\boldsymbol{x}_i, \boldsymbol{s}_i)$ by a piecewise regression function $\boldsymbol{x}_i' \sum_{m=1}^{M} \boldsymbol{\beta}_m^{(b)} \mathrm{I}_{(\boldsymbol{s}_i \in C_m^{(b)})}$, and seek to solve the following minimization problem,

$$(\widehat{C}_m^{(b)}, \hat{\boldsymbol{\beta}}_m^{(b)}) = \underset{(C_m^{(b)}, \boldsymbol{\beta}_m^{(b)})}{\arg\min} \frac{1}{n} \sum_{i=1}^{n} \phi \left(y_i, \hat{f}^{(b-1)}(\boldsymbol{x}_i, \boldsymbol{s}_i) + \boldsymbol{x}_i' \sum_{m=1}^{M} \boldsymbol{\beta}_m^{(b)} \mathrm{I}_{(\boldsymbol{s}_i \in C_m^{(b)})} \right).$$

Our Algorithm 1 can be adopted to find an approximate solution to the above minimization problem. Under L_2-loss, we can compute the residuals $y_i - \hat{f}^{(b)}(\boldsymbol{x}_i, \boldsymbol{s}_i)$ and fit the residuals on $(\boldsymbol{x}_i, \boldsymbol{s}_i)$ 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 evaluate the pseudo residuals $u_i = -\frac{\partial}{\partial f}\phi(y_i, f)\Big|_{f=\hat{f}}$ at the fitted tree, which will be used as response in the next iteration. The value of pseudo residuals depends on the loss function and the current model estimate. Under 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,

$$\hat{f}^{(B)} = \hat{f}^{(0)} + \nu \sum_{b=1}^{B} \sum_{m=1}^{M} \boldsymbol{x}_i' \hat{\boldsymbol{\beta}}_m^{(b)} \mathrm{I}_{(\boldsymbol{s}_i \in \widehat{C}_m^{(b)})}$$

which is again a tree model. The boosting model is based on a more granular partition of the space expanded by $\boldsymbol{s}$. 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 $\hat{f}^{(0)}$. An alternative initial fit is $\hat{f}^{(0)} = \boldsymbol{x}_i'\hat{\boldsymbol{\beta}}$ with constant coefficient $\hat{\boldsymbol{\beta}}$ that minimizes the empirical risk (6). The final model $\hat{f}^{(B)}(\boldsymbol{x}_i, \boldsymbol{s}_i)$ can be represented as $\boldsymbol{x}_i'\hat{\boldsymbol{\beta}}(\boldsymbol{s}_i)$ 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 response variable.

Algorithm 2 Boosted "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 ν – 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, $\hat{f}^{(0)} = \arg\min_f \frac{1}{n}\sum_{i=1}^n \phi(y_i, f(\boldsymbol{x}_i, \boldsymbol{s}_i))$, where $f(\boldsymbol{x}_i, \boldsymbol{s}_i) = \boldsymbol{x}_i'\boldsymbol{\beta}$.

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

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

   (b) Fit u_i on $\boldsymbol{s}_i$ and $\boldsymbol{x}_i$ using the iterative "PartReg" algorithm (Algorithm 1) to obtain
   $$\widehat{u}_i = \sum_{m=1}^{M} \boldsymbol{x}_i'\hat{\boldsymbol{\beta}}_m^{(b)} \mathrm{I}_{(\boldsymbol{s}_i \in \widehat{C}_m^{(b)})}.$$

   (c) Update the fitted regression function by $\hat{f}^{(b)} = \hat{f}^{(b-1)} + \nu\sum_{m=1}^M \boldsymbol{x}_i'\hat{\boldsymbol{\beta}}_m^{(b)} \mathrm{I}_{(\boldsymbol{s}_i \in \widehat{C}_m^{(b)})}$.

3. Output the fitted model $\hat{f} = \hat{f}^{(B)}$.

The boosting algorithm involves four tuning parameters: the number of boosting iterations B, the learning rate ν, 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 ν lead to superior predictive performance on test data (reference). The parameters ν and B heavily depend on each other, and having a smaller value of ν often requires a greater number of iterations to obtain comparable out-of-sample prediction power. The learning rate ν 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. A tree with M terminal nodes allows up to order $M-1$ interactions. In our numerical studies, we fix all the tuning parameters except the number of boosting iterations, and use the genearlization error to choose this tuning parameter.

