Quantitative Finance

ISSN: 1469-7688 (Print) 1469-7696 (Online) Journal homepage: http://www.tandfonline.com/loi/rquf20

Evolution of high-frequency systematic trading: a performance-driven gradient boosting model

Nan Zhou, Wen Cheng, Yichen Qin & Zongcheng Yin

To cite this article: Nan Zhou, Wen Cheng, Yichen Qin & Zongcheng Yin (2015) Evolution of high-frequency systematic trading: a performance-driven gradient boosting model, Quantitative Finance, 15:8, 1387-1403, DOI: 10.1080/14697688.2015.1032541

To link to this article: http://dx.doi.org/10.1080/14697688.2015.1032541

Quantitative Finance, 2015
Vol. 15, No. 8, 1387–1403, http://dx.doi.org/10.1080/14697688.2015.1032541

Routledge
Taylor & Francis Group

Evolution of high-frequency systematic trading: a performance-driven gradient boosting model

NAN ZHOU†, WEN CHENG‡, YICHEN QIN§ and ZONGCHENG YIN*¶

†Quest Partners LLC, New York, NY, USA
‡J.P. Morgan & Co., New York, NY, USA
§Lindner College of Business, University of Cincinnati, Cincinnati, OH, USA
¶College of Economics & Management, Anhui Agricultural University, Hefei Anhui, China

(*Received 10 September 2013; accepted 19 February 2015*)

This paper proposes a performance-driven gradient boosting model (pdGBM) which predicts short-horizon price movements by combining nonlinear response functions of selected predictors. This model performs gradient descent in a constrained functional space by directly minimizing loss functions customized with different trading performance measurements. To demonstrate its practical applications, a simple trading system was designed with trading signals constructed from pdGBM predictions and fixed holding period in each trade. We tested this trading system on the high-frequency data of SPDR S&P 500 index ETF (SPY). In the out-of-sample period, it generated an average of 0.045% return per trade and an annualized Sharpe ratio close to 20 after transaction costs. Various empirical results also showed the model robustness to different parameters. These superior performances confirm the predictability of short-horizon price movements in the US equity market. We also compared the performance of this trading system with similar trading systems based on other predictive models like the gradient boosting model with L2 loss function and the penalized linear model. Results showed that pdGBM substantially outperformed all other models by higher returns in each month of the testing period. Additionally, pdGBM has many advantages including its capability of automatic predictor selection and nonlinear pattern recognition, as well as its simply structured and interpretable output function.

Keywords: Gradient boosting; High-frequency; TAQ; Systematic trading; Trading performance

JEL Classification: C4, C8, G1

1. Introduction

1.1. Background

Over the last few years, more and more stocks, bonds and other financial products have been traded electronically. Electronic trading transforms the market by improving trading transparency and providing opportunities to greatly reduce trading costs. Automated trading systems have been built by brokers and asset managers to execute trading orders by consistently following systematic rules. These systems even gradually replace human traders.‖

Meanwhile, technology advances in this field have led to wide availability of reliable high-frequency financial data. These huge amounts of trades and quotes information from exchanges and markets have attracted significant attentions from both academia and industry. Many interesting research topics have been studied, such as market microstructure noise (O'Hara 1995, Harris 2002, Aït-Sahalia and Yu 2009, Nehren *et al*. 2012), dynamics of limit order book (Bouchaud *et al*. 2002, Avellaneda and Stoikov 2008) and intra-day volatility estimations (Andersen *et al*. 2003, Zhang *et al*. 2005, Todorov 2009, Zhou 2011).

Given the existence of various off-the-shelf predictive models, researchers from the fields of machine learning and statistics also have tried different methods for predicting price movements of financial products. To name a few, neural networks were applied to predict stock prices as early as in White (1989), and they were also used to predict price movements of stock indices (Kulkarni 1996), stock index futures (In and Kim 2006), currencies (LeBaron 1998, Dempster and Leemans 2006) and index options (Marzi and Turnbull 2007); genetic algorithms (Holland 1975, Allen and Karjalainen 1999) were also popularly utilized with their advantage of interpretability as a combination of trading rules. There have also been some

*Corresponding author. Email: zongcheng.yin@gmail.com
‖http://www.bloomberg.com/news/2012-11-06/million-dollar-traders-replaced-with-machines-credit-markets.html

applications of support vector machine (Tay and Cao 2001) and reinforcement learning (Moody and Saffell 2001, Ganchev *et al*. 2010) among many others. An overview of these applications can be found in Zhang and Zhou (2004) and Doumpos *et al*. (2012).

Most of the aforementioned studies were based on low-frequency daily information from the markets. Because of the extreme high volume and irregularity of high-frequency data, building a system based on these data to predict short-horizon price movements is much more challenging. However, this is practically meaningful in different aspects. For example, in the business of algorithmic trading as client solutions in many international banks, short-horizon price predictions have been used to optimize trading executions and help clients to save transaction costs. Combined with limit order placement models (Lo, MacKinlay *et al*. 2002) and market impact models (Kissell and Glantz 2003), predictions could help trading algorithms tactically deciding the right level of aggressiveness for order placements. If the current price trend is expected to stay favourable, the algorithms may adopt more aggressive trades (crossing and paying the bid-ask spread) to improve the averaged execution costs. On the other side, the algorithms may use passive limit orders instead. Another example is, more directly, as in these proprietary trading firms, such as Jump Trading, Tower Research Capital, Virtu Financial and many others, accurate short-horizon price predictions are arguably the most important work to pursue abnormal investment returns, though whether high-frequency traders rig the stock markets have been recently debated (Lewis 2014, Kovac 2014).

1.2. Contributions

Our research has been partially motivated by Creamer (2012), who implemented a Logitboost model to select and combine different versions of technical indicators and tested them on Euro futures. Though we share a similar objective, there are some significant differences between this article and the aforementioned one.

First, Logitboost in Creamer (2012) is naturally a classification method. Although the probability of its binary prediction could be regarded as a continuous variable, there is neither a direct connection between this probability and the strength of price movements, nor clear evidence of the benefit gained from using the probability instead of the binary prediction. As a comparison, the pdGBM model in this paper generates continuous predictions of price movements. Predictions with relatively larger movements (either positive or negative) and smaller errors could be used for generating trading signals with more confidence. Similar to Logitboost, the output function from pdGBM could be written as a summation of nonlinear response functions of each predictor. Therefore, the prediction from pdGBM is more like a summation of scores, each of which is individually calculated from one predictor. This structure is simpler and more interpretable. At the same time, pdGBM retains the features of automatic predictor selection and the capability of nonlinear pattern recognition. Details are discussed in section 3.3.

Second, an important feature of pdGBM is its introduction of performance-driven loss functions. This feature provides the flexibility to incorporate investors preferences into the model. Investors have different purposes and risk appetites. Some care more about pure cumulative profits and are willing to take more risks, while others prefer to sacrifice the returns partially for lower risks. Considering these differences, the pdGBM model and its corresponding trading strategy not only develop a single model, but also provide a whole prediction and trading system that is capable of incorporating different client-oriented performance preferences. To demonstrate the benefits from using a performance-driven loss function, we compared the performance of this new pdGBM model with the gradient boosting model (GBM) using a L2 loss function and a penalized linear model. As shown in section 6.1, pdGBM significantly outperforms the other models in monthly returns, Sharpe ratios and other performance measurements.

