Journal of Statistical Software

MMMMMM YYYY, Volume VV, Issue II. *doi: 10.18637/jss.v000.i00*

pboost: A Profile Boosting Framework for Feature Selection in Parametric Models

Zengchao Xu
Shanghai Normal University

Yi Zhang
Shanghai Lixin University of Accounting and Finance

Abstract

We propose a boosting feature selection framework for a broad range of parametric models, including generalized linear models, quantile regression, proportional hazards models, and beta regression models, based on profile loss analysis. The R package **pboost** implements this framework with a unified interface, providing seamless usage for many widely used regression models. The proposed profile boosting method is computationally scalable in ultra-high-dimensional settings while maintaining stable accuracy in model selection. Extensive numerical simulations demonstrate that the profile boosting approach achieves high accuracy in identifying significant features with low false discovery rates, even in scenarios with large p and small n.

Keywords: profile boosting, feature selection, partial profile score, parametric regression.

1. Introduction

In regression models, covariates can be incorporated through their linear, non-linear, or interaction effects with other covariates. In this paper, we refer to all types of effects induced by covariates as features. When the full model that includes all potential features is not optimal or even fails to fit, it becomes essential to identify a reduced model formulated by a suitable subset of candidate features, which is beneficial to eliminate collinearity among features, stabilize coefficient estimation, and improve model prediction accuracy. Therefore, feature selection plays an indispensable role in identifying the most important features from the candidates under consideration.

In the domain of statistical learning, regularization is the most popular technique for feature

selection. Various regularization strategies primarily bifurcate into penalization methods and boosting methods:

(1) Penalization is an explicit regularization scheme that mitigates overfitting through imposing a penalty term on feature effects in the model loss function. The essence of penalization lies in shrinking the penalized feature effects towards zero. As a result, features with exact zero effects are dropped from the model, achieving feature selection. Numerous penalty functions have been proposed to accommodate various scenarios, such as Lasso (Tibshirani 1996), SCAD (Fan and Li 2001), Elastic-Net (Zou and Hastie 2005), and Adaptive Lasso (Zou 2006). Penalization methods generally require solving optimization problems, which can be computationally challenging in high dimensions, and significant efforts have been devoted to algorithmic development and model tuning strategies to handle these issues (Friedman *et al.* 2007, 2010). For an extensive review on penalization methods, the reader is refered to Hastie *et al.* (2015).

(2) Conversely, boosting constitutes an implicit regularization scheme that produces a continuous feature path in its nested fitting procedure, achieving feature selection through early stopping of iterative fitting (Bühlmann and Hothorn 2007). In boosting procedure, it starts with an initial model (often the null model) and progressively adds feature information until the model is sufficiently saturated. Consequently, features included before iteration stops are selected to build the final model. Compared to penalization methods, boosting avoids solving large-scale optimization problems and performs more efficiently and scalably under ultra high-dimensional and complex models (Bühlmann and Yu 2003; Bühlmann 2006; Lutz and Bühlmann 2006), which makes it particularly appealing for feature selection task.

This paper focuses on a specific boosting strategy called "component-wise boosting" to develop a feature selection framework for parametric regression models by exploiting the profile information of the empirical risk function. Formally, let Y be the response variable of interest in regression, and $\boldsymbol{X} \in \mathbb{R}^p$ be the p-dimensional feature vector related to Y. A parametric regression model establishes the relationship between characteristics of Y, such as the conditional expectation $\mu(\boldsymbol{x}) = \mathsf{E}[Y \mid \boldsymbol{X} = \boldsymbol{x}]$ or its conditional quantiles, and the linear predictor $\eta(\boldsymbol{x}) = \boldsymbol{x}^\top \boldsymbol{\beta}$ through a pre-specified link function $\eta = f(\mu)$, where $\boldsymbol{\beta}$ represents the effects of features to be estimated. Denote by $\{(y_i, \boldsymbol{x}_{(i)}), 1 \le i \le n\}$ the independent observations of $(Y, \boldsymbol{X})$, and $\boldsymbol{x}_j = (x_{1j}, x_{2j}, \cdots, x_{nj})^\top$ the j-th feature vector. In the full model, the effect vector $\boldsymbol{\beta}$ is estimated by minimizing the empirical risk function:

$$\min_{\boldsymbol{\beta}} \sum_{i=1}^{n} \ell(y_i, \eta_i), \tag{1}$$

where $\eta_i = \boldsymbol{x}_{(i)}^\top \boldsymbol{\beta}$ and the loss function $\ell(y, \eta)$ is generally assumed to be differentiable with respect to η. However, the full model may be problemistic in many practical scenarios, especially when the number of features p is large. Feature selection aims to identify a reduced model that includes only a suitable subset of features to build the relationship between Y and $\boldsymbol{X}$ while maintaining comparable performance to the full model.

In the classical component-wise boosting procedure (Hothorn and Bühlmann 2006; Hothorn *et al.* 2010), let $\widehat{\boldsymbol{\beta}}^{(t)}$ be the estimate of $\boldsymbol{\beta}$ after the t-th step of boosting, and $\eta_i^{(t)} = \boldsymbol{x}_{(i)}^\top \widehat{\boldsymbol{\beta}}^{(t)}$.

