

Segmented Regression Estimators for Massive Data Sets

*Ramesh Natarajan, Edwin Pednault**

Abstract

We describe two methodologies for obtaining segmented regression estimators from massive training data sets. The first methodology, called Linear Regression Tree (LRT), is used for continuous response variables, and the second and complementary methodology, called Naive Bayes Tree (NBT), is used for categorical response variables. These are implemented in the IBM ProbETM (Probabilistic Estimation) data mining engine, which is an object-oriented framework for building classes of segmented predictive models from massive training data sets. Based on this methodology, an application called ATM-SETM for direct-mail targeted marketing has been developed jointly with Fingerhut Business Intelligence [1]).

Keywords

Segmentation-based models, decision trees, linear regression, logistic regression, feature selection, Naive Bayes.

1 Introduction

Fingerhut Companies Inc. is a leading database marketing company with considerable experience in the use of statistical methods for consumer modeling and targeted marketing. In one of their applications, predictive models constructed from historical customer spending, demographic and psychographic data are used to score and rank potential customers for new marketing promotions. The construction of the predictive models, however, presents challenges that are not well addressed by

*IBM T.J. Watson Research Center, Yorktown Heights, NY 10598

current statistical techniques. First, the relevant marketing databases contain massive amounts of training data with millions of customer records and thousands of potential modeling attributes per customer record. Second, the data is complex and heterogeneous with many sub-populations, and within each sub-population the response variable has a distinctive structural dependence on the covariates.

In Fingerhut’s existing modeling approach, the large data set was sub-sampled, and preliminary feature selection and data transformations were performed to obtain a reduced data set for which the relevant statistical computations are tractable. Also, appropriate customer segments or sub-populations were selected *a priori* (perhaps by drawing on prior modeling experience with the data), and linear or logistic regression models were constructed in the individual segments. Fingerhut’s analysts were aware that this overall methodology could be “sub-optimal”, due to the model variability from the data sub-sampling, and due to the *ad hoc* decoupling of the feature selection and segmentation steps from the subsequent predictive modeling computations. The need for a methodology that would (a) be computationally tractable for massive training data sets without pre-processing, and (b) jointly optimize the feature selection, segmentation and modeling steps, was the impetus for the development of IBM ATM-SETM solution (Advanced Targeted Marketing - Single Events) for direct mail retailing [1].

The two new predictive modeling methodologies on which ATM-SE is based are Linear Regression Tree (LRT) for continuous response variables, and the complementary methodology called Naive Bayes Tree (NBT) for categorical response variables. These are implemented in IBM ProbETM (for Probabilistic Estimation) data mining engine, which is an object-oriented framework for building segmented predictive models from massive, out-of-core training data sets. In the ATM-SE application for direct-mail marketing, NBT was used to predict the customer response probability, and LRT was used to predict the customer gross sales in dollars for a given catalog mailing. A benefits analysis of ATM-SE by Fingerhut [1] showed that even when used “out-of-the-box”, the accuracy of the resulting models were comparable or better than their own benchmark models (see also Section (6)).

The basic segmentation approach is based on decision tree algorithms (e.g., CART [6], C4.5 [21]), especially the developments in [17], [11], [12] where more elaborate probability models are used in the tree leaf nodes. A detailed qualitative comparison with this previous work is given in sections (4.3) and (5.3), and experimental results are given in Section (6). The primary difference is that our computational emphasis is not on the typically small, homogenous benchmark data sets in the statistical and machine learning literature ([4], [5]), but on massive (out-of-core), heterogeneous training data sets (treated without sub-sampling) that are appearing in many emerging data mining applications.

2 Background

Let y denote the response variable, which depending on the context may either be a real-valued continuous variable, or a categorical response variable taking on K values denoted by $1, 2, \dots, K$, respectively. Let the set of J covariate predictors for

y be denoted by X_1, X_2, \dots, X_J respectively, and denote the corresponding vector of covariates by

$$\mathbf{X} = [X_1 \quad X_2 \quad \dots \quad X_J]^T. \quad (1)$$

Let $\theta_y(\mathbf{X}) = P(y|\mathbf{X})$ denote the conditional response probability, and $\{\mathbf{x}_i, y_i\}_{i=1}^N$ denote the training data, where $\mathbf{x}_i = [x_{1,i} \quad x_{2,i} \quad \dots \quad x_{J,i}]^T$, with $x_{j,i}$ being the value of the j 'th covariate in the i 'th training data record. We consider the empirical expected negative log-likelihood of this training data given by

$$\mathcal{L}_{TR} = -\frac{1}{N} \sum_{i=1}^N \log \theta_{y_i}(\mathbf{x}_i). \quad (2)$$

Most modeling approaches obtain estimates $\hat{\theta}_k(\mathbf{X})$ for $\theta_k(\mathbf{X})$ by minimizing \mathcal{L}_{TR} in one way or the other.

For example, in decision tree algorithms (e.g., CART [6], C4.5 [21]), this minimization is performed by recursively partitioning the input space of the covariates into mutually-exclusive, collectively-exhaustive segments, with a simple conditional response probability model within each segment.