Additionally, we provided extensive out-of-sample tests to examine the robustness of pdGBM's performances. Empirical results show that these abnormal returns remain statistically significant with more conservative transaction cost assumptions, longer holding periods or different base learners.

The rest of this paper is organized as follows. The next section introduces the general predictive problem and the framework of the boosting model. Section 3 presents technical details of the pdGBM model. We first present several different performance-driven functions, and then develop the approximate gradient functions for nondifferentiable loss functions. We also discuss the rationale of selecting a decision stump as the base learner in pdGBM in this section. Section 4 designs a simple systematic trading system based on pdGBM predictions. A step-by-step illustration of model calibration and systematic trading is presented in section 5. More empirical results, comparisons with other models and model robustness are analysed in section 6. The last section concludes and discusses future research directions.

2. Predictive problem and GBM

The model proposed in this paper relies on the boosting framework. In this section, we will briefly introduce the problem solved through boosting method and its general procedure.

2.1. Predictive problem

Consider a predictive problem with N observations and p predictors:

$$\{y_i, \mathbf{x}_i\}_{i=1}^N = \{y_i, x_{i1}, \cdots, x_{ip}\}_{i=1}^N,$$

which are random samples from an unknown population distribution, and the true model is assumed to be

$$Y = F(\mathbf{X}) + \epsilon, \text{ where } E(\epsilon) = 0 \text{ and } Var(\epsilon) < \infty \quad (1)$$

Here, Y could be either binary, categorical or continuous value. A common objective in predictive learning is to estimate the response function $\hat{F}(x) : \mathbf{x} \to y$, which minimizes the expectation of a given loss function $L(y, \hat{y})$:

$$\hat{F} = \arg\min_F \mathbf{E}_{x,y} L(y, F(x)) \quad (2)$$

F is usually restricted in a selected functional space such as order-p polynomial spaces, splines, etc.

2.2. Introduction to GBM

Boosting is one of the most powerful learning methods introduced in the last 20 years. Its most popular version, Adaboost (Freund and Schapire 1995) has proven to be very successful in providing high classification accuracy and avoiding overfitting. It was cited as 'best off-the-shelf classifier in the world' by Breiman (1996).

The basic idea of Boosting is to combine different weak base learners into a powerful ensemble to improve the model performance. After several years of its popularity in the machine learning community, Friedman (2001, 2002) developed a statistical procedure to link the boosting method with a linear additive model, and generalized the method into a regression version, the GBM. To solve the optimization problem in (2), GBM minimizes the sample loss $\frac{1}{n}\sum_{i=1}^{n} L(y_i, F(x_i))$ by a steepest-descending style two-step approximation procedure. It first calculates the negative gradient r_i of the loss function L at each data point, and then fits a base learner on these gradients $(x_i, r_i)_{i=1}^{n}$. The whole procedure is shown in Algorithm 1.

Algorithm 1 Gradient Boosting Model

INPUT: $\{y_i, \mathbf{x}_i\}_{i=1}^{N}$
Initialize $F_0(\mathbf{x}) = \bar{y}$

FOR m = 1 to M DO:

1.1 *Negative Gradient:* compute $\left\{r_{im} = -\left[\frac{\partial L(y_i, f)}{\partial f}\right]|_{f=F_{m-1}(\mathbf{x}_i)}\right\}_{i=1}^{N}$

1.2 Fit a base learner (tree, regression, etc.) $\hat{f}_m(\mathbf{x}; \hat{\theta}_m)$ on $\{(\mathbf{x}_i, r_{im})\}_{i=1}^{N}$

2. Update $F_m \leftarrow F_{m-1} + \lambda * \hat{f}_m$, where λ is the shrinkage factor
END

OUTPUT: $\hat{F}(\mathbf{x}) = F_M$

3. Performance-driven boosting model

Similar to the development procedure in GBM, this paper developed a performance-driven gradient boosting model (pdGBM) equipped with new loss functions based on trading performance measurements in the rest of this section.

3.1. Loss function—profit-and-loss (PnL) and Sharpe ratio

In Algorithm 1, loss function $L(y, f)$ is not specified. Some commonly used functions and their corresponding gradients (Friedman 2001, 2002) are presented in figure 1. With a specified loss function, the general procedure of GBM model implementation is estimating model parameters by minimizing in-sample loss, and then using the calibrated model to make out-of-sample predictions. With a careful model validation method like cross-validation or bootstrapping, the expected prediction error will be approximately minimized. As shown by these graphs, different loss functions assign different loss weights for the tails (large deviations). These differences play a key role in model properties like robustness and efficiency. Statistical properties of these loss functions have been discussed in the literature (Hastie *et al*. 2009, Fan and Lv 2010).

However, for applications in financial markets, none of these loss functions is directly linked with trading performance measurements used by practitioners. It is inconsistent to care about one measurement but use a different measurement in the model. A natural question then arises as to whether we could modify the model calibration procedure by directly optimizing performance measurements? This modification will make the predictive model more understandable and acceptable for investors, and also may generate better trading performances rather than smaller sum squared errors which is optimized in L2 loss.

An intuitive generalization could be constructed from the expected profit-and-loss (PnL). Let us denote $y_t = S_{t+H} - S_t$ as the price movement from time t to $t + H$, $\hat{y}_t$ as model prediction and trading cost as c. Suppose our trading decision is that if the model prediction is significant positive, $\hat{y} > \alpha$ (or negative, $\hat{y} < -\alpha$), investors will long (or short) one share immediately and close the position after H seconds; otherwise, no trade will occur. α is a selected significance level. Then, PnL within the period of $[t, t + H]$ could be calculated as:

$$PnL(y_t, \hat{y}_t; c, \alpha) = \begin{cases} y - c, & \hat{y} \in (\alpha, \infty); \\ 0, & \hat{y} \in [-\alpha, \alpha]; \\ -y - c, & \hat{y} \in (-\infty, \alpha). \end{cases}$$

The corresponding loss function is simply defined as its negative function as shown in figure 2:

$$\begin{aligned} &L_{pnl}(y, \hat{y}; c, \alpha) \\ &\quad = -PnL(y, \hat{y}; c, \alpha) \\ &\quad = -[I(\hat{y} > \alpha) \cdot (y - c) + I(\hat{y} < -\alpha) \cdot (-y - c)] \end{aligned} \quad (3)$$

PnL is a direct measurement of absolute investment returns. Another common performance measurement considering risk-adjusted returns is Sharpe ratio (Sharpe 1998), commonly defined as

$$SR = \frac{\text{Average Return}}{\text{Standard Deviation}}$$

where

sample average return is $A = \frac{1}{N}\sum_{i=1}^{N} PnL(y_i, \hat{y}_i; c, \alpha)$

sample standard deviation is $\sqrt{\frac{\sum_{i=1}^{N}(PnL(y_i, \hat{y}_i; c, \alpha) - A)^2}{N-1}}$