Given the negative gradient vector of the empirical risk function with respect to $\{\eta_i^{(t)}\}$

$$\boldsymbol{g}^{(t)} = -\Big[\frac{\partial \ell(y_1, \eta_1^{(t)})}{\partial \eta}, \ \frac{\partial \ell(y_2, \eta_2^{(t)})}{\partial \eta}, \ \cdots, \ \frac{\partial \ell(y_n, \eta_n^{(t)})}{\partial \eta}\Big]^\top,$$

the principle of plain component-wise boosting in $(t+1)$-th step is described as follows:

(1) **Selecting Rule**: Select the feature that best fits the negative gradient vector $\boldsymbol{g}^{(t)}$ in least squares, i.e.,

$$j^* = \arg\min_{1 \le j \le p} \|\boldsymbol{g}^{(t)} - b_j \boldsymbol{x}_j\|^2,$$

and let $\widehat{b}_{j^*}$ be the estimated coefficient of b_{j^*}.

(2) **Updating Rule**: Update the j^*-th component in $\widehat{\boldsymbol{\beta}}^{(t)}$ with a small portion of $\widehat{b}_{j^*}$ while leaving the others intact, i.e.,

$$\widehat{\beta}_{j^*}^{(t+1)} = \widehat{\beta}_{j^*}^{(t)} + \nu \widehat{b}_{j^*}, \ 0 < \nu < 1,$$
$$\widehat{\beta}_{j}^{(t+1)} = \widehat{\beta}_{j}^{(t)}, \ j \neq j^*.$$

(3) **Stopping Rule**: Iterate the selection and updating steps until reaching the pre-specified maximal number of iterations.

In boosting terminology, the updating rule refers to a base-learning procedure that generates a path towards the minimizer of the empirical risk function. Component-wise boosting restricts the update direction from $\widehat{\boldsymbol{\beta}}^{(t)}$ to $\widehat{\boldsymbol{\beta}}^{(t+1)}$ exactly parallel to the j^*-th axis in $\mathbb{R}^p$ and orthogonal to the others. The factor ν is learning rate, playing a critical role in regularization since a small ν yields a shrinkage estimate when iterative fitting is stopped early enough. Consequently, it achieves feature selection as some features are not yet included in the model when iterations stop (Zhang and Yu 2005; Yao *et al.* 2007).

In R (R Core Team 2025), the **mboost** package (Hothorn and Bühlmann 2006; Hothorn *et al.* 2010; Hofner *et al.* 2014; Hothorn *et al.* 2024) implements component-wise boosting for a wide range of parametric regression models, including generalized linear, additive, and interaction models. Further implementations for additional models are available in the **boost-R** repository (`https://github.com/boost-R`).

The essence of component-wise boosting corresponds to replacing the classical Fisher scoring algorithm for maximum likelihood estimation with a gradient descent algorithm, yielding a reduced model as a byproduct due to the early stopping setup. However, its base-learning strategy has some limitations from the perspective of feature selection:

(1) The selection rule based on partial approximation to the overall gradient is insufficient to evaluate the contribution of features to the model, and a feature may be revisited multiple times during the boosting procedure.

(2) In the updating rule, a small learning rate ν might overshoot the effects of significant features, while a large ν could make the effects of irrelevant features too close to those in the original model.

(3) In the stopping rule, Hofner *et al.* (2014) suggests stopping the learning procedure after a finite number of iterations determined by cross-validation. While this method can help prevent overfitting, it may be too restrictive to achieve a satisfactory reduced model with balanced complexity and saturation.

In this paper, we develop a profile boosting feature selection approach for parametric regression models by redefining the three rules in component-wise boosting:

(1) In the selection step, we propose a novel criterion, partial profile score, to select the next potential feature with the most significant contribution to decreasing the empirical risk. The partial profile score is defined based on the empirical risk function profiled on the selected features, thus providing a more accurate evaluation of the significance of unselected features for the model.
(2) In the updating step, the selected feature is added into the current model, and the effects of all selected features are re-estimated through refitting the augmented model.
(3) In the stopping step, the updated model is assessed by a model selection criterion to evaluate its saturation and complexity. The boosting procedure is stopped when the criterion level of the updated model reaches the optimal level.

The profile boosting feature selection approach highlights in three key aspects: First, it is applicable to a broad class of parametric regression models that have the formulation as in Equation (1), including generalized linear models, quantile regression, proportional hazards models and beta regression models. Second, it possess the selection consistency property, meaning it can correctly identify relevant features with high probability under finite samples, which is desirable for feature selection methods. Third, it is highly scalable in practice due to its extreme computational effeciency, even under ultra high-dimensional and non-linear models.

In Section 2, we present the profile boosting feature selection framework for parametric regression. The implementation of the profile boosting framework in R is introduced in Section 3, including the generic interface and several wrapped models. The accurancy and time cost of profile boosting feature selection are justified through extensive numerical simulations in Section 4. A summary is provided in Section 5. The details of parital profile score are provided in Appendix A.

2. The profile boosting

Let $S = \{1, 2, \cdots, p\}$ be the universal index set of all potential features $\boldsymbol{X} = (X_1, X_2, \cdots, X_p)$. Denote by s the index set of selected features, which is a subset of S, and denote by $\bar{s}$ the unselected features, which is the complementary set of s in S. The notation s also refers to the reduced model with features in s. Furthermore, let $\boldsymbol{\beta}_s$ and $\boldsymbol{\beta}_{\bar{s}}$ be the sub-vectors of $\boldsymbol{\beta}$ with components indexed by s and $\bar{s}$ respectively.