Specifically, for continuous response y in segment t , decision tree algorithms typically assume the Gaussian model $\theta_y(\mathbf{x}, t) \sim \mathcal{N}(y(t), \sigma(t)^2)$, using the estimates $\hat{y}(t) = \frac{1}{N(t)} \sum_{i=1}^{N(t)} y_i$ for $y(t)$ and $\hat{\sigma}^2(t) = \frac{1}{N(t)-1} \sum_{i=1}^{N(t)} (y_i - \hat{y}(t))^2$ for $\sigma(t)$ (these are the response sample mean and variance from the $N(t)$ training data points in the segment). The resulting empirical negative log-likelihood $\mathcal{L}_{TR}(t)$ in segment t is given by

$$\mathcal{L}_{TR}(t) = \frac{1}{2} \left[\log 2\pi\hat{\sigma}^2(t) + \frac{N(t)-1}{N(t)} \right]. \quad (3)$$

Similarly, for categorical response y , if $N_k(t)$ denotes the number of training data points in segment t having response k (note that $\sum_{k=1}^K N_k(t) = N(t)$), $\theta_k(\mathbf{x}, t)$ is given by the multinomial probability estimate $\pi_k(t) = (N_k(t) + \lambda_k)/(N(t) + \sum_{k'=1}^K \lambda_{k'})$. Here the frequency count estimates have been smoothed to avoid overfitting with small sample sizes $N(t)$, and to avoid potential singularities due to zero counts when using this model to evaluate the empirical log-likelihood function for unseen data (the smoothing parameters λ_k can be regarded as the parameters of a Bayesian Dirichlet conjugate prior for the multinomial probability [2], with the special case $\lambda_k = 1$ being the well-known Laplace smoothing criterion). With this model for $\pi_k(t)$, we then have

$$\mathcal{L}_{TR}(t) = -\sum_{k=1}^K \frac{N_k(t)}{N(t)} \log(\pi_k(t)). \quad (4)$$

The recursive partitioning procedure for the minimization of \mathcal{L}_{TR} in (2) is performed by starting from a single segment t which covers all the data, and examining the set of all possible axis-parallel binary splits in the range of each covariate, to obtain the binary split that leads to two child segments t_L and t_R with the maximum decrease in the quantity

$$\Delta\mathcal{L} = [N(t_L)\mathcal{L}_{TR}(t_L) + N(t_R)\mathcal{L}_{TR}(t_R)] - N(t)\mathcal{L}_{TR}(t). \quad (5)$$

Here $N(t_L)+N(t_R) = N(t)$, and the last term in (5) is invariant for all $\{t_L, t_R\}$ pairs. This splitting procedure is then applied recursively in each resulting child segment, until terminated by some stopping condition. This partitioning leads to a binary tree structure \mathcal{T}_F with F leaf segments denoted by $\{t_1, t_2, \dots, t_F\}$ respectively, and the overall response conditional probability is then given by

$$\hat{\theta}_k(\mathbf{x}) = \sum_{f=1}^F \pi_k(t_f) \mathcal{I}_{[t_f]}(\mathbf{x}), \quad (6)$$

where $\mathcal{I}_{[t_f]}(\mathbf{x}) = 1$ if $x \in t_f$ or 0 otherwise. Various pruning and model selection procedures based on test-set error or cross-validation error are described in [6] and [21] for obtaining the “right-sized” decision tree model with the best predictive accuracy.

One limitation of the original decision tree approach is the simplicity of the segment probability models described above. Particularly, for heterogenous data sets, the resulting models are overly-complex with poor model interpretability and predictive accuracy.

3 ProbE Computational Model

3.1 Overview

ProbE is an object-oriented framework specifically designed for segmented predictive modeling with massive training data sets. In ProbE, the overall model is defined by a collection of rules of the form

$$\text{”IF } \langle \textit{rule_predicate} \rangle \text{ THEN } \langle \textit{rule_model} \rangle \text{”}. \quad (7)$$

In particular, ProbE can generate tree-structured models where the *rule_predicate* is a conjunction of univariate binary split conditions on the range of values of the individual explanatory features. These binary splits are of the form $x_i \leq c$ or $x_i > c$ for c in the range of an ordered explanatory feature x_i , or of the form $x_j \in \text{subset}\{c_1, c_2, \dots, c_m\}$ for a categorical explanatory feature x_j with m denoted categories. The set of all *rule_predicates* partitions the space of explanatory features into a set of mutually-exclusive, collectively-exhaustive segments. Each *rule_predicate* defines a segment, with the associated *rule_model* being a response probability model with parameters estimated from the training data in the segment.

Operationally, in the training mode, ProbE generates an overall model by recursively applying a “model expansion” procedure to the set of rules in the existing model. This model expansion step involves two distinct but complementary mechanisms. The first mechanism is a “rule split” which involves a specific univariate binary split of an existing rule into two descendant rules. The second mechanism is a “model extension” which involves adding a single covariate to the response probability model in the consequent of an existing rule. These two complementary mechanisms, viz., rule splits and model extensions, are placed on the same footing in the model expansion process, which is a principled approach to improving

the training model accuracy by adding new overall degrees of freedom. This dual approach to model expansion is a novel feature of the ProbE framework.

As in Section (2), the best model expansion for an existing rule is evaluated by using the empirical negative log-likelihood (2) as the relevant scoring function for selecting the best rule split or model extension over all possible candidates.

The evaluation of the binary rule splits for each feature is carried out by first considering a set of multi-way splits on each feature (after partitioning the range of continuous features into several subintervals, and using each individual category for categorical features). Once the probability models for each multiway split of an existing rule have been trained, the best possible binary rule split on a given feature is found by a bottom-up binary merging procedure analogous to that used in CHAID [10]. Specifically, for each feature, two segments of this multi-way split are merged so that for the resulting segment this leads to the minimum increase in the scoring function. This binary merging procedure is recursively applied until only two segments remain, which then represents the best binary split for that feature in the existing rule using this bottom-up heuristic.

This model expansion procedure is recursively applied until terminated either by a user-specified “stopping condition” or by an internal cross-validation heuristic. The final “right-sized” model is a decision tree represented as a set of rules, where the *rule.predicates* are the conjunctions of the individual univariate binary splits starting from the root node to each leaf node, and the corresponding *rule.models* are the response probability models with parameters estimated from the relevant training data in that leaf node.