To simplify the function, we introduce $B = \sum_{i=1}^{N} PnL(y_i, \hat{y}_i; c, \alpha)^2/N$, and then the loss function based on Sharpe ratio could be defined as a scaled negative Sharpe ratio, and it is not hard to show that:

$$\begin{aligned} &L_{sr}(y, \hat{y}; c, \alpha) \\ &\quad = -\sqrt{\frac{N}{N-1}} \cdot SR \\ &\quad = -\sqrt{\frac{N}{N-1}} \cdot \frac{\sum_{i=1}^{N} PnL(y_i, \hat{y}_i; c, \alpha)/N}{\sqrt{\sum_{i=1}^{N}(PnL(y_i, \hat{y}_i; c, \alpha) - A)^2/(N-1)}} \\ &\quad = -\frac{A}{\sqrt{B - A^2}} \end{aligned}$$

3.2. Approximated negative gradient

Replacing loss function with $L_{pnl}(y, \hat{y}; c, \alpha)$ or $L_{sr}(y, \hat{y}; c, \alpha)$ in Algorithm 1 is not a trivial step. These specialized loss functions are not differentiable at α and $-\alpha$ because of the indicator function $I(x)$. One solution is replacing the indicator function by a smoothed differentiable function $K(x)$, which minimally satisfies (Kordas 2006, Zheng 2012):

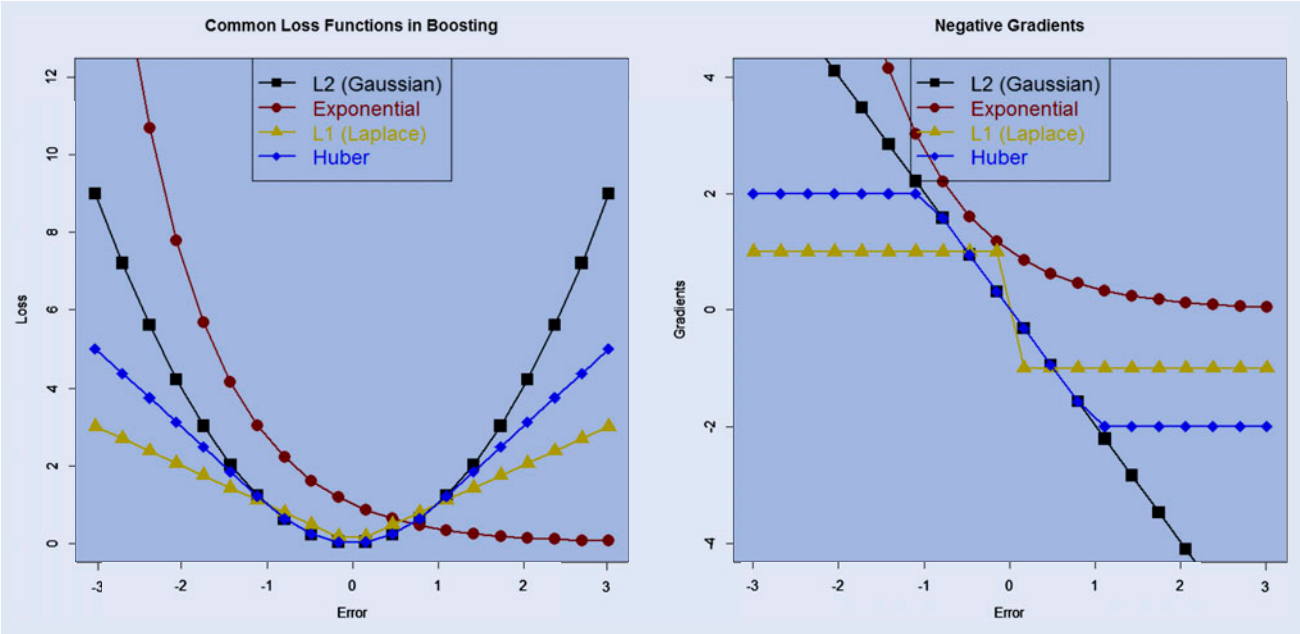

Figure 1. Two different loss functions. The horizontal axis is the prediction error and the vertical axis is the value of the loss function. These loss functions are L2 loss: $L(e) = e^2$, L1 loss: $L(e) = |e|$, exponential loss: $L(e) = exp(-e)$ and Huber Loss: $L(e) = e^2 I(|e| \leq 1) + (2|e| - 1) I(|e| > 1)$.

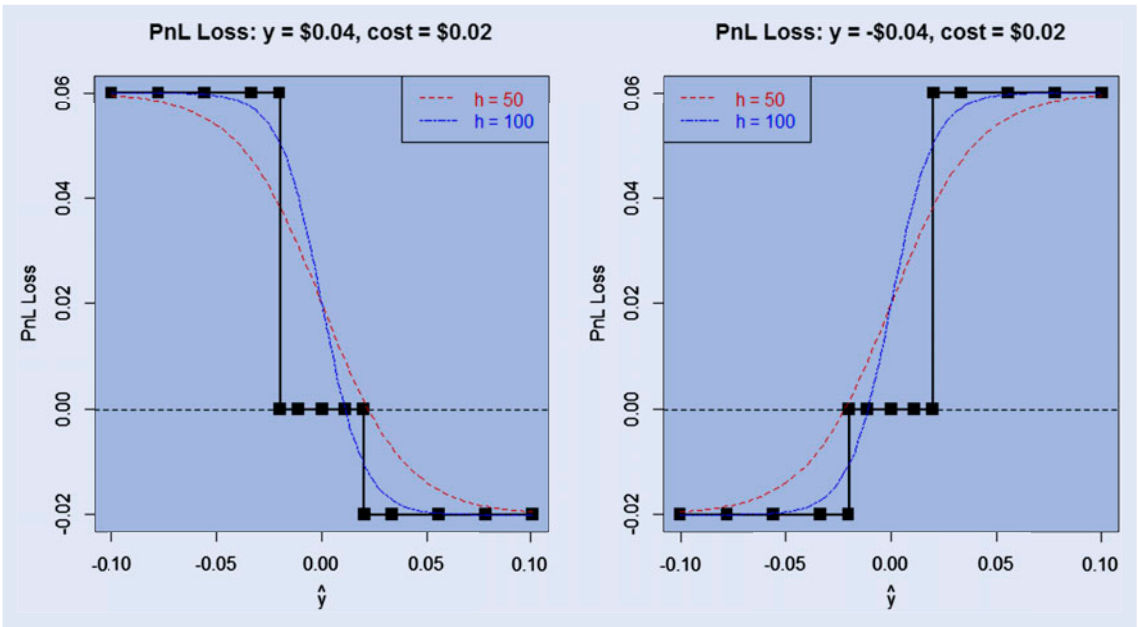

Figure 2. Loss function based on the PnL. The function changes with a different prediction $\hat{y}$. On the left is the graph of the loss function where true target value $y = \$0.04$, and on the right is the one with true value $y = -\$0.04$. Black lines are real PnL loss functions. Blue lines are approximated smoothing function replacing indicator function by logit function $K(x; h) = exp(hx)/(1 + exp(hx))$ with $h = 100$. Red lines use a different parameter $h = 50$. Smaller value of h gives smoother approximation but smaller gradient value around zero. More discussions could be found in section 3.2.