In the profile boosting feature selection framework, we propose a novel criterion, partial profile score (PPS), to evaluate the potential importance of unselected features for improving the model saturability. Specifically, given a selected model s, the PPS of an unselected feature $j \in \bar{s}$ is defined using its gradient of the empirical risk function profiled on the selected model s. Recall that the empirical risk function is given by $L(\boldsymbol{\beta}) = L(\boldsymbol{\beta}_s, \boldsymbol{\beta}_{\bar{s}}) = \sum_{i=1}^{n} \ell(y_i, \eta_i)$, wherein $\eta_i = \boldsymbol{x}_{(i)}^\top \boldsymbol{\beta} = \boldsymbol{x}_{(i)s}^\top \boldsymbol{\beta}_s + \boldsymbol{x}_{(i)\bar{s}}^\top \boldsymbol{\beta}_{\bar{s}}$. The profile risk function of $\boldsymbol{\beta}_{\bar{s}}$ that exploits the effects of features in s is given by

$$L_p(\boldsymbol{\beta}_{\bar{s}}) = \inf_{\boldsymbol{\beta}_s} L(\boldsymbol{\beta}_s, \boldsymbol{\beta}_{\bar{s}}).$$

We define the PPS for the unselected feature $j \in \bar{s}$ as

$$\psi(j \mid s) = \left. \frac{\partial L_p(\boldsymbol{\beta}_{\bar{s}})}{\partial \beta_j} \right|_{\boldsymbol{\beta}_{\bar{s}}=0}.$$

A feature $j \in \bar{s}$ with larger absoluate PPS will decrease the empirical risk more significantly when it is added into the current model s. The definition of PPS makes sense for parametric models in that the loss function $\ell(y, \eta)$ is almost differentiable with respect to η.

As default, we recommend using the extended Bayesian information criteria (EBIC, Chen and Chen (2008, 2012)) to assess the saturability and complexity of an identified model s in the profile boosting procedure. The EBIC for model s is given by

$$\mathrm{EBIC}(s) = 2\widehat{L}(s) + |s| \log n + 2\gamma \log \binom{p}{|s|},$$

where $\widehat{L}(s)$ is the minimal value of the empirical risk function fitted on model s and $\gamma = \max\{1 - \log n/(2 \log p), 0\}$. The EBIC extends the classical BIC by adding an extra penalty term $2\gamma \log \binom{p}{|s|}$ to control the model size, which encourages the model selection consistency property for the EBIC in high-dimensional regimes.

By integrating the PPS for selecting rule and EBIC for stopping rule, the strategy of profile boosting feature selection framework is summarized as follows:

(1) Initialize $s = \emptyset$ (or a set of pre-selected features).

(2) **Selecting Rule**: Identify the feature with the largest absolute PPS among the unselected features, i.e.,

$$j^* = \arg\max_{j \in \bar{s}} |\psi(j \mid s)|.$$

(3) **Boosting Rule**: The current model s is tentatively boosted as $s^+ = s \cup \{j^*\}$, which is fitted by minimizing the restricted empirical risk:

$$\min_{\boldsymbol{\beta}_{s^+}} L(\boldsymbol{\beta}_{s^+}, \mathbf{0}).$$

(4) **Stopping Rule**: If $\mathrm{EBIC}(s^+) < \mathrm{EBIC}(s)$, update $s = s^+$ and go back to the selecting rule; otherwise, stop the profile boosting procedure and output s as the final model.

The PPS plays a core role in profile boosting to identify the candidate feature in each step. In Appendix A, we show that the PPS has the equivalent form:

$$\psi(j \mid s) = \sum_{i=1}^{n} x_{ij} \left. \frac{\partial \ell(y_i, \eta_i)}{\partial \eta} \right|_{\eta_i = \widehat{\eta}_i(s)} = \langle \boldsymbol{x}_j, \widehat{\boldsymbol{g}}(s) \rangle, \tag{2}$$

where $\widehat{\boldsymbol{g}}(s) = \left(\frac{\partial \ell(y_1, \hat{\eta}_1(s))}{\partial \eta}, \frac{\partial \ell(y_2, \hat{\eta}_2(s))}{\partial \eta}, \cdots, \frac{\partial \ell(y_n, \hat{\eta}_n(s))}{\partial \eta} \right)^\top$, $\widehat{\eta}_i(s) = \boldsymbol{x}_{(i)s}^\top \widehat{\boldsymbol{\beta}}_s = \sum_{j \in s} x_{ij} \widehat{\beta}_j$, and $\widehat{\boldsymbol{\beta}}_s$ is the minimizer of $L(\boldsymbol{\beta}_s, \mathbf{0})$. Therefore, we need only fit the simple model s, and the PPS for all unselected features are computed through a vector inner product, which is much promising under high dimensions. Furthermore, this unified inner-product form facilitates us to implement a generic routine for the profile boosting feature selection framework for parametric regression models.

3. Implemention to profile boosting

The R package **pboost** implements the profile boosting feature selection for a broad range of parametric models. It provides a generic interface `pboost()` to general parametric regression models (Section 3.1), and wrapped implementations for several popular models based on `pboost()` with more consistent usage to the original model fitting functions (Section 3.2). In Section 3.3, we provide an illustrative example to show how to tailor the generic function `pboost()` for a specific model.

3.1. The pboost package

In this package, the workhorse function `pboost()` provides the core and unified interface to profile boosting feature selection for generic regression problems defined in Equation (1). The arguments of `pboost()` are

```
pboost(formula, data, fitFun, scoreFun, stopFun, ...,
       keep = NULL, maxK = NULL, verbose = FALSE)
```

where `formula` plus `data` is a fairly standard approach for model and data specification in R. The right-hand side (RHS) in `formula` specifies all of the candidate features for linear predictor $\eta = \sum_j \beta_j X_j$. For computational efficiency in parsing a high-dimensional RHS, there are some restrictions and recommendations applied to `formula`:

(1) All variables appearing on the RHS must be numeric in the supplied `data`.
(2) Each term on the RHS must correspond to a single column appeared in the model matrix. Supported expressions include main effects (`X1`), interactions (`X1:X2`), and simple transformations (`log(X1)`, `I(X1^2)`, etc.). Complex terms that expand into multiple columns – such as `poly(X1, degree=2)`, `bs(X1)` or `ns(X1)` – are not supported.
(3) Offset terms should not be included in the `formula`. Instead, it should be passed via the dedicated `offset` argument of `fitFun()`.