3.2 Implementation details and Performance Issues

ProbE is designed so that the massive out-of-core training data sets are accessed sequentially for I/O efficiency reasons. The I/O cost of each data scan is an important performance consideration, since several data scans are typically required to fully train a new ProbE model. Therefore, as shown below, it is important to be able to estimate the parameters of a segment probability models after a single scan of the relevant training data, as far as possible. Similarly, in order to examine the maximum number of individual segment probability models in each training data scan, it is also important to minimize the data-structure storage for each individual model.

The model expansion step described in Section (3.1) requires an enormous number of new segment probability models (of the order of the number of existing rules \times the number of features \times the number of multiway splits per feature) to be trained and evaluated. However, depending on the number of probability model objects that can be concurrently held in-core, more than one data scan may be required to complete each model expansion step. As the training data is scanned record by record, the sufficient statistics of these probability model objects are updated as relevant, and used to obtain their model parameters at the end of the training data scan. Subsequently, the bottom-up merging procedure is performed as described in Section (3.1) to find the best split in each rule.

The model expansion procedure used in ProbE imposes two important re-

quirements on the individual probability model objects. First, as the training data is scanned, each relevant probability model object being examined must be able to update the internal data structures for the “sufficient statistics” in a storage and computationally efficient way.

Second, it should be possible to combine the sufficient statistics of two or more model objects corresponding to different segments, in such a way that the parameters of the resulting probability model object can be obtained without having to rescan the training data. That is, the result should be the same as if the resulting model object were trained using the union of the original training data for the segments that are merged. This requirement enables the entire binary bottom-up merging procedure described earlier to be performed without having to rescan the training data.

The ProbE framework is valuable because these two conditions are satisfied for a large class of probability models, although in some case it raises interesting computational research issues. We now describe the detailed implementations of the LRT model in Section (4), and the NBT model in Section (5).

4 LRT Implementation Details

The LRT methodology is based on the use of the Gaussian probability model

$$\theta_y(\mathbf{X}, t) \sim \mathcal{N}(a_0 + \sum_j^J a_j(t)X_j, \sigma(t)^2), \quad (8)$$

in each segment t , with the regression parameters computed by the well-known normal equations method [7] (see p. 224), [3] (see p. 49). This method satisfies the two requirements specified at the end of Section (3), since the sufficient statistics (in this case the data means and covariances) can be obtained from a single pass over the training data. Furthermore, the sufficient statistics from two or more model objects can be combined to compute the regression parameters for the model in the resulting combined segment. Our implementation of the normal equations method incorporates feature selection in a way that regularizes the computations, and provides stable estimates of the non-zero regression coefficients.

The feature selection algorithm used in each probability model object is based on the use of hold-out data. Specifically, each relevant data record that is scanned is randomly designated as a training record (used to determine the order in which features are introduced in the regression model) or a hold-out record (used to determine the optimum subset size in this ordered set of features). Each probability model object has separate data structures for storing and updating the means and covariances of the relevant training or hold-out data records.

After the sufficient statistics are obtained from a training data scan, a forward stepwise feature selection procedure is used with the training covariance matrix. Subsequently the optimal subset size from this feature ordering is determined using the hold-out covariance matrix as shown below. Finally, the separate means and covariances for the training and hold-out data are combined, to obtain the final estimates for the regression parameters with the chosen feature subset.

In the binary merging step in ProbE, the means and covariances for the training and hold-out data of the two individual probability model objects are separately merged, and the feature selection and parameter estimation in the resulting model object are performed just as described earlier.

The details of each of these steps are discussed in the following subsections.

4.1 Online Updating and Merging of Mean and Covariance Estimates

Let $\xi = \{\mathbf{x}, y\}$ denote the training data record, where \mathbf{x} denotes the J explanatory features and y is the response. Typically, only the continuous explanatory features are included for the segment models, although categorical features may be incorporated by explicitly encoding them in the input data using dummy indicator variables in the usual way.

For a given probability model, let $\{\xi_i\}_{i=1}^M$ denote the previously scanned records. The mean vector $\boldsymbol{\mu}_M$ and covariance matrix \mathbf{S}_M are given by

$$\boldsymbol{\mu}_M = \frac{1}{M} \sum_{i=1}^M \xi_i, \quad \mathbf{S}_M = \sum_{i=1}^M (\xi_i - \boldsymbol{\mu}_M)(\xi_i - \boldsymbol{\mu}_M)^T. \quad (9)$$

For analytic convenience, the constant normalizing factor is always omitted in the covariance matrix.

When a new record ξ_{M+1} is added to the set, the following updates can be used to incrementally update the values $\boldsymbol{\mu}_M$ and \mathbf{S}_M in (9),

$$\begin{aligned} \boldsymbol{\mu}_{M+1} &= \frac{M\boldsymbol{\mu}_M + \xi_{M+1}}{M+1}, \\ \mathbf{S}_{M+1} &= \mathbf{S}_M + \frac{(M+1)}{M} (\boldsymbol{\mu}_{M+1} - \xi_{M+1})(\boldsymbol{\mu}_{M+1} - \xi_{M+1})^T. \end{aligned} \quad (10)$$

As noted earlier, each new record is randomly assigned to update either the training data means and covariances $(\boldsymbol{\mu}, \mathbf{S})$, or the hold-out data means and covariances $(\tilde{\boldsymbol{\mu}}, \tilde{\mathbf{S}})$.