Our Algorithm 2 is readily generalizable to the estimation of nonlinear and generalized linear models, under a wide variety of loss functions. One can use more robust loss functions like the absolute error loss or Huber-loss (Bühlmann and Hothorn 2007), or use likelihood as the optimization criterion. We briefly describe the extension to generalized linear model here. Similar to Fan and Zhang (2008), we denote the canonical link function as $g(\cdot)$, and log conditional density as $l(\cdot)$. We seek to estimate the coefficient surface as follows

$$\widehat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} -2\sum_{i=1}^{n} l(y_i, g^{-1}(\boldsymbol{x}_i'\boldsymbol{\beta}(\boldsymbol{s}_i))).$$

In the notation of (6), the model fit is $f(\boldsymbol{x}_i, \boldsymbol{s}_i) = g^{-1}(\boldsymbol{x}_i'\boldsymbol{\beta}(\boldsymbol{s}_i))$ and the constrained function space is $\mathcal{F}_n = \{f(\boldsymbol{x}, \boldsymbol{s}) \mid f(\boldsymbol{x}, \boldsymbol{s}) = g^{-1}(\boldsymbol{x}'\boldsymbol{\beta}(\boldsymbol{s}))\}$. Commonly used generalized linear models including logistic regression and Poisson regression can be rephrased this way. Then we can apply Algorithm 2 to estimate the semiparametric generalized linear model with boosted varying-coefficient trees.

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 $\boldsymbol{x}_i$ are often key predictor variables thus it is not necessary to test the importance of variables in $\boldsymbol{x}_i$. 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 amounts to the reduction in the sum of squared error contributed by the splitting variable under L_2-loss. Here, we first define the importance of variable s_j in a single tree T as,

$$\mathcal{I}_j^2(T) = \sum_{l=1}^{M-1} \Delta\text{SSE}_l \ \text{I}_{(v(l)=s_j)},$$

where $v(l)$ denotes the variable chosen for splitting in the l-th step, and the improvement in model fit, ΔSSE_l, is defined in Algorithm 1. Thus the importance of the j-th variable s_j is defined as the sum of ΔSSE_ls when the variable is chosen for splitting the tree. In boosting, we denote the b-th tree as T_b, and compute the relative importance of s_j as the average of $\mathcal{I}_j^2(T_b)$ among all iterations, namely,

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

We can then rescale the $\mathcal{I}_j^2$s so that the largest importance index is 100, and then represent the variable importance metrics via barplots as in Figure 8.

In partial dependence plot, we split the predictors into two mutually exclusive sets, one set containing variables whose effects 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 $\boldsymbol{x}$, and how this relationship varies at different values of $\boldsymbol{s}$. For ease of presentation, we vary a single partition variable s_j at a time. It is challenging to depict how the coefficient surface changes with s_j, since this depends on what values we condition on for the remaining partition variables. Let $\boldsymbol{s}_{i,(-j)}$ denote the subvector of $\boldsymbol{s}_i$ with the j-th element deleted. We define

$$\bar{\boldsymbol{\beta}}(s_j) = \frac{1}{n}\sum_{i=1}^{n} \widehat{\boldsymbol{\beta}}(s_j, \boldsymbol{s}_{i,(-j)}),$$

which is the expectation of the estimated coefficient surface over the empirical distribution of $\boldsymbol{s}_{i,(-j)}$. The marginal prediction function remains to be a linear function of $\boldsymbol{x}$ with coefficients $\bar{\boldsymbol{\beta}}(s_j)$. We can specify different values of s_j, and examine how $\boldsymbol{x}'\bar{\boldsymbol{\beta}}(s_j)$ changes as a function of $\boldsymbol{x}$ when we vary s_j.

Another interest is to fit the mean of the coefficient surface, and obtain the partial dependence of y on $\boldsymbol{x}$, with all partition variables being averaged over. The resulting partial dependence is linear with coefficients defined as,

$$\bar{\boldsymbol{\beta}} = \frac{1}{n}\sum_{i=1}^{n} \widehat{\boldsymbol{\beta}}(\boldsymbol{s}_i).$$

This partial dependence function is generally different than the linear model estimates. The function $\boldsymbol{x}'\bar{\boldsymbol{\beta}}$ is theoretically equivalent to the ordinary linear regression function if squared error loss is used and the regression variables are independent of the partition variables. When $\boldsymbol{x}_i$ and $\boldsymbol{s}_i$ are dependent, $\boldsymbol{x}'\bar{\boldsymbol{\beta}}$ can be quite different than ordinary regression since $\boldsymbol{x}'\bar{\boldsymbol{\beta}}$ computes the expectation of $\hat{f}(\boldsymbol{x}, \boldsymbol{s}_i)$ over the empirical distribution of $\boldsymbol{s}_i$, not the conditional distribution $[\boldsymbol{s}_i \mid \boldsymbol{x}]$.