The three user-specified functions, `fitFun()`, `scoreFun()` and `stopFun()`, are key components to leverage the `pboost()` for a specific regression model, conducting model fitting, gradient computation, and model evaluation respectively during the boosting procedure. The argument `fitFun` is a function to fit the model with given features, which takes arguments `formula`, `data` and other model-specific arguments passed via `...`, and returns an object that contains at least the estimated coefficients. The argument `scoreFun` is a function to compute the gradient vector $\widehat{\boldsymbol{g}}$ in Equation (2) by accepting the model object fitted by `fitFun()`. The argument `stopFun` is a function to evaluate the performance of a fitted model, which takes a model object returned by `fitFun()` as an argument. An illustrative example of implementing these three functions for a logistic model is provided in Section 3.3.

The argument `keep` refers to a pre-specified set of features to be kept in the model during profile boosting, and `maxK` limits the maximal number of features to be selected. If `maxK` is specified, it will supress the stopping rule evaluated by `stopFun()`, meaning that the profile boosting procedure continues until the procudure identifies `maxK` features, producing a nested feature path of a given length.

The **pboost** package provides an `S3` interface `EBIC(object, p, p.keep, ...)` as the stopping function, and its methods for the models listed in Table 1 are also available. Theoretically, the EBIC identifies a much parsimonious model that is consistent to the underlying

true model with high probability for many high-dimensional regression models (Luo and Chen 2014; Xu *et al.* 2022; Yu and Luo 2022).

3.2. The wrapped models

The **pboost** package provides wrapped implementations based on `pboost()` for some popular models that have consistent usage to their plain model fitting functions. The supported models are summarized in Table 1.

Table 1: Supported models in **pboost** package.

Model	Depends **package**`::function()`	Profile boosting function
Linear model	**stats**`::lm()`	`plm()`
Generalized linear models	**stats**`::glm()`	`pglm()`
Proportional hazards models	**survival**`::coxph()`	`pcoxph()`
Regression quantile models	**quantreg**`::rq()`	`prq()`
Beta regression models	**betareg**`::betareg()`	`pbetareg()`

Linear and generalized linear models

The linear model is the most basic model for continuous response, and generalized linear models (GLMs) cover many popular mean regression models for diverse types of response, including continuous data (Gaussian), finite discrete data (Binomial), count data (Poisson), and nonnegative continuous data (Gamma, Log-normal, and Inverse Gaussian).

In the **pboost** package, the following two functions that wrap the `pboost()` function implement the profile boosting feature selection for linear models and GLMs:

```
plm(formula, data, subset, weights, na.action, method = "qr",
    model = TRUE, x = FALSE, y = FALSE, qr = TRUE, singular.ok = TRUE,
    contrasts = NULL, offset, ...,
    stopFun = EBIC, keep = NULL, maxK = NULL, verbose = FALSE)

pglm(formula, family = gaussian, data, weights, subset, na.action,
    start = NULL, etastart, mustart, offset, control = list(...),
    model = TRUE, method = "glm.fit", x = FALSE, y = TRUE, singular.ok = TRUE,
    contrasts = NULL, ...,
    stopFun = EBIC, keep = NULL, maxK = NULL, verbose = FALSE)
```

Both `plm()` and `pglm()` have identical arguments to the model fitting functions `lm()` and `glm()` built-in R package **stats**, respectively, except for the last line specific to profile boosting. Therefore, it is seamless to switch from plain model fitting to profile boosting feature selection by replacing `lm()` or `glm()` with `plm()` or `pglm()`. For example, one can specify the fitting method in `pglm()` via the argument `method = "glm.fit2"` from the **glm2** package for stable fitting, or use `method = "brglmFit"` from the **brglm2** package to reduce the bias in model fitting.

For illustration, we apply the profile boosting feature selection to the prostate tumor gene expression data in package **doBy** using logistic regression model. In the prostate data, it contains $n = 102$ subjects of male (52 patients and 50 healthy controls) with $p = 6,033$ effect sizes of genes. To improve the estimation accuracy, we employ the bias reduction method provided in R package **brglm2** to fit the logistic model during profile boosting.

```
> library(glm2)
> library(pboost)
>
> data("prostate", package = "doBy")
> prostate <- data.frame(y = prostate[["y"]], prostate[["x"]])
> dim(prostate)

[1]  102 6034

> pglm(y ~ ., "binomial", prostate, method = "glm.fit2", verbose = TRUE)

Initial model with level=158.773

Adding X1839: stopping level=114.908

Adding X4336: stopping level=107.381

Adding X5621: stopping level=100.536

Adding X2388: stopping level=103.474


Call:  glm(formula = y ~ X1839 + X4336 + X5621, family = "binomial",
    data = prostate, method = "glm.fit2")

Coefficients:
(Intercept)        X1839        X4336        X5621
     0.5083       3.8692      -1.5279      -3.5748

Degrees of Freedom: 101 Total (i.e. Null);  98 Residual
Null Deviance:            141.4
Residual Deviance: 35.57         AIC: 43.57
```

Regression quantile models

Quantile regression models are distribution-free and establish relationships between the quantiles of the response variable and the features. This approach relaxes most of assumptions of traditional mean regression, making it particularly flexible for handling heteroscedasticity and skewness in the response.