Consider two models which respectively have M and N records at the end of a training scan. For the merged model with $M+N$ records, we have

$$\begin{aligned} \boldsymbol{\mu}_{M+N} &= \frac{M\boldsymbol{\mu}_M + N\boldsymbol{\mu}_N}{M+N}, \\ \mathbf{S}_{M+N} &= \mathbf{S}_M + \mathbf{S}_N + \frac{N(M+N)}{M} (\boldsymbol{\mu}_{M+N} - \boldsymbol{\mu}_N)(\boldsymbol{\mu}_{M+N} - \boldsymbol{\mu}_N)^T. \end{aligned} \quad (11)$$

The means and covariances of the training and hold-out sets are separately merged to produce the corresponding statistics for the merged model. The symmetric covariance matrices are stored in a packed format to save storage, and the rank-one updates in (10) and (11) are implemented using standard routines in the BLAS library [7].

4.2 Feature Selection and Linear Regression

The training covariances are computed with all variables centered about their sample means, so that the “intercept” term can be omitted from the linear regression model. Thus, with M training data records, we denote the $M \times (J + 1)$ centered data matrix as

$$\begin{bmatrix} \mathbf{X} & \mathbf{y} \end{bmatrix} = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \dots & \mathbf{x}_J & \mathbf{y} \end{bmatrix}. \quad (12)$$

This large data matrix is a notational convenience, and it is never explicitly stored in the model object. Consider the following partitioning of the $(J + 1) \times (J + 1)$ covariance matrix \mathbf{S} ,

$$\mathbf{S} = \begin{bmatrix} \mathbf{X}^T \mathbf{X} & \mathbf{X}^T \mathbf{y} \\ \mathbf{y}^T \mathbf{X} & \mathbf{y}^T \mathbf{y} \end{bmatrix}, \quad (13)$$

where $\mathbf{X}^T \mathbf{X}$ refers to the covariance of the predictors, $\mathbf{X}^T \mathbf{y}$ to the correlation of the predictors with the response, and $\mathbf{y}^T \mathbf{y}$ to the variance of the response about its mean. Then, the solution of the least squares problem

$$\arg \min_{\mathbf{a}} \|\mathbf{y} - \mathbf{X}\mathbf{a}\|^2, \quad (14)$$

where $\mathbf{a} = \{a_1, \dots, a_J\}$ are the regression coefficients, is obtained from the normal equations [3]

$$(\mathbf{X}^T \mathbf{X})\mathbf{a} = \mathbf{X}^T \mathbf{y}. \quad (15)$$

The Cholesky factorization of \mathbf{S} with the same partitioning as Equation (13), but with a specific symmetric row and column permutations to the leading J rows and columns of \mathbf{S} (i.e., excluding the last row and column which are always held fixed), is given by

$$\mathbf{S} = \begin{bmatrix} \mathbf{P}(\mathbf{X}^T \mathbf{X})\mathbf{P}^T & \mathbf{P}\mathbf{X}^T \mathbf{y} \\ \mathbf{y}^T \mathbf{X}\mathbf{P}^T & \mathbf{y}^T \mathbf{y} \end{bmatrix} = \begin{bmatrix} \mathbf{R}^T & \\ \mathbf{z}^T & \rho \end{bmatrix} \begin{bmatrix} \mathbf{R} & \mathbf{z} \\ & \rho \end{bmatrix}, \quad (16)$$

where \mathbf{P} is a row permutation matrix. The coefficient vector \mathbf{a} can then be obtained by solving the triangular system

$$\mathbf{a} = \mathbf{P}^T \mathbf{R}^{-1} \mathbf{z}, \quad (17)$$

and $\rho = \sqrt{\mathbf{y}^T \mathbf{y} - \mathbf{z}^T \mathbf{z}}$ is the corresponding minimum residual squared norm in (14). This implementation of the Cholesky factorization with symmetric permutation can be carried out in place with the matrix \mathbf{S} stored in packed format.

The sequence of symmetric permutations are chosen to include the most useful covariates for modeling the response, while excluding collinear features that degrade the numerical accuracy. Consider the Cholesky factorization as a sequence of in-place transformations of the matrix \mathbf{S} of the form

$$\mathbf{S} \equiv \mathbf{S}_{(0)} \longrightarrow \mathbf{P}_{(1)} \mathbf{S}_{(1)} \mathbf{P}_{(1)}^T \longrightarrow \dots \longrightarrow \mathbf{P}_{(n)} \mathbf{S}_{(n)} \mathbf{P}_{(n)}^T, \quad (18)$$

where $\mathbf{P}_{(i+1)} = \Pi_{(i+1)} \mathbf{P}_{(i)}$ are permutation matrices with $\Pi_{(i+1)}$ denoting a row permutation in the $(i + 1)$ row of the matrix, and with the i 'th transition above

only modifying the lower $(J + 1 - i) \times (J + 1 - i)$ submatrix. For example, the transition from $\mathbf{S}_{(0)}$ to $\mathbf{S}_{(1)}$, after some symmetric permutation of the leading $J \times J$ submatrix has been applied (see below), is given by

$$\mathbf{S}_{(1)} = \begin{bmatrix} \|\mathbf{x}_1\|^2 & \dots & \hat{\mathbf{x}}_1^T \mathbf{x}_i & \dots & \hat{\mathbf{x}}_1^T \mathbf{y} \\ & \ddots & \vdots & & \vdots \\ & & \underbrace{\|\mathbf{x}_i - (\hat{\mathbf{x}}_1^T \mathbf{x}_i) \hat{\mathbf{x}}_1\|^2}_a & \dots & \frac{\mathbf{x}_i^T \mathbf{y} - (\hat{\mathbf{x}}_1^T \mathbf{x}_i)(\hat{\mathbf{x}}_1^T \mathbf{y})}{\|\mathbf{x}_i - (\hat{\mathbf{x}}_1^T \mathbf{x}_i) \hat{\mathbf{x}}_1\|} \\ & & & \ddots & \vdots \\ & & & & \underbrace{\|\mathbf{y} - (\hat{\mathbf{x}}_1^T \mathbf{y}) \hat{\mathbf{x}}_1\|^2}_b \end{bmatrix}, \quad (19)$$

where $\hat{\mathbf{x}}_1 = \mathbf{x}_1 / \|\mathbf{x}_1\|$. Note that only the upper-triangular part of the symmetric matrix is displayed, in which ultimately the strict upper triangular part contains the elements, and the diagonal contains the square of the elements in the corresponding Cholesky factor. The term denoted by b in Equation (19) is the square of the residual norm of the model response after including the explanatory feature \mathbf{x}_1 , and the row permutation $\Pi_{(1)}$ (and its corresponding column permutation). Therefore the explanatory feature $\hat{\mathbf{x}}_1$ is chosen such that $\hat{\mathbf{x}}_1^T \mathbf{y}$ is maximized.