5 Simulations

In this section, we present simulation results on the performance of both tree and boosting varying-coefficient models. We are interested in comparing the predictive performance of the two models, as well as the efficiency of reconstructing varying-coefficient functions. We consider varying-coefficient simple linear regression, where the intercept and slope are both functions of a 10-dim vector $\boldsymbol{s}$. The model is $y_i = \beta_0(\boldsymbol{s}_i) + \beta_1(\boldsymbol{s}_i)x_i + \epsilon_i,\ i = 1, \cdots, n$, where x_i follows a standard normal distribution, and $\boldsymbol{s}_i$s are random samples from a uniform distribution on $[0, 1]^{10}$. The model errors ϵ_is follow a normal distribution with mean zero and variance 0.5^2. The intercept and slope are both additive functions with $\beta_0(\boldsymbol{s}_i) = 2\sin^2(2\pi s_{1i}) + \exp(2s_{2i} - 1)$, and $\beta_1(\boldsymbol{s}_i) = 2\cos^2(2\pi s_{1i}) + 8s_{2i}(1 - s_{2i})$, which only depend on s_1 and s_2 for sparsity. The sample size is $2,000$, 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 boosted 4-node trees with a regularization parameter of $\nu = 0.01$. 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 choose M by examining the risk measure on a test sample. Let $(\boldsymbol{s}_i', \boldsymbol{x}_i', y_i), i = n+1, \cdots, N$ denote the test observations, and $(\hat{\boldsymbol{\beta}}_m, \widehat{C}_m)$ denote the estimated regression coefficients and partitions based on training sample and $\mathcal{M}$ denote the range of tree sizes. The number of terminal nodes M is chosen by minimizing the out-of-sample prediction error,

$$\widehat{M} = \underset{M \in \mathcal{M}}{\arg\min} \sum_{i=n+1}^{N} \left(y_i - \sum_{m=1}^{M} \boldsymbol{x}_i' \hat{\boldsymbol{\beta}}_m \mathrm{I}_{(\boldsymbol{s}_i \in \widehat{C}_m)} \right)^2 . \tag{7}$$

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 to the true residual variance against the tuning parameter. The L_2-risk for the tree becomes constant when M is large enough, 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 has been alleviated by specifying the minimum node size; and boosting does not have serious issue of overfitting. Overall, the optimal predictive performance of boosting is better than that of the tree-based method, which is true for a number of other simulation scenarios as well. For both methods, we automate the selection of tuning parameters by choosing the tuning parameter that minimizes the L_2-risk on test sample.

To examine the estimation of β_0 and β_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 β_0 and β_1 as marginal functions of s_1 and s_2, where we have taken expectations with respect to $\boldsymbol{s}_{(-1)}$ and $\boldsymbol{s}_{(-2)}$. The estimated curves based on either tree-based or boosting varying-coefficient regression are plotted alongside as dashed

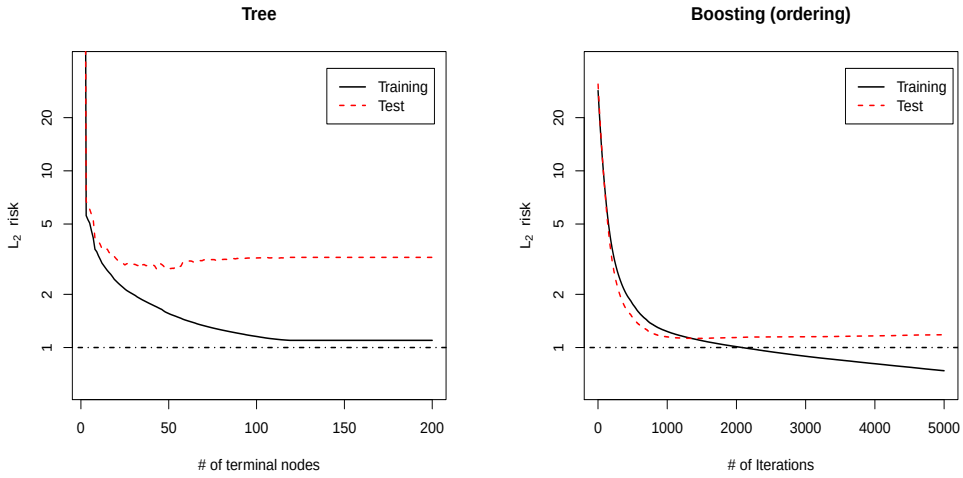

Figure 2: L_2 risk for tree-based and boosting varying-coefficient models with default tree growing algorithm; the solid line represents the L_2-risk on training data and the dashed line represents the L_2-risk on test data; the vertical axis shows the ratio of the mean squared error to the true error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

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.