To formulate the τ-quantile of the response using $\eta = \boldsymbol{x}^\top \boldsymbol{\beta}$, the quantile regression model specifies the loss function as an asymmetrically weighted absolute error:

$$\ell(y, \eta; \tau) = w_\tau(y, \eta)|y - \eta|,$$

where $w_\tau(y, \eta) = \begin{cases} 1-\tau, & y < \eta \\ 0, & y = \eta \\ \tau, & y > \eta \end{cases}$ is the influence function. Since the case $y = \eta$ occurs with probability zero, the non-differentiability of $\ell(y, \eta; \tau)$ does not cause practical issues, allowing profile boosting to be effectively applied to quantile regression problems.

In the **pboost** package, the function `prq()` wraps `pboost()` for quantile regression models, with usage:

```
prq(formula, tau = 0.5, data, subset, weights, na.action,
    method = "br", model = TRUE, contrasts = NULL, ...,
    stopFun = EBIC, keep = NULL, maxK = NULL, verbose = FALSE)
```

This aligns with the usage of the `rq()` function in the **quantreg** package for fitting a quantile regression model.

Numerical simulations in Section 4 demonstrate the accuracy and computational efficiency of profile boosting for quantile regression.

Beta regression models

The beta regression model is commonly used to analyze response variables that take values within the standard unit interval $(0, 1)$, such as rates and proportions. The R package **betareg** provides comprehensive tools for mean regression of beta-distributed responses (Cribari-Neto and Zeileis 2010). By leveraging the `betareg()` function from this package, **pboost** offers `pbetareg()` for implementing profile boosting for the mean part in beta regression:

```
pbetareg(formula, data, subset, na.action, weights, offset,
    link = c("logit", "probit", "cloglog", "cauchit", "log", "loglog"),
    link.phi = NULL, type = c("ML", "BC", "BR"), dist = NULL, nu = NULL,
    control = betareg.control(...), model = TRUE, y = TRUE, x = FALSE, ...,
    stopFun = EBIC, keep = NULL, maxK = NULL, verbose = FALSE)
```

This function accepts all arguments compatible with the standard `betareg()` function in the **betareg** package.

Proportional hazards models

The proportional hazards model is widely used for incorporating feature information into the regression analysis of censored data. In the **pboost** package, the function `pcoxph()` implements profile boosting feature selection for the proportional hazards model by calling the standard `coxph()` function from the **survival** package:

```
pcoxph(formula, data, weights, subset, na.action, init, control,
    ties = c("efron", "breslow", "exact"), singular.ok = TRUE,
```

```
robust, model = FALSE, x = FALSE, y = TRUE, tt, method = ties,
id, cluster, istate, statedata, nocenter = c(-1, 0, 1), ...,
stopFun = EBIC, keep = NULL, maxK = NULL, verbose = FALSE)
```

This function accepts all arguments consistent with the standard `coxph()` function.

3.3. Illustration: profile boosting with logistic model using pboost()

In this section, we demonstrate the implementation of profile boosting feature selection using the basic `pboost()` interface to build a parsimonious logistic model for the aforementioned prostate data.

For the logistic model with a logit link function $\log \frac{\mathrm{P}(Y=1)}{\mathrm{P}(Y=0)} = \eta$, the empirical risk function is given by the negative log-likelihood:

$$\ell(y, \eta) = y\eta - \log(1 + \mathrm{e}^{\eta}),$$

where $y \in \{0, 1\}$ represents the binary response variable and $\eta = \boldsymbol{x}^\top \boldsymbol{\beta}$ denotes the linear predictor. To use the `pboost()` function for profile boosting in logistic regression, it suffices to specify the following three functions:

(1) The model-fitting function has the form `fitFun(formula, data, ...)`, where `formula`, `data`, and ellipsis `...` are passed by the arguments in `pboost()`. In R, the standard logistic model is fitted using the built-in function `glm(formula, family, data)` from the **stats** package by specifying `family = "binomial"`.

(2) The score function has the form `scoreFun(object)` to obtain the gradient vector $\widehat{\boldsymbol{g}}(s)$ in Equation (2), profiled on the selected features s. Here, `object` is the fitted model returned by `fitFun()`. For the logistic model, the partial gradient of $\ell(y, \eta)$ at $\eta = \widehat{\eta}(s)$ is given by

$$\left.\frac{\mathrm{d}\ell(y,\eta)}{\mathrm{d}\eta}\right|_{\eta=\hat{\eta}(s)} = y - \frac{\mathrm{e}^{\hat{\eta}(s)}}{1 + \mathrm{e}^{\hat{\eta}(s)}}.$$

Accordingly, the `scoreFun()` for the logistic model is defined as follows:

```
scoreLogis <- function(object) {
    eta.hat <- object[["linear.predictors"]]
    score <- object[["y"]] - 1.0 / (1.0 + exp(-eta.hat))
    return(score)
}
```

(3) The `stopFun(object)` extracts the model performance of a fitted model `object` to determine whether to continue or stop the profile boosting process, as described in Section 2. For example, one can assess the model performance using the Bayesian Information Criterion (BIC) by specifying `stopFun = BIC`. In high-dimensional problems, the EBIC is more preferred, which can be implemented as follows:

```
ebicLogis <- function(object, p = NCOL(data) - 1) {
    n <- nobs(object)
    dof <- attr(logLik(object), "df")
    ebic.factor <- max(0.0, 1.0 - log(n) / (2.0 * log(p)))
    ebic.penalty <- 2.0 * ebic.factor * lchoose(p, dof)
```

```
    return(BIC(object) + ebic.penalty)
}
```