$$K(x) > 0, \lim_{x \to \infty} K(x) = 1, \text{ and } \lim_{x \to -\infty} K(x) = 0$$

There are several different choices. For example, one could use inverse logit function $K(x; h) = \frac{exp(hx)}{1+exp(hx)}$ as used in logistic regression or cumulative normal distribution $K(x; h) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}} e^{-\frac{h^2 x^2}{2}} dx$ as used in the probit model (Pindyck and Rubinfeld 1981). $K(x; h)$ with a smaller h value is smoother, and its corresponding gradient function assigns less weights in the range around zero. Figure 2 shows two different functions with $h = 50$ and $h = 100$. Practically, because most of sample points are centered in the range around zero, the pdGBM with smoother approximated function gives smaller gradient and thus takes more steps to converge. On the other side, h cannot be too big, which will narrow the gradient-descending in a very

tiny range. As we tested, the model performance is insensitive with h between 50 and 100. The selection and property of different approximated negative gradients are not the focus of this paper. We use inverse logit function with $h = 100$ in the rest of this paper.

Remark 1 Rather than the PnL and Sharpe ratio, there are other popular performance measurements such as information coefficient, Sortino ratio that replaces the risk measurement in the Sharpe ratio by downside risk and Sterling ratio that focuses on maximum drawdown. Performance functions (also known as utility functions) have been well studied in the economic literature. For example, Samuelson (1990) used logarithmic and power law utility functions to evaluate simple asset allocation and market timing strategies. Satchell and Timmermann (1995) applied wealth utility and Sharpe ratio as selection criteria for trading systems. The selection of different economic utility functions and its connections with different risk preferences is not the focus here. In this paper, most discussions and results are based on the PnL and Sharpe ratio loss functions.

3.3. Choice of base learner

Another important component in the boosting model is the base learner. The choice of base learner is usually driven by the predictive capability or the expected structure of the response function. Recall that in Algorithm 1, after M iterations, the estimated response function becomes $F_M = F_0 + \lambda \sum_{m=1}^{M} \hat{f}_m(\mathbf{x}; \hat{\theta}_m)$. So the structure of F_M is induced by the structure of $\hat{f}_m(\mathbf{x}; \hat{\theta}_m)$. For example, if $\hat{f}_m(\mathbf{x}; \hat{\theta}_m)$ is restricted to be a linear function $\mathbf{x}B_m$, Algorithm 1 using L2 loss function becomes the procedure of forward stagewise linear regression, which generates a response function as linear regression. Act ually, this L2-boosting model was proposed by Bühlmann and Yu (2003) and shown to be very close to a popular Lasso model that is useful for solving high-dimensional linear problems.

Our pdGBM model chooses the decision stump (Iba and Langley 1992) as base learner, which is the simplest regression tree with two terminal nodes. There are several important reasons for this selection. First, regression tree performs automatic variable subset selection. In the decision stump, the parameter is only one splitting variable and the splitting value. Therefore, in each iteration, the decision stump selects one predictor, not necessarily a different one from other iterations but one that might be split by a different value.

Additionally, boosting model with regression tree could capture the nonlinear patterns that are common in financial data. As shown in Friedman *et al.* (2000), the boosting model using a regression tree as base learner is similar to the model of multiple additive regression splines, which is an extension of linear models that automatically captures nonlinearities of predictors. Theoretically, there exists a sequence of step functions (piecewise constant functions) converging uniformly to any regulated function (the function has both left and right limits for its supporting points). As we will show below, the estimated response function from the pdGBM model with a decision stump is exactly a combination of piecewise constant functions. Therefore, using a decision stump in our model is powerful enough to capture any common nonlinear pattern. Empirically, Friedman *et al.* (2000) also showed that boosting model with a decision tree accurately estimated various nonlinear functions with simulated noises.

There are several popular regression trees like CART, C4.5 or ADT tree as used in Creamer (2012). We selected decision stump because the pdGBM with decision stump exclusively provides more interpretable structures. To explain this interpretability, let us take a look at one example of our estimated response function (details could be found in section 5.3.2):

$$\begin{aligned} F(x) = & -0.00024 + 0.1681 * I(\text{BBands}(20) < -0.45) \\ & + (-0.0003) * I(\text{BBands}(20) \geq -0.45) \\ & + 0.0844 * I(\text{BBands}(15) < -0.37) \\ & + (-0.0002) * I(\text{BBands}(15) \geq -0.37) \\ & + 0.0033 * I(\text{RSI}(7) < 54.13) \\ & + (-0.0065) * I(\text{RSI}(7) \geq 54.13) \\ & + 0.0015 * I(\text{RSI}(10) < 55.56) \\ & + (-0.0031) * I(\text{RSI}(10) \geq 55.56) \\ & + (-0.0001) * I(\text{BBands}(10) < 1.22) \\ & + (-0.0471) * I(\text{BBands}(10) \geq 1.22) \end{aligned}$$

Though the original output function is formatted as a summation of 10 different decision stumps, after rearranging, the response function becomes a linear combination of different score functions based on selected predictors. The score of each predictor is a piecewise constant function, assigning different values on different regions. This is a common structure which practitioners are familiar with. Each score function could be treated as a single trading signal or could be combined into a stronger signal as we use in pdGBM.

One of the biggest differences between decision stump and other trees is that CART or ADT tree consider the interaction of predictors additional to single predictors. For example, as we will compare in section 6.2, the output structure from pdGBM based on the CART tree contains the interaction term of two predictors, BBands20 and RSI7, as shown in figure 3. The performance of models combining relatively uncorrelated factors would gain useful diversification benefits. However, overfitting is the biggest potential problem. It usually comes from two causes, either from overly tuning parameters in the factor or designing overly complex structured factors. Therefore, in practical applications, we should always avoid unnecessary and purely data-driven pattern recognition without fundamental explanation. Rather than automatically introducing interactions by regression trees, we prefer to pre-define a reasonable combined signal as a fuzzy logic function, and then use it as a combined indicator to be included in the pool of potential predictors. Some good examples could be found in Chen *et al.* (2013).

Remark 2 A general guideline for choosing the base learner in boosting models is to choose the base learner with low variance and fewer degree of freedom at the price of larger estimation bias. More discussions can be found in Bühlmann and Yu (2003). We will provide an empirical comparison between using a decision stump as base learner and using a two-level CART tree in section 6.2. Statistically, results show that there is no benefit from using two-level CART tree.