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 records 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, and includes several leading mobile com-

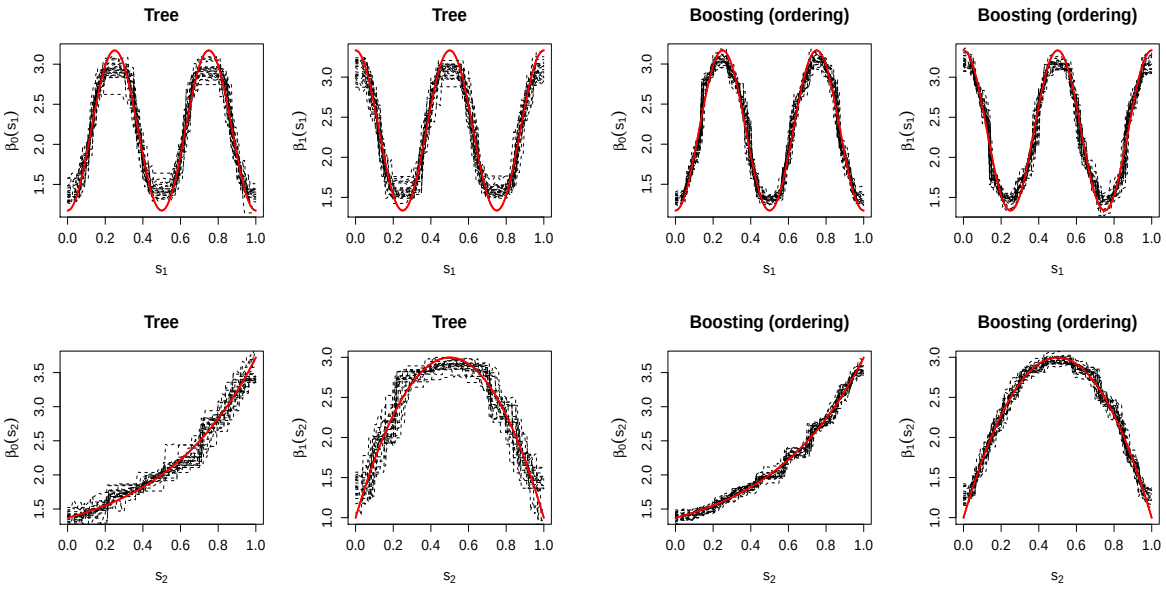

Figure 3: Reconstructed varying-coefficient surfaces by tree and boosting; the left panel shows the tree estimates and right panel shows the boosting estimates; 20 simulations are conducted under each method; the solid curves show the true functions and the dashed curves show our estimates.

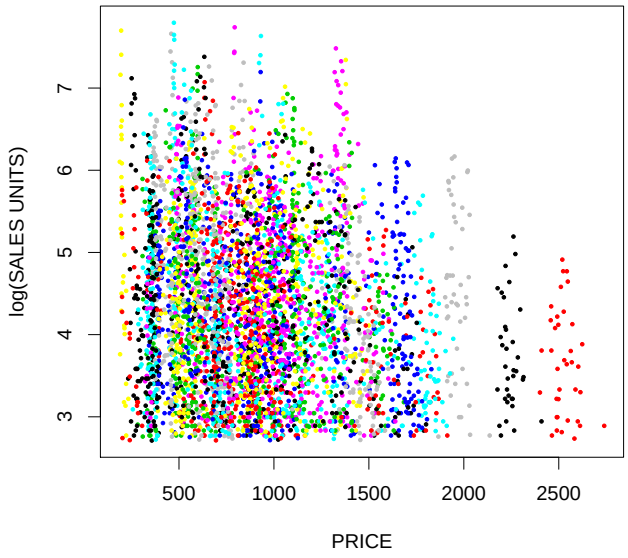

Figure 4: The log-transformed sales units against the average selling price. Various product configurations are represented by different colors and have shown to have varying demand functions, which motivates the use of varying-coefficient models.

puter brands in Austalia. Each row of the dataset contains detailed configurations of the respective product, the sales volume, and 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 volume. We have preprocessed the data and screened the products that sell very few. The data screening leaves us with 231 distinct product configurations and more than 5,500 observations.

The product prices range from nearly 200 to over 2500 U.S. dollars. We observe that the marginal distribution of the untransformed sales is highly skewed while the distribution of the log-transformed sales is more symmetric. Thus we use the log-transformed variable as our modeling target. The log-transformed sales units are plotted against price in Figure 4. Let y_i denote the sales volume, x_i denote the average selling price and $\boldsymbol{s}_i$ denote the set of eighteen varying-coefficient variables, with categorical variables including state, sales

channel, brand, operating system, GPU model and so on, and numerical variables including month and numerical PC attributes. The varying-coefficient model is

$$\log(y_i) = \beta_0(\boldsymbol{s}_i) + \beta_1(\boldsymbol{s}_i)x_i + \epsilon_i, \tag{8}$$

which implies that the price elasticity is proportional to price for a fixed product in a month, state and sales channel. An alternative model that uses the logarithm of price, which implies constant elasticity, has also been considered, and is commented later. At the kind suggestion of a referee, we have fitted an ordinary linear regression model as benchmark, which takes the varying-coefficient variables and price as linear predictors without interaction. The average L_2 risk on training and test sample is 0.686 and 0.708 respectively, based on a five-fold cross-validation. Five-fold cross-validation is also implemented on the various semiparametric models for comparison.

The varying-coefficient linear model (8) is estimated via tree-based method and boosting. 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 mean squared error on a test sample. The L_2 risk 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.