In summary, the following code chunk demonstrates how to implement profile boosting feature selection for the logistic model using the basic interface `pboost()`, yielding results identical to those above obtained with `pglm()`:

```
> pboost(y ~ ., data=prostate, fitFun=glm, scoreFun=scoreLogis, stopFun = ebicLogis,
+        family = "binomial", method = "glm.fit2", verbose = TRUE)

Initial model with level=158.773

Adding X1839: stopping level=114.908

Adding X4336: stopping level=107.381

Adding X5621: stopping level=100.536

Adding X2388: stopping level=103.474


Call:  fitFun(formula = fml, family = "binomial", data = data, method = "glm.fit2")

Coefficients:
(Intercept)        X1839        X4336        X5621
     0.5083       3.8692      -1.5279      -3.5748

Degrees of Freedom: 101 Total (i.e. Null);  98 Residual
Null Deviance:            141.4
Residual Deviance: 35.57         AIC: 43.57
```

For non-standard models, one can define the fitting function `fitFun()` that minimizes an empirical risk function using optimization packages in R, such as `mle()` from the **stats4** package or `mle2()` from the **bbmle** package. The `scoreFun()` function can also be defined using numerical derivative tools, such as `numericDeriv()` from the **stats** package or `grad()` from the **numDeriv** package.

4. Accuracy and timings

We provide systematic numerical simulations to demonstrate the performance of profile boosting feature selection for the models introduced in Section 3.2. The standard models considered in the simulations include linear model, logistic model, quantile regression with a quantile level $\tau = 0.5$, and beta regression model with logit link.

Throughout the simulations, the data generation under different models shares the following settings:

(1) The dimension of potential features is taken as $p = 200, 400, 600, 800$, or 1000. For linear and quantile regression models, the sample sizes are fixed at $n = 200$. Since the feature selection for logistic and beta regression models are more challenging tasks, the sample sizes are increased to $n = 300$.
(2) The feature data are simulated as $x_{ij} \sim N(0, 1)$ for $1 \le i \le n$ and $1 \le j \le p$.
(3) The linear predictor is set as $\eta_i = \sum_{j=1}^{10} \beta_j x_{ij}$ $(1 \le i \le n)$. The true effects are generated as $\beta_j \sim \text{Unif}(1.5, 2.0)$ for $j = 1, 2, \cdots, 10$, which represents relatively weak signal levels. For example, under such true effects, the signal-to-noise ratio (SNR) is approximately $\frac{1}{5}$ in the linear models. Therefore, only the first ten features are significant in the true model, meaning that the index set of true features is:

$$s_0 = \{1, 2, \cdots, 10\}.$$

For each simulated observation, y_i is independently sampled from the corresponding model using the linear predictor η_i: (1) In the linear models, $y_i \sim N(\eta_i, 1)$. (2) In the logistic models, $y_i \sim \text{Bernoulli}(1/(1 + \exp(-\eta_i)))$. (3) In the quantile regression model, y_i is sampled from a non-central $t(2)$ distribution with parameter η_i. (4) In the beta regression model, y_i is sampled from a beta distribution with mean $\mu = 1/(1 + \exp(-\eta_i))$ and dispersion $\phi = 1$.

Let $\hat{s}$ be the index set of features identified by the profile boosting. Under large-p-small-n scenarios, we use the positive discovery rate (PDR) and false discovery rate (FDR) to assess the accuracy of feature selection, which are given by

$$\text{PDR} = \frac{|\hat{s} \cap s_0|}{|s_0|}, \ \text{FDR} = \frac{|\hat{s} \setminus s_0|}{|\hat{s}|}.$$

The simulation results are summarized in Tables 2 and 3, where the PDR, FDR, and computational time (in seconds) are averaged over 100 replications for each setting.

The numerical results demonstrate that the profile boosting feature selection achieves high accuracy in identifying true features with low false discovery rates under large-p-small-n problems. Moreover, the computational time grows moderately as the dimension of features increases, verifying the computational efficiency of the profile boosting framework. Overall, profile boosting provides a promising and efficient tool for feature selection in high-dimensional parametric regression models.

5. Summary

This paper introduces a profile boosting framework for feature selection in parametric regression models. We present the R package **pboost**, which provides a unified and flexible interface to implement profile boosting. By wrapping standard model-fitting functions available in R, the **pboost** package offers ready-to-use functions for several popular models. Extensive numerical simulations demonstrate that profile boosting achieves high accuracy under finite sample sizes and remains highly scalable to high-dimensional problems. Future research and implementation could extend beyond the single structure discussed in this paper. Potential areas for further exploration include mixed effects models, generalized additive models, and generic parametric models, which can accommodate complex dependcies and dynamic structures.

Table 2: ($n = 200$) PDR, FDR and time cost under linear and quantile models.

Model	p	PDR	FDR	Time
Linear	400	1.000(0.000)	0.127(0.098)	0.110(0.011)
	600	1.000(0.000)	0.116(0.100)	0.212(0.024)
	800	1.000(0.000)	0.118(0.102)	0.358(0.038)
	1000	1.000(0.000)	0.142(0.109)	0.560(0.070)
	2000	1.000(0.000)	0.141(0.118)	2.240(0.300)
	3000	1.000(0.000)	0.131(0.118)	5.048(0.543)
	4000	1.000(0.000)	0.152(0.131)	9.411(1.196)
	5000	1.000(0.000)	0.146(0.127)	15.149(1.938)
Quantile	400	1.000(0.000)	0.039(0.060)	0.120(0.013)
	600	1.000(0.000)	0.041(0.062)	0.234(0.019)
	800	1.000(0.000)	0.036(0.053)	0.375(0.023)
	1000	0.990(0.084)	0.035(0.060)	0.549(0.053)
	2000	0.970(0.162)	0.037(0.061)	2.040(0.314)
	3000	0.946(0.211)	0.045(0.087)	4.500(0.871)
	4000	0.941(0.219)	0.044(0.116)	8.051(1.577)
	5000	0.931(0.226)	0.054(0.107)	12.831(2.526)

References

Bühlmann P (2006). "Boosting for High-Dimensional Linear Models." *The Annals of Statistics*, **34**(2), 559–583. ISSN 0090-5364. doi:10.1214/009053606000000092.