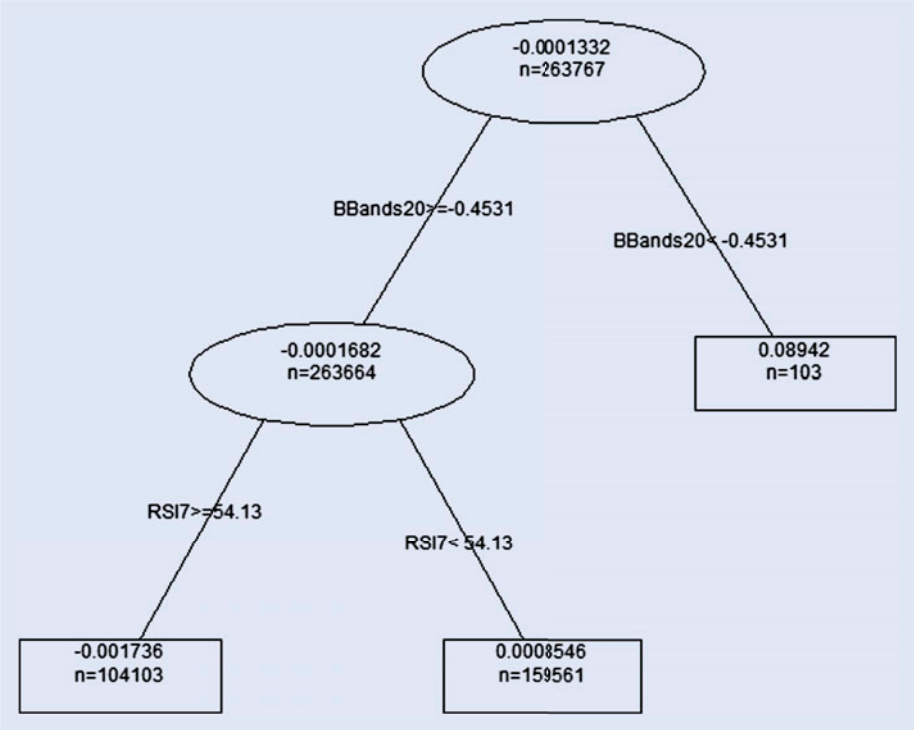

Figure 3. First selected base learner of two-level CART trees based on 2012 December SPY data. More details can be found in section 6.2.

With selected performance-driven loss function and decision stump as base learner, following the general steps in GBM, we finally obtain the pdGBM Model as shown in Algorithm 2.

Algorithm 2 Performance-Driven Gradient Boosting

INPUT: $\{y_i, \mathbf{x}_i\}_{i=1}^N$, $\lambda = 0.1$
Initialize $F_0(\mathbf{x}) = \bar{y}$;
Performance-Driven Loss:

$$L_{pnl}(y, \hat{y}; c, \alpha) = -[I(\hat{y} - \alpha > 0) * (y - c) + I(-\alpha - \hat{y} > 0) * (-y - c)]$$

FOR m = 1 to M DO:

1.1 *Negative Gradient:* compute $r_{im} = -\left[\frac{\partial L(y_i, f)}{\partial f}\right]|_{f=F_{m-1}(\mathbf{x}_i)}$
$= -K'(F_{m-1} - \alpha) * (y - c) - K'(-\alpha - F_{m-1}) * (y + c), i = 1, \ldots, N$

1.2 Fit a decision stump based on $\{(\mathbf{x}_i, r_{im})\}_{i=1}^N$:
$\hat{f}_m(\mathbf{x}; \hat{\theta}_m) = \hat{f}_m^- \cdot I(x < x_m) + \hat{f}_m^+ \cdot I(x \geq x_m)$

2. Update $F_m \leftarrow F_{m-1} + \lambda * \hat{f}_m$
END

OUTPUT: $\hat{F}(\mathbf{x}) = F_M$

4. Systematic trading design

4.1. A simple trading strategy

To evaluate the performance of the new proposed model, we built a simple systematic trading framework based on pdGBM predictions and tested its profitability based on real data. Briefly, we constructed different predictors from historical financial time series and use them as inputs to pdGBM as described in Algorithm 2. After calibrating the model and estimating the output response function, we were able to make a prediction for price movements in the future. This trading system executes a buy order if the model predicts that prices will significantly move up or a sell order if prices will significantly move down in the next H periods. All positions are liquidated immediately after H seconds. The whole procedure is described below:

Systematic Trading Framework:

Input: Pre-selected holding period H; trading cost per round trade c
Predictors: $X_{p,t}$, $p = 1, \ldots, P$;
Significance Level: α

Predictions at time t: $\hat{y}_t^H = \hat{F}(\mathbf{x}) = F_M(\mathbf{x})$;

Entry Signals:
Open one market **buy** order at time t, if $\hat{y}_t^H \geq \alpha$;
Open one market **sell** order at time t, if $\hat{y}_t^H \leq -\alpha$;
No action, otherwise.

Exit Signals:
Liquidate the position (opened at time t) using market order at time t+H.

Output: Profit-and-Loss (PnL) vector: $PnL_t = \begin{cases} y_t - c, & \text{if } \hat{y}_t^H \geq \alpha; \\ -y_t - c, & \text{if } \hat{y}_t^H \leq -\alpha; \\ 0, & \text{otherwise.} \end{cases}$

4.2. Transaction cost and market impact

Transaction cost has been well studied in the literature (e.g. Brennan and Copeland 1988, Kissell 2006). It has been commonly defined as the difference between the execution price assumed in the backtest and the realized execution price in real trade. Transaction cost usually arises from the information acquisition and liquidity taking of orders, and it generally increases when relatively large volumes are put into the market. Different algorithmic trading strategies (e.g. Almgren and Chriss 2001, Almgren *et al*. 2005, Johnson 2010) could be

used to reduce the cost. However, for very liquid products as we tested in this paper, the transaction cost is usually lower than one bid-ask spread, which is approximately \$0.02 per round trade for SPY. Therefore, it is reasonable to assume a constant transaction cost of \$0.02 per round trade throughout this paper. And we will also test the robustness of the pdGBM model under more conservative cost assumptions in section 6.3.

5. Empirical illustrations

We now turn to the back-testing results based on real data. The data employed in this article were obtained from the NYSE Trade and Quote (TAQ) database, which contains quotes and trades information from nationwide exchanges consolidated into National Best Bid and Offer (NBBO) data of all securities listed on the New York Stock Exchange (NYSE), American Stock Exchange (AMEX) and Nasdaq.

5.1. Data preparation

The major data used in this paper are of the market SPDR S&P 500 ETF (NYSE listed symbol: SPY), which tracks one of the most popular indices in the world, the S&P 500 Index. It is one of the earliest and most liquid ETFs. We consolidated all NBBO transaction data of SPY into standard open-high-low-close (OHLC) format by every second. Only data within the regular trading hours (9:30:00, 16:00:00] EST are used to guarantee sufficient liquidity. We also pre-cleaned the data by removing records with zero price or zero trading size and removing records with questionable trading prices (e.g. movements larger than 1% in one tick). A more sophisticated pre-cleaning procedure for TAQ data can be found in Barndorff-Nielsen *et al.* (2009) and Cont *et al.* (2013). An example of cleaned data is illustrated in figure 4.

All back-testing results were generated on a monthly base. Following a similar procedure as in Creamer (2012), in each month, the first 70% of the records were used to train the model, and the following 30% parts were kept for out-of-sample testing. The rest of this section will focus on SPY 2011 December data. The training period has 263 767 valid records from 1 December 2011 9:30 am to 20 December 2011 11:43 am, while the testing period has 113 023 records in the following trading times.

Remark 3 We want to point out that all data used in this paper are before adjustment of dividends and splits. This is actually an irrelevant issue because our trading strategy is for short-horizon trading period without any overnight position.