Competing methods

To better assess the performance of the proposed methods, we have fitted model (8) using two existing methods: *elastic net* based penalized linear regression (Zou and Hastie 2005) and kernel smoothing (Fan and Gijbels 2003).

Elastic net varying-coefficient linear model

In elastic net model, we first create dummy variables based on categorical variables, and then generate design matrix Z by including both the quadratic effect of individual variables

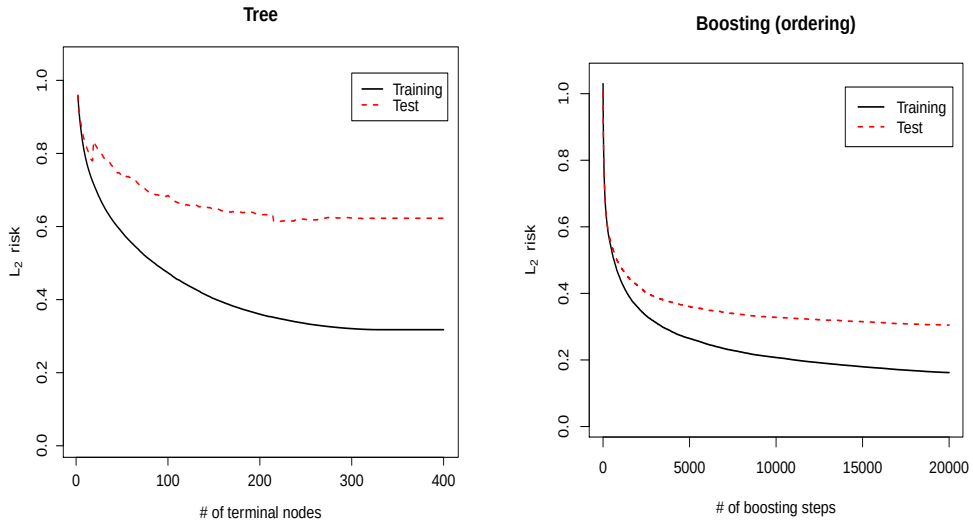

Figure 5: L_2 risk for tree-based varying-coefficient model and boosting varying-coefficient model; the left panel shows the results based on a single tree and the right panel shows boosting that orders the categories in growing each base tree; the solid and dashed lines represent the L_2 risk on training and test data, respectively.

and first-order interaction effect between pairs of variables. We denote the i-th row of matrix Z as $\boldsymbol{z}_i'$, and then specify $\beta_0(\boldsymbol{s}_i)$ as $\boldsymbol{z}_i'\boldsymbol{\beta}_0$ and $\beta_1(\boldsymbol{s}_i)$ as $\boldsymbol{z}_i'\boldsymbol{\beta}_1$. Finally, we solve the following penalized least squares problem where the penalty on the regression coefficients is a convex combination of L_1 and L_2 penalty:

$$(\widehat{\boldsymbol{\beta}}_0, \widehat{\boldsymbol{\beta}}_1) = \underset{(\boldsymbol{\beta}_0, \boldsymbol{\beta}_1)}{\arg\min} \sum_{i=1}^{n} \{y_i - \boldsymbol{z}_i'\boldsymbol{\beta}_0 - (\boldsymbol{z}_i' x_i)\boldsymbol{\beta}_1\}^2 + \lambda \left\{ \alpha \sum_{i,j} |\beta_{ij}| + \frac{(1-\alpha)}{2} \sum_{i,j} \beta_{ij}^2 \right\}.$$

Figure 6 shows the training and test sample L_2 risk against the logarithm of the tuning parameter λ. We have chosen two different values of α, where $\alpha = 1$ reduces to LASSO regression (Tibshirani 1996), and $\alpha = 0.5$ is the default value in the `R` package `glmnet`. The numerical results show that the test sample L_2 risk is minimized when λ is close to zero under either value of α, which suggests that overfitting is not a serious concern in the first-order interaction model. The optimal out-sample L_2 risk is 0.48 under both values of α, and inferior to boosting (out-sample L_2 risk=0.30).