Proceeding recursively, the application of step i of the Cholesky factorization in Equation (19) replaces the lower $(J+1-i) \times (J+1-i)$ submatrix by the covariance matrix for the remaining $(J-i)$ features and the response, with all features being adjusted for their partial correlations on the first i selected features. If the value of the term b at the i 'th step is denoted $b_{(i)}$, then the model variance in the Gaussian error model in (8) using the first i features is estimated by

$$\sigma_{(i)}^2 = \frac{b_{(i)}}{M - i - 1} \quad 0 \leq i \leq J'. \quad (20)$$

By examining the diagonal terms (denoted by a in Equation (19) above) for small values, it is possible to detect collinear features and apply permutations that will exclude these features from inclusion into the model (even though these features may have a large partial correlation with the adjusted response).

This procedure therefore selects non-collinear features through a stepwise approach where the addition of each new feature leads to a maximum decrease in the residual variance of the response over the training data. The procedure terminates after identifying the set of $J' \leq J$ features with no further non-collinear features.

From this ordered list of J' features, the optimum subset of size $\hat{J} \leq J'$ is the one that has minimum value empirical negative log-likelihood for the corresponding linear regression model on the hold-out data. For example, let $\mathbf{a}_{(i)}$ denote the vector of linear regression coefficients, which has zero entries except for the estimates of the coefficients of the first i features from the stepwise-selection procedure. Then the corresponding residual error $\tilde{\rho}_{(i)}^2$ on the hold-out data is given by

$$\tilde{\rho}_{(i)}^2 = \begin{bmatrix} -\mathbf{a}_{(i)}^T & 1 \end{bmatrix} \left[\tilde{\mathbf{S}} - \tilde{M}(\tilde{\boldsymbol{\mu}} - \boldsymbol{\mu})(\tilde{\boldsymbol{\mu}} - \boldsymbol{\mu})^T \right] \begin{bmatrix} -\mathbf{a}_{(i)} \\ 1 \end{bmatrix}. \quad (21)$$

Note that the hold-out data covariance matrix $\tilde{\mathbf{S}}$ has been re-centered to the training data mean, and \tilde{M} denotes the number of hold-out data records. The optimum feature subset size \hat{J} is then obtained as

$$\hat{J} = \arg \min_i \frac{1}{2} \left[\log 2\pi\sigma_{(i)}^2 + \frac{\tilde{\rho}_{(i)}^2}{\tilde{M}\sigma_{(i)}^2} \right], \quad 0 \leq i \leq J', \quad (22)$$

where $\sigma_{(i)}$ is given by (20). Having determined the optimum \hat{J} set of features for inclusion in the model from (22), the means and covariances of the training and hold-out data are combined as in Section (4.1) and accurate final estimates of the coefficients of the selected features are obtained using Equations (16) and (17) in the usual way.

Finally, if $\hat{J} = 0$ from (22), then the probability model response is just the mean over the relevant training data. This situation may naturally arise from the feature selection heuristic, or may be imposed explicitly by excluding all explanatory features from the specification of the linear regression models. In either case, this results in the special case of the piece-wise constant LRT model, which differ in the implementation details, but yield results similar to earlier programs such as CART [6] and C4.5 [21].

4.3 Comparison with Previous Work

Decision tree algorithms with linear regression node models have been considered before ([17], [11], [20]). The Tree-Structured Regression (TSR) code in [17] does not use feature selection in the segment models, and the computationally-expensive regression algorithms used there are only practical with small memory-resident data sets. The description of the Regression Tree Induction System (RETIS) [11] does not provide enough implementation details to assess its scalability for large training data sets. In [20], a modification to the usual C4.5 methodology [21] is proposed. Here the usual decision tree constructed as described in Section (2) is considered, and for each interior node, a linear regression model is considered using only those features appearing in the splitting conditions of the rooted subtree at that node. If the resulting model is more accurate than using the original rooted subtree, then this internal node is converted into a leaf node with the corresponding linear regression model. This post-processing is effective in reducing the size and improving the interpretability of the usual decision tree model in Section (2) for many examples. However, it cannot be used to reconstruct even simple piecewise-linear regression functions, since the segmentation in the initial decision tree cannot identify the slope discontinuities in the generative regression model.

5 NBT Implementation Details

In contrast to LRT, the implementation of NBT with segment logistic regression models is more problematic, since with the usual logistic fitting approach, several data scans are required to fit even a single model, and there is no efficient way to use hold-out data for feature selection. Furthermore, there is no set of sufficient

statistics that allows two or more individual logistic regression model objects to be combined in the bottom-up merging step in ProbE.

These difficulties are addressed by using the Naive Bayes assumption, which leads to a simplified logistic model in which nonlinear and interaction effects are omitted [2] (see pp. 92-93). The fixed coefficients of the logistic model using this assumption can directly estimated from the data, along with certain heuristics for the feature selection and binary merging step as described below.