Bühlmann P, Hothorn T (2007). "Boosting Algorithms: Regularization, Prediction and Model Fitting." *Statistical Science*, **22**(4). ISSN 0883-4237. doi:10.1214/07-STS242.

Bühlmann P, Yu B (2003). "Boosting With the L_2 Loss: Regression and Classification." *Journal of the American Statistical Association*, **98**(462), 324–339. ISSN 0162-1459, 1537-274X. doi:10.1198/016214503000125.

Chen J, Chen Z (2008). "Extended Bayesian Information Criteria for Model Selection with Large Model Spaces." *Biometrika*, **95**(3), 759–771. ISSN 0006-3444, 1464-3510. doi:10.1093/biomet/asn034.

Chen J, Chen Z (2012). "Extended BIC for Small-n-Large-P Sparse GLM." *Statistica Sinica*, **22**(2). ISSN 10170405. doi:10.5705/ss.2010.216.

Cribari-Neto F, Zeileis A (2010). "Beta Regression in *R*." *Journal of Statistical Software*, **34**(2). ISSN 1548-7660. doi:10.18637/jss.v034.i02.

Table 3: ($n = 300$) PDR, FDR and time cost under logistic and beta models.

Model	p	PDR	FDR	Time
Logistic	400	1.000(0.007)	0.132(0.125)	0.148(0.025)
	600	0.998(0.014)	0.154(0.146)	0.272(0.048)
	800	1.000(0.000)	0.138(0.133)	0.416(0.065)
	1000	0.997(0.020)	0.155(0.131)	0.597(0.089)
	2000	0.999(0.012)	0.169(0.141)	2.352(0.348)
	3000	0.993(0.030)	0.179(0.148)	5.405(0.772)
	4000	0.996(0.023)	0.172(0.135)	9.716(1.219)
	5000	0.990(0.048)	0.161(0.136)	15.600(1.948)
Beta	400	1.000(0.000)	0.000(0.000)	0.296(0.024)
	600	1.000(0.000)	0.000(0.000)	0.399(0.024)
	800	0.991(0.060)	0.002(0.014)	0.548(0.040)
	1000	0.991(0.060)	0.002(0.014)	0.710(0.053)
	2000	0.952(0.153)	0.000(0.000)	2.012(0.250)
	3000	0.898(0.251)	0.000(0.000)	4.029(0.890)
	4000	0.892(0.251)	0.005(0.021)	7.199(1.620)
	5000	0.768(0.336)	0.000(0.000)	10.046(3.247)

Fan J, Li R (2001). "Variable Selection via Nonconcave Penalized Likelihood and Its Oracle Properties." *Journal of the American Statistical Association*, **96**(456), 1348–1360. ISSN 0162-1459, 1537-274X. doi:10.1198/016214501753382273.

Friedman J, Hastie T, Höfling H, Tibshirani R (2007). "Pathwise Coordinate Optimization." *The Annals of Applied Statistics*, **1**(2). ISSN 1932-6157. doi:10.1214/07-AOAS131.

Friedman J, Hastie T, Tibshirani R (2010). "Regularization Paths for Generalized Linear Models via Coordinate Descent." *Journal of Statistical Software*, **33**(1), 1–22. ISSN 1548-7660. doi:10/bb3d.

Hastie T, Tibshirani R, Wainwright M (2015). *Statistical Learning with Sparsity: The Lasso and Generalizations*. Chapman and Hall/CRC. ISBN 978-0-429-17158-1. doi:10.1201/b18401.

Hofner B, Mayr A, Robinzonov N, Schmid M (2014). "Model-Based Boosting in R: A Hands-on Tutorial Using the R Package Mboost." *Computational Statistics*, **29**, 3–35. ISSN 0943-4062, 1613-9658. doi:10.1007/s00180-012-0382-5.

Hothorn T, Buehlmann P, Kneib T, Schmid M, Hofner B, Otto-Sobotka F, Scheipl F, Andreas M (2024). ***mboost****: Model-Based Boosting*. R package version 2.9-11, URL https://CRAN.R-project.org/package=mboost.

Hothorn T, Bühlmann P (2006). "Model-Based Boosting in High Dimensions." *Bioinformatics*, **22**(22), 2828–2829. ISSN 1367-4811, 1367-4803. doi:10.1093/bioinformatics/btl462.

Hothorn T, Bühlmann P, Kneib T, Schmid M, Hofner B (2010). "Model-Based Boosting 2.0." *Journal of Machine Learning Research*, **11**(71), 2109–2113.

Luo S, Chen Z (2014). "Sequential Lasso Cum EBIC for Feature Selection With Ultra-High Dimensional Feature Space." *Journal of the American Statistical Association*, **109**(507), 1229–1240. ISSN 0162-1459, 1537-274X. doi:10.1080/01621459.2013.877275.

Lutz RW, Bühlmann P (2006). "Boosting for High-Multivariate Responses in High-Dimensional Linear Regression." *Statistica Sinica*, **16**(2), 471–494. ISSN 1017-0405. 24307555.