5.1.1. Target variables. For each data-set, we denote OHLC observations in training data as $\{O_t, H_t, L_t, C_t\}_{t=1}^{N}$. The target variable we are going to predict is defined as forward price movement: $Y_t = C_{t+H} - C_t$. Here, H is the number of time units to hold each trading position. We pre-select H as 10 s. Results using different holding periods are presented in section 6.4.

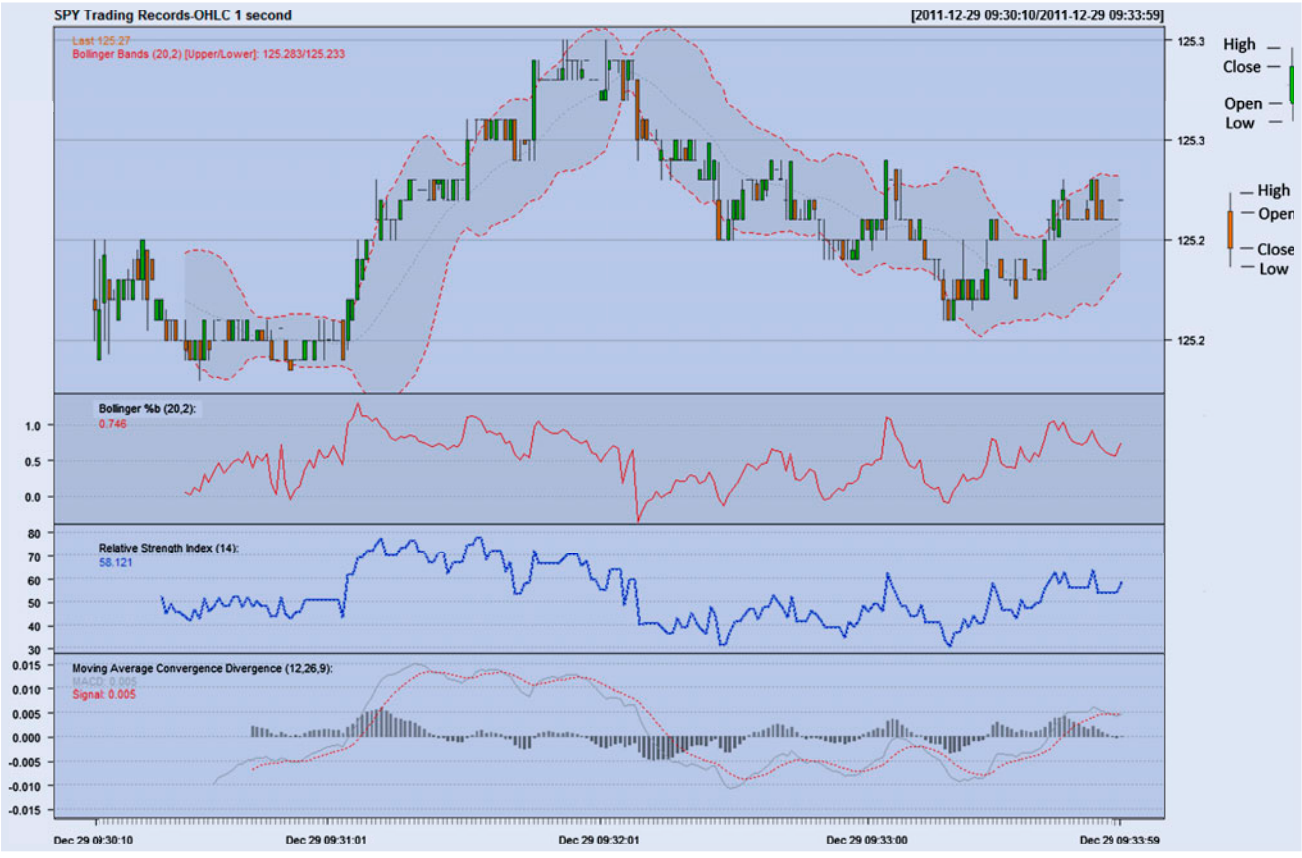

Figure 4. Open-high-low-close stock price at every second interval of SPY trading data within first several minutes of market opening on the last trading day of 2011. Technical indicators of Bollinger Band, relative strength index and moving average convergence divergence are also plotted. Details of their constructions are presented in section 5.1.2.

5.1.2. Predictors. How to construct an efficient predictor has always been a hot topic in both academia and industry. The pool of potential predictors usually include hundreds or even thousands of different factors like fundamental factors as debt to equity ratio and price to book ratio; quant factors like momentums and values, and also many technical indicators.

Over the years, whether technical indicators actually work has been a controversy. For example, Allen and Karjalainen (1999) concluded that using genetic algorithms to combine technical indicators cannot generate consistent excess returns, while Lo, Mamaysky *et al.* (2002) argued that 'technical analysis may well be an effective means for extracting useful information from market prices' by comparing the unconditional empirical distribution of daily stock returns to the conditional distribution (conditional on specific conditions of technical indicators) over 31 years of US stocks data.

Notwithstanding academic sceptics, technical indicators have always been useful tools for practitioners. Usually, each individual technical indicator has different versions with various parameters. For example, moving average (MA) has been popularly used as a trend indicator. Comparing 10-day moving average with 200-day moving average (MA10x200) would identify a much longer term trend than MA10x75 or MA10x40. A comparison of different MA systems that identify trend across commodity and currency markets can be found in Koulajian and Czkwianianc (2010) and Koulajian *et al.* (2014). Different practitioners will choose different set-ups to fit their own trading preferences.

In this paper, our model mainly uses three different types of technical indicators:

- Reversal indicators—relative strength index (RSI):
$$\mathrm{RSI}(n) = 100 * \frac{\mathrm{SMA}(U, n)}{\mathrm{SMA}(U, n) + \mathrm{SMA}(D, n)}$$
where $U = \{(C_t - C_{t-1}) \cdot I(C_t \geq C_{t-1})\}$, $D = \{(C_{t-1} - C_t) \cdot I(C_t < C_{t-1})\}$; $SMA(x, n)$ is the simple average of $x_t, x_{t-1}, \ldots, x_{t-n+1}$;
RSI indicator compares the recent uptrend sizes with downtrend sizes. It is commonly used as a reversal indicator. From the definition, the RSI ranges from 0 to 100. RSI above 70 level is explained by traders to be an overbought signal. Likewise, RSI below 30 level is an oversold signal.
- Momentum and trend indicators - moving average cross divergence (MACD):
$$\mathrm{MACD}(n, m, k) = \mathrm{SMA}(\mathrm{SMA}(C_t, n) - \mathrm{SMA}(C_t, m), k)$$
where $SMA(x, n)$ is defined as above;
MACD is a popular indicator comparing shorter term moving average with longer term moving average of the price. Normally, a positive value of MACD indicates a recent uptrend and a higher chance that the market will continue the trend for a while.
- Volatility indicators - Bollinger Bands (BBands):
$$\mathrm{BBands}(n) = [C_t - LowerBand(C, n)]/[Range(C, n)]$$
where $LowerBand(C, n) = SMA(C, n) - 0.5 \cdot Range(C, n)$; $Range(C, n) = 4 \cdot \sigma(C, n)$; $\sigma(C, n)$ is the sample standard deviation of $C_t, C_{t-1}, \cdots, C_{t-n+1}$.
BBands is an indicator based on both moving average and its standard deviation. It provides a relative position of current price in the range of estimated reasonable high and low price boundary.