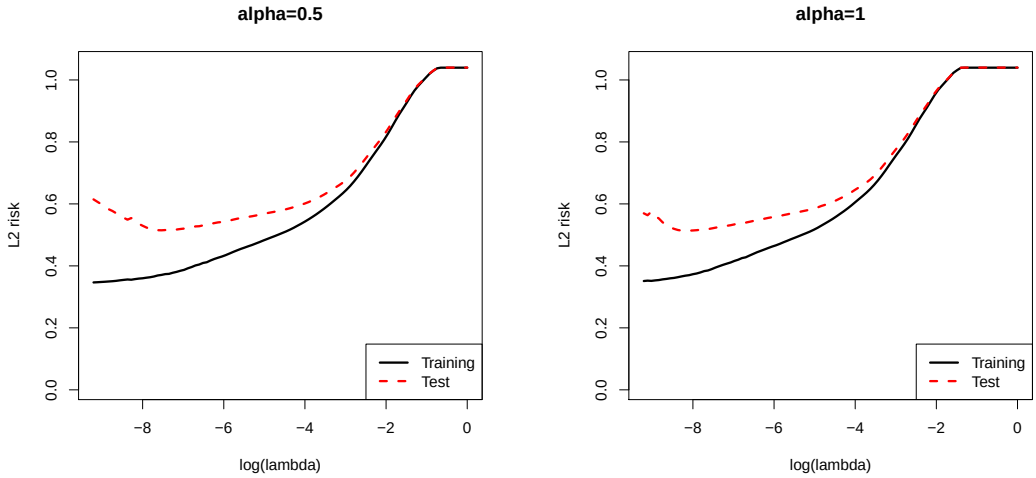

Figure 6: L_2 risk for elastic net based varying-coefficient model with $\alpha = 0.5$ (left panel) and 1 (right panel). The solid and dashed lines represent the L_2 risk on training and test data, respectively.

Kernel regression-based varying-coefficient linear model

Kernel regression is often used for fitting varying-coefficient models, and, in this application, is applied to model (8) for comparison. We have implemented a locally constant smoothing estimator. To estimate the coefficient function at location $\boldsymbol{u}$, we specify kernel functions and smoothing parameters on both categorical and continuous variables, calculate kernel weights and then seek to solve the weighted least squares problem,

$$\widehat{\boldsymbol{\beta}}(\boldsymbol{u}) = \arg\min_{\boldsymbol{\beta}} L(\boldsymbol{\beta}) = \arg\min_{\boldsymbol{\beta}} \sum_{i=1}^{n} (y_i - \boldsymbol{x}_i'\boldsymbol{\beta})^2 K\left(\boldsymbol{u}; \boldsymbol{s}_i\right),$$

where $K\left(\boldsymbol{u}; \boldsymbol{s}_i\right)$ denotes the kernel weight of observation i when predicting at $\boldsymbol{u}$. The multi-dimensional kernel weight, which involves defining kernel functions on a mixed type of variables, is defined as

$$K\left(\boldsymbol{u}; \boldsymbol{s}_i\right) = \prod_{j \in \Omega_N} \frac{1}{h} K\left(\frac{s_{ij} - u_j}{h}\right) \prod_{j \in \Omega_C} \left\{ \lambda_j \mathrm{I}_{(s_{ij} = u_j)} + \frac{1 - \lambda_j}{c_j - 1} \mathrm{I}_{(s_{ij} \neq u_j)} \right\}, \tag{9}$$

with $\lambda_j = \alpha + (1 - \alpha)/c_j$, $\alpha \in [0, 1]$ and c_j is the number of levels for variable j. In (9), Ω_N denotes the indices of numerical variables and Ω_C denotes the collection of categorical

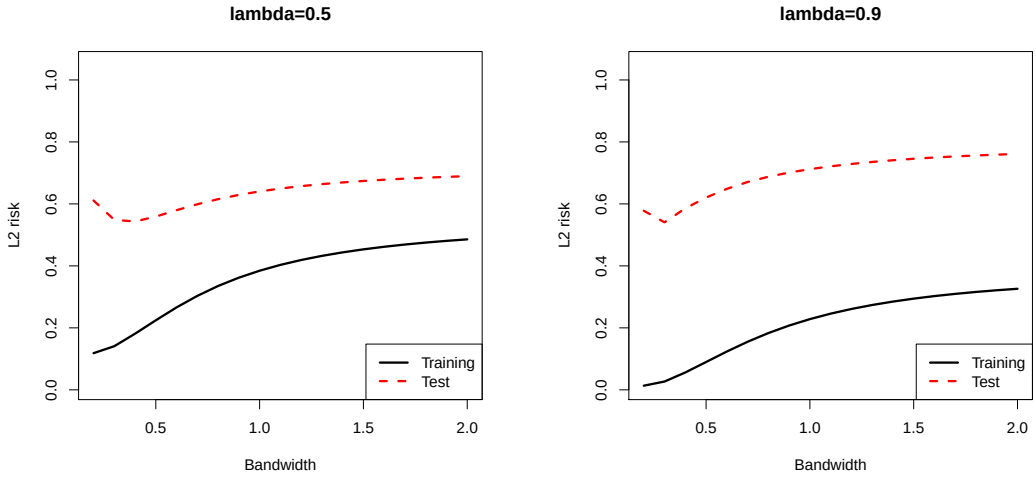

Figure 7: L_2 risk for kernel smoothing based varying-coefficient model on training (left panel) and test (right panel) sample. The solid and dashed lines represent the L_2 risk on training and test data, respectively.

variables. The continuous variables have been standardized a priori, and a single bandwidth is applied to all continuous variables for parsimony. The kernel on categorical variables depends on whether the two factor levels are the same or not (Aitchison and Aitken 1976). The parameter λ_j is a convex combination of 1 and $1/c_j$, where 1 corresponds to no contribution from unmatched categories and $1/c_j$ means equal contribution from all categories. The parameters h and α are tuning parameters that control the amount of smoothing on continuous and categorical variables, respectively. In figure (7), we plot the L_2 risk on training and test sample against the bandwidth, under $\alpha = 0.5$ and $\alpha = 0.9$. The case with $\alpha = 1$ is numerically unstable, especially for categories with small sample size, thus has been replaced by $\alpha = 0.9$. The optimal out-sample L_2 risk that can be achieved by kernel smoothing is 0.54, which is clearly outperformed by boosting.