5.1 Naive Bayes Model

The response conditional probability in segment t can be expressed using Bayes rule as

$$\theta_k(\mathbf{x}, t) = \frac{P(\mathbf{X} = \mathbf{x}|y = k, t)\pi_k(t)}{\sum_{k'=1}^K P(\mathbf{X} = \mathbf{x}|y = k', t)\pi_{k'}(t)}, \quad (23)$$

where $\pi_k(t)$ is defined in Section (2). If $\theta_k(\mathbf{x}, t)$ is assumed to be independent of the covariates in (23) (i.e., $\theta_k(\mathbf{x}, t) = \pi_k(t)$), we obtain the usual model (see Section (2)) used in most previous decision tree algorithms. An extension of that simple model to include covariate dependencies is provided by the Naive Bayes assumption

$$P(\mathbf{X} = \mathbf{x}|y = k, t) = \prod_{j=1}^J P(X_j = x_j|y = k, t), \quad (24)$$

in which the covariates $\{X_j\}_{j=1}^J$ in \mathbf{X} are assumed conditionally independent given the response y . Using this model, the the relevant conditional probabilities in (23) can be estimated from a set of sufficient statistics obtained from a single training data scan.

Continuous-valued covariates can be used in this model (24) by fitting parametric or non-parametric models to the relevant univariate conditional distributions from the training data [9], or as we have done, by discretizing and binning, to obtain a corresponding derived categorical variable [13]. The simplest uniform discretization can be very effective for Naive Bayes modeling as shown by [8].

Therefore, if the covariate X_j takes on M_j values denoted by $1, 2, \dots, M_j$ respectively, the estimate $P_{jmk}(t)$ for $P(X_j = m|y = k, t)$ in (24) is given by

$$P_{jmk}(t) = \frac{N_{jmk}(t) + \lambda_{jmk}}{N_k(t) + \sum_{m'}^{M_j} \lambda_{jm'k}}, \quad (25)$$

(we note that $\sum_{m=1}^{M_j} N_{jmk}(t) = N_k(t)$), with the empirical frequency counts being smoothed as described earlier in Section (2), and in particular, the parameter values $\lambda_{jmk} = 1$ correspond to Laplace smoothing. The frequency counts in (25) are the sufficient statistics for estimating $P_{jmk}(t)$, and can be accumulated in a single pass over the data.

The empirical negative log-likelihood with this model is given from (23), (24) and (25) as,

$$\mathcal{L}_{TR}(t) = - \underbrace{\sum_{k=1}^K \frac{N_k(t)}{N(t)} \left[\log \pi_k(t) + \sum_{j=1}^J \sum_{m=1}^{M_j} \frac{N_{jmk}(t)}{N_k(t)} \log P_{jmk}(t) \right]}_{\mathcal{A}}$$

$$+ \underbrace{\frac{1}{N(t)} \sum_{i=1}^{N(t)} \log\left(\sum_{k'=1}^K \{\prod_{j=1}^J P_{jx_{j,i}k'}(t)\} \pi_{k'}(t)\right)}_{\mathcal{B}}. \quad (26)$$

where $x_{j,i}$ denotes the value of x_j for the i 'th training data point. The exact evaluation of $\mathcal{L}_{TR}(t)$ requires an additional training data scan, since the sufficient statistics and the estimates of $\pi_k(t)$ and $P_{jmk}(t)$ from Section (2) and (25) allow only the term denoted by \mathcal{A} in (26) to be evaluated exactly. In addition, when merging the sufficient statistics from two or more probability model objects, again only the \mathcal{A} term can be exactly evaluated for the resulting combined model object. In all cases, the evaluation of the term denoted by \mathcal{B} in (26) requires a separate pass over the training data, with the contribution of each data point being evaluated and summed. In [19], we described a Monte Carlo heuristic to approximate the troublesome term \mathcal{B} in (26), which in conjunction with a BIC penalized likelihood approach [22] can be used to obtain the best feature set for a given Naive Bayes model using just two training data scans. Furthermore, along with some further heuristics, the binary merging step in ProbE can also be performed with just three training data scans, as shown below.

5.2 Feature selection and Computational heuristics

The forward-selection algorithm for introducing features in the Naive Bayes model is similar to [16] but with a different feature selection criterion based on the maximum induced decrease in the empirical negative log-likelihood of the training data (this is consistent with the scoring function used in the overall model expansion in ProbE). For each segment Naive Bayes model, feature selection is implemented by performing a first data scan to collect the frequency counts in (25). With these statistics and the Monte Carlo heuristic referred to earlier, estimates for \mathcal{L}_{TR} can be obtained for ordering the covariate features by the forward-selection algorithm. Finally, a second data scan is used to evaluate \mathcal{L}_{TR} exactly for all J model subsets of size $1, 2, \dots, J$, with this ordering of covariate features. The minimum value of the BIC-penalized empirical negative log-likelihood criterion [22] is then used to identify the optimal-sized model from this sequence. As shown in [19], this approach leads to models that are comparable in predictive accuracy to those obtained by more computationally-intensive algorithms, and is quite practical in a ProbE implementation.

In the binary merging algorithm with the Naive Bayes models in ProbE, the Monte Carlo heuristic is used to complete this step with at most three training data scans. In the first phase, which involves the multi-way splits for each dependent variable, the frequency counts are collected from a training data scan. The Monte Carlo estimate for \mathcal{L}_{TR} is then obtained without feature selection. The binary merging steps are carried out by merging the frequency counts from two or more model objects, and then using the Monte Carlo heuristic at each step to estimate the scoring function until the best binary split has been identified for each feature (see Section (3.1)). In the second phase, a training data scan is used to provide an

exact evaluation of the scoring function (26) for all the individual binary splits on each feature. From this exact evaluation, the best binary split over all features can be identified. Finally, in the third phase, this best binary split is considered and the Naive Bayes models in the resulting segments are re-trained using feature selection as described in the previous paragraph.