Remark 4 For parameters in technical indicators, some common recommendations by practitioners are RSI(14), MACD(12, 26, 9) and BBands(20). Figure 4 provides some examples of them based on second-by-second data. To add more flexibility, we create more predictors using different parameters and let the pdGBM automatically select and combine them. Specially, we set-up the pool of predictors as RSI($14k$), MACD($12k$; $26k$; 9) and BBands($20k$), $k = 0.5, 0.75, 1, \ldots, 10$. Sufficient back-testing results as we will discuss later show that technical indicators provide useful predictive information to consistently generate excess returns.

5.2. Data visualization

In a journey of statistical discovery, we always start with data visualization. Figure 5 shows scatter plots between the target variable and some predictors. We can observe that some non-ignorable negatively correlated pattern exists when predictors are within a certain range of values, as highlighted in these red boxes. Directly applying linear regression usually cannot provide any useful results because of the extremely noisy data in the middle range. An alternative method might be fitting separate regressions within pre-split regions. However, the pre-splitting region is very subjective. Regression tree is known for its availability of capturing nonlinear patterns and its robustness to noisy data. This is exactly what motivated us to start this research in the first place.

5.3. Model calibration

After preparing target variable and predictors, we fit our pdGBM onto the training data. As discussed before, we used the shrinkage parameter $\lambda = 0.1$ as suggested in Hastie *et al.* (2009).

5.3.1. Cross validation for optimal M. The number of iterations M is optimally selected by a K-fold cross-validation procedure with $K = 5$. For different iteration numbers M, we calculated the cross-validation error $CVerror(M)$ as follows: the whole training set is randomly split into K parts. For each part, the pdGBM model was trained on the other $K - 1$ parts and the loss function was calculated using the Kth part. Averaging these K values, we got the value of $CVerror(M)$. $\hat{M} = \arg\min\ CVerror(M)$ was selected for the final model implementation. Figure 6 shows the graph of $CVerror(M)$. The optimal number of iteration, $M = 30$, is selected for further testing.

5.3.2. The structure of output function. The output F_m described in Algorithm 2 seems complex as a summation of

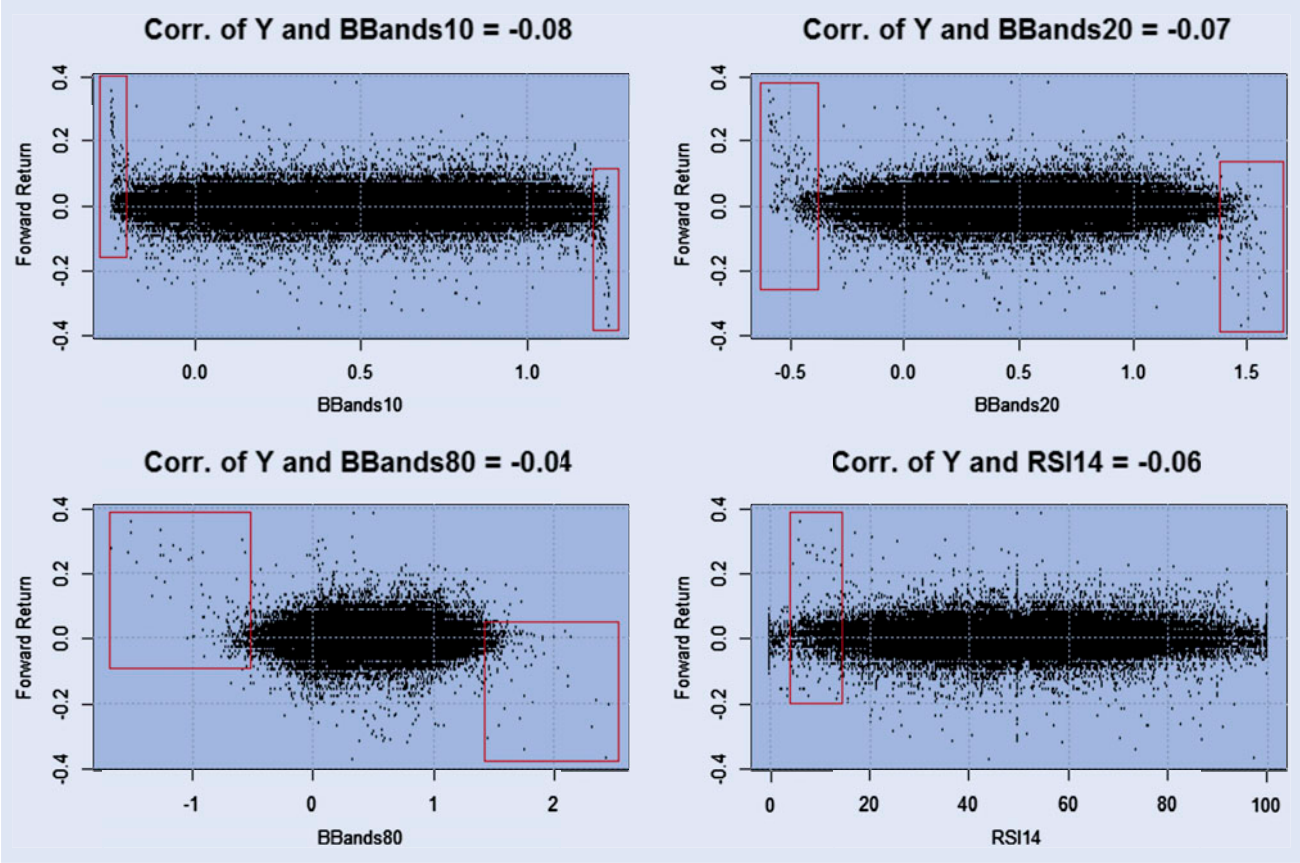

Figure 5. Scatter plots between target variable (price movement in the next 10 s) and several predictors. Taking BBands20 as an example, we can observe that the negative correlation exists for the region of predictors around −0.5 and 1.5 as circled by red boxes, while there is no clear pattern in the middle region. Given its nonlinear but interpretable output and its accuracy and robustness, pdGBM is expected to be a good candidate fitting these data.