The varying-coefficient model (8) takes price as a linear predictor. Variations to this model have also been attempted, including hedonic regression and log-transformation of the

price. Hedonic regression is widely used in econometrics, and can be used to remove the strong collinearity between product features and price. The second option, namely using the logarithm of price in place of original price, implies constant elasticity for a product sold through a fixed channel in a certain month and region. We have tried both variations, but neither variation leads to much superior results. As a consequence, we will not present the numerical results from these two variations.

Figure 8 provides a graphical display of variable importance under the boosting model. The variables with an importance measure less than 10 are suppressed from this plot. Boosting, as a forward stagewise selection approach, selects all three environmental variables including the sales channels (`REPORT`), state (`REG`) and month (`MONTH`). Variables like the brand (`BRAND`), clock speed (`CSP.MH`), GPU model (`GPU.MO`) and processor (`PROCESS`) are also important in determining the price-demand relationship. Figure 9 shows the partial dependence plot on brand. In the context of varying-coefficient models, the key interest is how the regression function changes 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 then uses boosting to improve the predictive performance as well as smoothing the regression coefficient. 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

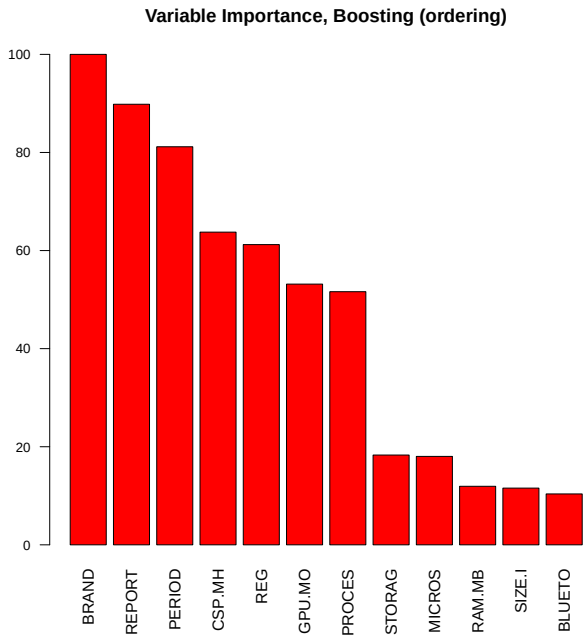

Figure 8: The variable importance plot from the boosting algorithm.

to varying-coefficient models, and at the same time, can be regarded as a structured learning method that has realistic implications.

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

Aitchison, J. and C. G. G. Aitken (1976). Multivariate binary discrimination by the kernel method. *Biometrika 13*, 301–320.

Bartlett, P. and M. Traskin (2007). AdaBoost is consistent. In B. Schölkopf, J. Platt, and T. Hoffman (Eds.), *Advances in Neural Information Processing Systems 19*, pp.

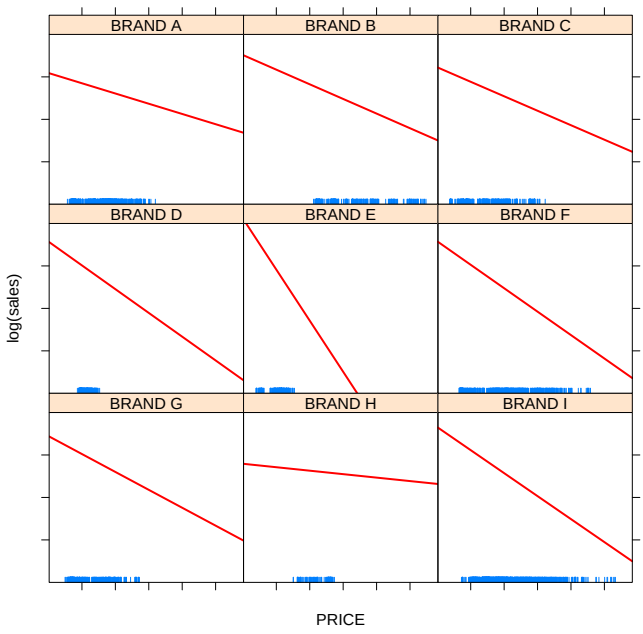

Figure 9: The partial dependence plot of demand on product price and brand. The actual laptop brands are suppressed and denoted by alphabetical letters. The figure shows how the volume (level or intercept) and sensitivity (slope) of the demand vary between different brands.