In contrast to the model expansion step in LRT where only one data scan may suffice, each model expansion step in NBT requires at least three data scans (despite the use of the Monte Carlo heuristic).

5.3 Comparison with Previous Work

In [12], a decision tree approach with Naive Bayes probability models in the segments is described. Here the usual maximum entropy or Gini criterion in CART [6] (see (2)) is used to identify a certain “best” candidate split for each feature (see also Section (2)). From this set of candidate splits, by using Naive Bayes models in the resulting set of child nodes from these candidate splits, and using the scoring function based on the cross-validated misclassification error, the best split over all features is then selected. In comparison, our present approach is more exhaustive in searching for the best possible binary split, since the full Naive Bayes models are used in the child segments throughout the model expansion procedure. Furthermore, our approach uses feature selection in the segment Naive Bayes models, which can be advantageous in data mining domains that invariably contain a large number of conditionally correlated features whose inclusion can degrade the accuracy the resulting Naive Bayes models.

6 Results

The LRT and NBT methodologies have been applied to a number of synthetic and benchmark data sets to validate the correctness of algorithms. The results are of academic interest and omitted for brevity. Since the methodologies are optimized for massive, heterogenous data sets, we are in the process of identifying such data sets for our benchmarking studies. The results reported here used the stopping criterion for ProbE model expansion as the first minimum of the empirical negative log-likelihood on internal holdout data. This stopping criterion has worked well, even though early stopping due to a local minimum is possible with smaller data sets (we are planning to include pruning and other model search methods to overcome this limitation in future work).

6.1 Fingerhut data set

This data set is an example of the motivating application from Fingerhut in Section (1). It contains historic data on the customer response to an annual catalog mailing. The data set had 1380 fields, with 1231 continuous fields, 124 nominal fields and 25 ordinal fields, respectively. The percentage of responders in the entire data set was about 7.5%. There were 7 different potential response features in the data that were possible target variables for predictive modeling. These included for

example, gross sales (dollars), buy/no-buy indicator, credit delinquency indicator, among others. The explanatory features in the data included purchase history details, credit history, and demographic features. The entire data set consisted of about 500,000 records (about 2.5 Gigabytes) and is too large for most conventional statistical modeling packages.

For ProbE modeling, roughly a third of the data was set aside for model evaluation, and the remaining data was used for model training. Three ProbE models were considered, (A) a “Consolidated Payout Model” for predicting the total spending of a customer in dollars, (B) a “Response Model” for predicting the probability of the customer response to a catalog promotion, and (C) a “Conditional Payout Model” for predicting the total spending of a customer in dollars given that the customer responded to the mailing. A comparison of our results with Fingerhut’s proprietary models for the same data is given in ([1]). In summary, the predictive accuracy of our models obtained “out-of-the-box” from the raw data, was comparable or slightly better than the proprietary models. Since Fingerhut has extensively benchmarked their proprietary models favorably against other techniques and vendor modeling tools, we regard this result as an important practical validation of our approach.

We show results of the lift curves for various models on this data set. (the lift curves are generated by ranking the records in the evaluation data set by decreasing predicted response, and plotting the cumulative percentage of records against the cumulative actual response). Figure (1) shows the lift curves from the piecewise-constant and piecewise-linear models using LRT for model (A) for the customer gross sales response. Also shown are the lift curves for the piecewise-constant and piecewise-linear models using NBT for model (B) for predicting the 0/1 buy/no-buy response variable. In both cases, the piecewise-linear case, which incorporates the covariate feature dependencies in each segment response model, yields better lift over the entire range of the response variable, and this is consistent with our belief that the overall predictive accuracy benefits from using more elaborate segment probability models.

6.2 KDD-cup 98 competition data

The KDD-cup 98 competition data is based on a direct mail fund-raising effort, and details are given in www.kdnuggets.com/meetings/kdd98/kdd-cup-98.html. The training data set consists of 95,413 records with 481 fields and an LRT model was constructed for the net revenue field therein. This model is used to find all the respondents with revenue exceeding the mailing and production cost, on a separate validation data set with 96367 records. The actual net profit (i.e., revenue – mailing cost) from this subset of respondents on the validation data was obtained. The winning entry in this competition had a net profit of \$ 14,712. In our case, the LRT model with this data yielded a net profit of \$ 12,745. After adding some derived features and eliminating some existing features (based on the guidelines from the top three winning entries in the competition) the revised LRT model yielded a net profit of \$ 13,778. These results would have been respectively in 8’t and 6’t place in this competition, which is creditable for an “out-of-the-box” experiment.

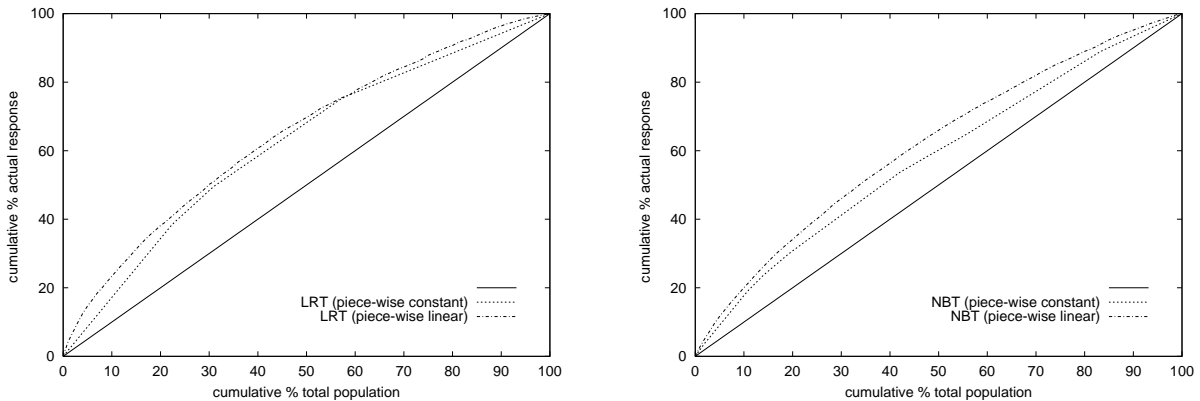


Figure 1. A comparison of the lift on the Fingerhut data for the “Consolidated Payout Model” using the LRT methodology (left), and for the “Response Model” using the NBT methodology (right).