R Core Team (2025). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Tibshirani R (1996). "Regression Shrinkage and Selection Via the Lasso." *Journal of the Royal Statistical Society: Series B (Methodological)*, **58**(1), 267–288. ISSN 0035-9246, 2517-6161. doi:10.1111/j.2517-6161.1996.tb02080.x.

Xu Z, Luo S, Chen Z (2022). "Partial Profile Score Feature Selection in High-Dimensional Generalized Linear Interaction Models." *Statistics and Its Interface*, **15**(4), 433–447. ISSN 19387989, 19387997. doi:10.4310/21-SII706.

Yao Y, Rosasco L, Caponnetto A (2007). "On Early Stopping in Gradient Descent Learning." *Constructive Approximation*, **26**(2), 289–315. ISSN 1432-0940. doi:10.1007/s00365-006-0663-2.

Yu K, Luo S (2022). "A Sequential Feature Selection Procedure for High-Dimensional Cox Proportional Hazards Model." *Annals of the Institute of Statistical Mathematics*, **74**(6), 1109–1142. ISSN 0020-3157, 1572-9052. doi:10.1007/s10463-022-00824-8.

Zhang T, Yu B (2005). "Boosting with Early Stopping: Convergence and Consistency." *The Annals of Statistics*, **33**(4). ISSN 0090-5364. doi:10.1214/009053605000000255.

Zou H (2006). "The Adaptive Lasso and Its Oracle Properties." *Journal of the American Statistical Association*, **101**(476), 1418–1429. ISSN 0162-1459, 1537-274X. doi:10.1198/016214506000000735.

Zou H, Hastie T (2005). "Regularization and Variable Selection Via the Elastic Net." *Journal of the Royal Statistical Society Series B: Statistical Methodology*, **67**(2), 301–320. ISSN 1369-7412, 1467-9868. doi:10.1111/j.1467-9868.2005.00503.x.

A. Computational details on partial profile score

Let $\widetilde{\boldsymbol{\beta}}_s(\boldsymbol{\beta}_{\bar{s}})$ be the maximizer of $\boldsymbol{\beta}_s$ given $\boldsymbol{\beta}_{\bar{s}}$ in the profile risk function, i.e.,

$$L_p(\boldsymbol{\beta}_{\bar{s}}) = \sup_{\boldsymbol{\beta}_s} L(\boldsymbol{\beta}_s, \boldsymbol{\beta}_{\bar{s}}) = L(\widetilde{\boldsymbol{\beta}}_s(\boldsymbol{\beta}_{\bar{s}}), \boldsymbol{\beta}_{\bar{s}}).$$

It follows that

$$\frac{\partial L(\boldsymbol{\beta}_s, \boldsymbol{\beta}_{\bar{s}})}{\partial \boldsymbol{\beta}_s}\Big|_{\boldsymbol{\beta}_s = \tilde{\boldsymbol{\beta}}_s(\boldsymbol{\beta}_{\bar{s}})} = \mathbf{0}.$$

For $j \in \bar{s}$, we have

$$\frac{\partial L_p(\boldsymbol{\beta}_{\bar{s}})}{\partial \beta_j} = \Big(\frac{\partial \widetilde{\boldsymbol{\beta}}_s^\top}{\partial \beta_j}\frac{\partial L}{\partial \boldsymbol{\beta}_s} + \frac{\partial L}{\partial \beta_j}\Big)\Big|_{\beta_s = \tilde{\beta}_s(\beta_{\bar{s}})} = \frac{\partial L}{\partial \beta_j}\Big|_{\beta_s = \tilde{\beta}_s(\beta_{\bar{s}})}.$$

Given $\boldsymbol{\beta}_{\bar{s}} = \mathbf{0}$, $\widetilde{\boldsymbol{\beta}}_s(\mathbf{0}) = \widehat{\boldsymbol{\beta}}_s$ is exactly the maximum likelihood estimator in the reduced model with features limited to s. Consequently, we have

$$\psi(j \mid s) = \frac{\partial L(\boldsymbol{\beta}_s, \boldsymbol{\beta}_{\bar{s}})}{\partial \beta_j}\Big|_{(\boldsymbol{\beta}_s, \boldsymbol{\beta}_{\bar{s}}) = (\hat{\boldsymbol{\beta}}_s, \mathbf{0})} = \sum_{i=1}^n \frac{\partial \ell(y_i, \eta_i)}{\partial \eta}\Big|_{\eta_i = \widehat{\eta}_i(s)} = \langle \boldsymbol{x}_j,\ \widehat{\boldsymbol{g}}(s)\rangle,$$

where $\boldsymbol{x}_j = (x_{1j}, x_{2j}, \cdots, x_{nj})^\top$ and $\widehat{\boldsymbol{g}}(s) = \Big(\frac{\partial \ell(y_1, \hat{\eta}_1(s))}{\partial \eta}, \frac{\partial \ell(y_2, \hat{\eta}_2(s))}{\partial \eta}, \cdots, \frac{\partial \ell(y_n, \hat{\eta}_n(s))}{\partial \eta}\Big)^\top$.

Affiliation:

Zengchao Xu
Department of Mathematics
Shanghai Normal University
E-mail: xuzengchao@shnu.edu.cn
Yi Zhang
School of Insurance
Shanghai Lixin University of Accounting and Finance
995 Shangchuan Road, Pudong New Area, Shanghai
E-mail: zhangyi@lixin.edu.cn

Journal of Statistical Software http://www.jstatsoft.org/
published by the Foundation for Open Access Statistics http://www.foastat.org/
MMMMMM YYYY, Volume VV, Issue II *Submitted:* yyyy-mm-dd
doi:10.18637/jss.v000.i00 *Accepted:* yyyy-mm-dd