105–112. MIT Press, Cambridge, MA.

Breiman, L. (1998). Arcing classifiers (with discussion). *Annals of Statistics 26*, 801–849.

Breiman, L. (1999). Prediction games and arcing algorithms. *Neural Computation 11*, 1493–1517.

Breiman, L., J. Friedman, R. Olshen, and C. Stone (1984). *Classification and Regression Trees*. Wadsworth, New York.

Bühlmann, P. (2006). Boosting for high-dimensional linear models. *Journal of the American Statistical Association 34*, 559–583.

Bühlmann, P. and T. Hothorn (2007). Boosting algorithms: regularization, prediction and model fitting (with discussion). *Statistical Science 22*, 477–505.

Bühlmann, P. and B. Yu (2003). Boosting with the l_2 loss: regression and classification. *Journal of the American Statistical Association 2298*, 324–339.

Chipman, H., E. George, and R. McCulloch (1998). Bayesian CART model search. *Journal of the American Statistical Association 93*, 935–960.

Chipman, H., E. I. George, and R. E. McCulloch (2002). Bayesian treed models. *Machine Learning 48*(1), 299–320.

Cleveland, W., E. Grosse, and W. Shyu (1991). Local regression models. In J. M. Chambers and T. J. Hastie (Eds.), *Statistical Models in S*. Wadsworth & Brooks; Pacific Grove.

Denison, D. G. T., B. K. Mallick, and A. F. M. Smith (1998). A Bayesian CART algorithm. *Biometrika 85*, 363–377.

Fan, G. and J. B. Gray (2005). Regression tree analysis using TARGET. *Journal of Computational and Graphical Statistics 14*(1), 1–13.

Fan, J. and I. Gijbels (2003). *Local Polynomial Modelling and its Applications*. London: Chapman and Hall.

Fan, J. and W. Zhang (2008). Statistical methods with varying coefficient models. *Statistics and its interface 1*, 179–195.

Fisher, W. (1958). On grouping for maximal homogeniety. *Journal of the American Statistical Association 53*(284), 789–798.

Freund, Y. and R. Schapire (1995). A decision-theoretical generalization of online learning and an application to boosting. In *Proceedings of teh Second European Conference on Computational Learning Theory*. Lecture Notes in Computer Science, Springer.

Freund, Y. and R. Schapire (1996). Experiments with a new boosting algorithm. In *Proceedings of the Thirteenth International Conference on Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA.

Freund, Y. and R. Schapire (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences 55*, 119–139.

Friedman, J. H. (2001). Greedy function approximation: a gradient boosting algorithm. *Annals of Statistics 29*, 1189–1232.

Goldberg, D. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading, MA: Addison-Wesley.

Hastie, T., R. Tibshirani, and J. Friedman (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer-Verlag, New York.

Hastie, T. and R. J. Tibshirani (1993). Varying-coefficient models. *Journal of the Royal Statistical Society, Series B 55*, 757–796.

Jiang, W. (2004). Process consistency for AdaBoost. *Annals of Statistics 32*(1), 13–29.

Logosi, G. and N. Vayatis (2004). On the Bayes-risk consistency of regularized boosting methods. *Annals of Statistics 32*(1), 30–55.

Loh, W.-Y. (1997). Regression trees with unbiased variable selection and interaction detection. *Statistica Sinica 12*, 361–386.

Loh, W.-Y. and H. Kim (2001). Boosting classification trees with unbiased multiway splits. *Journal of the American Statistical Association 96*, 589–604.

Loh, W.-Y. and Y.-S. Shih (1997). Split selection methods for classification trees. *Statistica Sinica 7*, 815–840.

McCullagh, P. and J. A. Nelder (1999). *Generalized Linear Models*. Chapman & Hall Ltd.

McFadden, D. (1974). Conditional logit analysis of qualitative choice behavior. In *Frontiers in Econometrics, ed. P. Zarembka*, pp. 105–142. Academic Press, New York.

Morgan, J. N. and J. A. Sonquist (1963). Problems in the analysis of survey data, and a proposal. *Journal of the American Statistical Association 58*, 415–434.

Papagelis, A. and D. Kalles (2001). Breeding decision trees using evolutionary techniques. In *Conference Proceedings ICML '01*. Williamstown, USA.

Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufman, San Mateo.

Tibshirani (1996). Regression shrinkage and selection via LASSO. *Journal of the Royal Statistical Society, Series B 58*, 267–288.

Zhang, T. and B. Yu (2005). Boosting with early stopping: convergence and consistency. *Annals of Statistics 33*, 1538–1579.

Zou, H. and T. Hastie (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society, Series B 67*(2), 301–320.