6.3 Adult data set

This is a standard data set from [4] with the 32561 training and 16281 test data and about 7% missing value records in the training data. The data has 6 continuous and 8 nominal features and the response class is a binary nominal feature. The NBT model built from the training data gave an error rate of 0.145 on the test data, while a standalone Naive Bayes within ProbE had a error rate of 0.177. These values reported in the literature are 0.141 (for the NBTree algorithm in [12]), 0.155 (for C4.5) and 0.161 (for Naive Bayes) - these reported results do not make use of the missing data training records.

7 Summary Remarks

We have described two new methods for segmented predictive modeling, viz., Linear Regression Tree (LRT) and Naive Bayes Tree (NBT). Their implementation in the ProbE data mining framework allows these methods to scale to the massive training data sets prevalent in emerging data mining applications. The significant aspects of our proposed methods are the ability to jointly optimize the segmentation with the segment response modeling, and the incorporation of non-trivial segment response models with covariate feature selection.

Acknowledgements

We thank Deb Campbell (Fingerhut), Chid Apte and Fateh Tipu (IBM) for their insights and advice.

Bibliography

- [1] C. Apte, E. Bibelnieks, R. Natarajan, E. Pednault, F. Tipu, D. Campbell and B. Nelson, *Segmentation-Based Modeling For Advanced Targeted Marketing*, Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco CA (2001).
- [2] A. Agresti, *Categorical Data Analysis*, John Wiley, New York (1990).
- [3] A. Björck, *Numerical Methods for Least Squares Problems*, SIAM, Philadelphia (1996).
- [4] C. Blake, E. Keogh and C. Merz, UCI repository of machine learning databases. (<http://www.ics.uci.edu/mllearn>).
- [5] P. Brazdil and J. Gama, Statlog project datasets, <http://www.nccp.up.pt/liacc/ML/statlog>.
- [6] L. Breiman, J. Friedman, R. Olshen and C. Stone, *Classification and Regression Trees*, Wadsworth, Belmont CA (1984).
- [7] G. H. Golub and C. F. Van Loan, *Matrix Computations*, Second Edition, Johns Hopkins University Press, Baltimore (1989).
- [8] C. N. Hsu, J. J. Kuang and T. T. Wong, *Why Discretization Works for Naive Bayesian Classifiers*, Proceedings of the Seventeenth International Conference on Machine Learning, Morgan Kaufman, San Francisco, pp. 399-406 (2000).
- [9] G. John and P. Langley, *Estimating Continuous Distributions in Bayesian Classifiers*, Proc. of the Eleventh Conference on Uncertainty in Artificial Intelligence, pp. 338-345 San Mateo CA (1995).
- [10] G. V. Kass, *An exploratory technique for investigating large amounts of categorical data*, Applied Statistics, Vol. 29, pp. 119-127 (1980).
- [11] A. Karalic, *Linear regression in regression tree leaves*, in Proceedings of ISSEK'92 (International School for Synthesis of Expert Knowledge) Workshop, Bled, Slovenia (1992), (also available from <http://obelix.ijs.si/AramKaralic/retis/>).

- [12] R. Kohavi, *Scaling Up the Accuracy of Naive-Bayes Classifiers: a Decision-Tree Hybrid*, Proc. of the Second International Conference on Knowledge Discovery and Data Mining, (1996).
- [13] R. Kohavi and M. Sahimi, *Error-Based and Entropy-Based Discretization of Continuous Features*, Proc. of the Second International Conference on Knowledge Discovery and Data Mining, pp. 114-119 (1996).
- [14] I. Kononenko, *Semi-Naive Bayesian Classifier*, Proc. of the Sixth European Working Session on Learning, Pittman, Porto, Portugal, (1991).
- [15] P. Kontkanen, P. Myllylämki, T. Silander, Henry Tirri, *BAYDA: Software for Bayesian Classification and Feature Selection*, Proc. of the Fourth International Conference on Knowledge Discovery and Data Mining, (1998).
- [16] P. Langley and S. Sage, *Induction of Selective Bayesian Classifiers*, Proc. of the Tenth Conference on Uncertainty in Artificial Intelligence, Morgan Kaufman, San Francisco (1994).
- [17] W. Y. Loh and C. Yan, *Tree-Structured Regression (TSR) User Guide*, available from <http://www.stat.wisc.edu/p/stat/ftp/pub/loh/treeprogs/tsr/tsrman.zip> (1995).
- [18] D. C. Montgomery and E. A. Peck, *Introduction to Linear Regression Analysis*, Wiley, New York (1982).
- [19] R. Natarajan and E. Pednault, *Using Simulated Pseudo Data To Speed Up Statistical Predictive Modeling*, to appear in Proceedings of the First SIAM International Conference on Data Mining, SIAM Philadelphia (2001). (1995).
- [20] J. Quinlan, *Learning with Continuous Classes*, Proceedings of Fifth Australian Joint Conference on Artificial Intelligence, World Scientific Press, Singapore (1992), pp. 343-348.
- [21] J. Quinlan, *C4.5: Programs for Machine Learning*, Morgan Kaufman, San Mateo (1986).
- [22] G. Schwarz, *Estimating the dimension of a model*, The Annals of Statistics, 6 (1985), pp. 461-464.