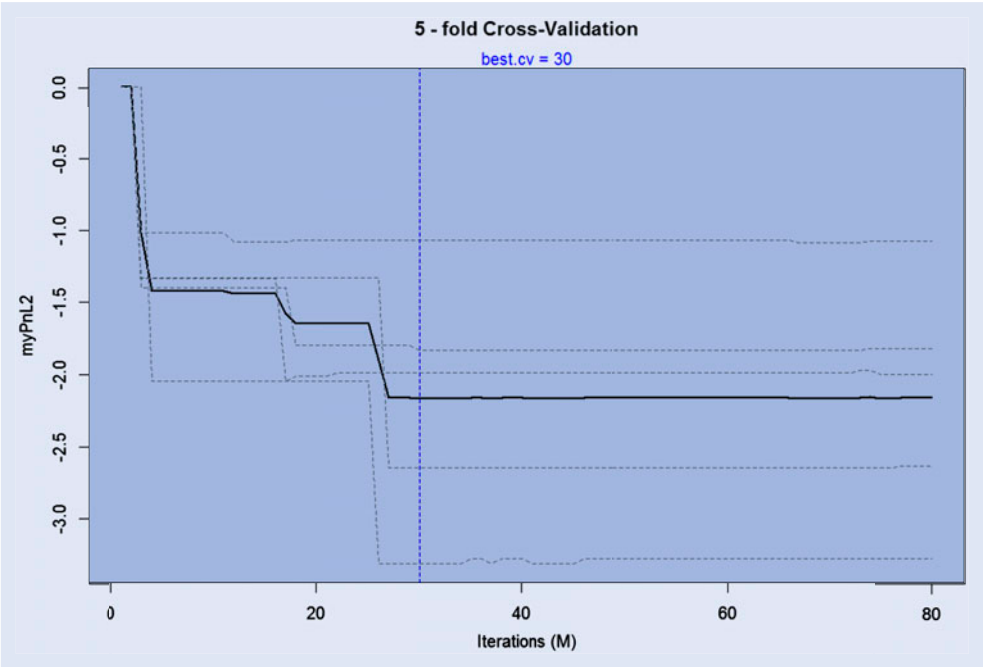

Figure 6. We used fivefold cross validation to determine the optimal iteration number in Algorithm 2. The training (in-sample) data were randomly divided into five parts. For each part, the pdGBM model was trained using the other four data parts with a given value of M, and the loss was calculated using the fifth part. Cross-validation error was defined as the average of these five loss values. We kept evaluating $CVerror(M)$ as $M = 1, 2, \ldots$ until $CVerror(M)$ no longer improve in the last 50 rounds. The black solid line represents $CVerror(M)$ as the average of grey dot lines, which represent training errors on each cross validation part. The value with the smallest CV error was selected for the final model ($M = 30$).

M different base learners. However, when we use the decision stump as the base learner, there is no interaction term and each base learner is only a two-value piecewise function of one selected predictor $x_{i(m)}$. Therefore, instead of the summation of these base functions, we can rewrite the response function as

$$F(\mathbf{x}) = F_0 + \sum_{m=1}^{M} \lambda \hat{f}_m(x_{i(m)}) = F_0 + \lambda \sum_{p=1}^{P} \hat{F}_p(x_p)$$

where F_0 is the averaged target value in the training data. For each selected predictor x_p, $\hat{F}_p(x_p) = \sum_{m:\, i(m)=p} \hat{f}_m(x_p)$ is the summation of all base learners which are based on this predictor. Now the final output function could be interpreted as a sparse additive structure based on P fitted response functions of selected predictors. Each iteration in pdGBM is actually selecting one predictor and refining the response function $\hat{F}_p(x_p)$. Only these predictors selected in these M iterations have nontrivial response functions. This also explains why pdGBM is able to automatically select subset predictors.

To help readers understand this procedure, figure 7 shows how response functions grow with more iterations. We only list graphs of these 6 predictors (out of 36 predictors) which are selected into the final output function. The whole iterative procedure is as follows:

- After the first iteration:
  as shown in the left panel, $BBands(20)$ turns out to be the first most significant predictor and -0.45 is selected as the splitting value. Scores on the left region and right region are estimated by the subsample means, and the response function is:

$$\begin{aligned} F(x) &= F_0 + \hat{f}_1 \\ &= -0.00024 + 0.0894 \\ &\quad \cdot I(BBands(20) < -0.45) + (-0.0002) \\ &\quad \cdot I(BBands(20) \geq -0.45) \end{aligned}$$

  This is consistent with the traditional usage of the Bollinger Band as a contrarian indicator, treating a close price below the band (which happens when $BBands(20) < 0$) as a signal for rebound. We can see that the pdGBM model selects a threshold as low as -0.45, which is roughly 0.01
- After 10 iterations:
  as shown in the middle panel, 5 predictors out of 36 potential ones are selected in these 10 iterations. The response function becomes:

$$\begin{aligned} F(x) = {} & -0.00024 + 0.1681 * I(\text{BBands}(20) \\ & < -0.45) + (-0.0003) * I(\text{BBands}(20) \\ & \geq -0.45) \\ & + 0.0844 * I(\text{BBands}(15) < -0.37) \\ & + (-0.0002) * I(\text{BBands}(15) \geq -0.37) \\ & + 0.0033 * I(\text{RSI}(7) < 54.13) \\ & + (-0.0065) * I(\text{RSI}(7) \geq 54.13) \\ & + 0.0015 * I(\text{RSI}(10) < 55.56) \\ & + (-0.0031) * I(\text{RSI}(10) \geq 55.56) \\ & + (-0.0001) * I(\text{BBands}(10) < 1.22) \\ & + (-0.0471) * I(\text{BBands}(10) \geq 1.22) \end{aligned}$$

- Final output:
  As shown in the right panel, after the optimal number ($M = 30$) of iterations, the final pdGBM selects six predictors and the final response function is the summation of these six nonlinear response functions. Though only one additional predictor is added into the final output function, functions of the other five predictors are all refined during the last 20 iterations.

5.4. Trading records and performances

Now that the estimated response function is ready, we could simply construct the out-of-sample test following the procedure as presented in section 4. Figure 8 shows graphical trading records based on SPY 2011 December data. The grey curve is the time series plot of SPY's close prices at every second. Green arrows indicate time points of an opening market order to buy one share, while red arrows are times of opening one market order to sell one share. All positions are held for exactly 10 s. The blue curve is the cumulative returns (the axis is on the right edge). Corresponding trading performances are reported in table 1. Overall, this strategy generates 70 trades in the testing period, with a 0.045% profit per contract and a 19.13 annualized Sharpe ratio after transaction cost ($0.02 per trade).

6. Robustness and comparisons

In this section, we will analyse the superior performance of the pdGBM model by comparing it with other predictive models and examining its robustness in a number of dimensions including different base learners, more conservative transaction cost assumptions and different holding periods.

6.1. Comparison with other boosting models

This subsection compares pdGBM with GBM and a linear model (LM). As we discussed before, the pdGBM is generalized from the GBM model by replacing quantitative loss functions with performance-driven loss functions. Therefore, we use a standard GBM with L2 Loss function as a benchmark to check the benefits of introducing the new loss function. Another standard benchmark model is the linear model. To avoid the overfitting, L-1 penalized linear model, also known as Lasso (Tibshirani 1996), is implemented here. We evaluated and compared different performance measurements as summarized in table 2.

Results show that both pdGBM and GBM generate significant positive out-of-sample profits. It is worth mentioning that GBM results have better R^2 and smaller SSE in both testing samples and selected samples (testing samples with trading signals). This is as expected because the L2 loss function in GBM is more relevant with these statistical measurements. As a comparison, pdGBM outperforms GBM in all trading performance evaluations such as profits per trade, Sharpe ratio, maximum drawdown, etc. This is exactly our target of developing this pdGBM model with performance-driven loss functions.

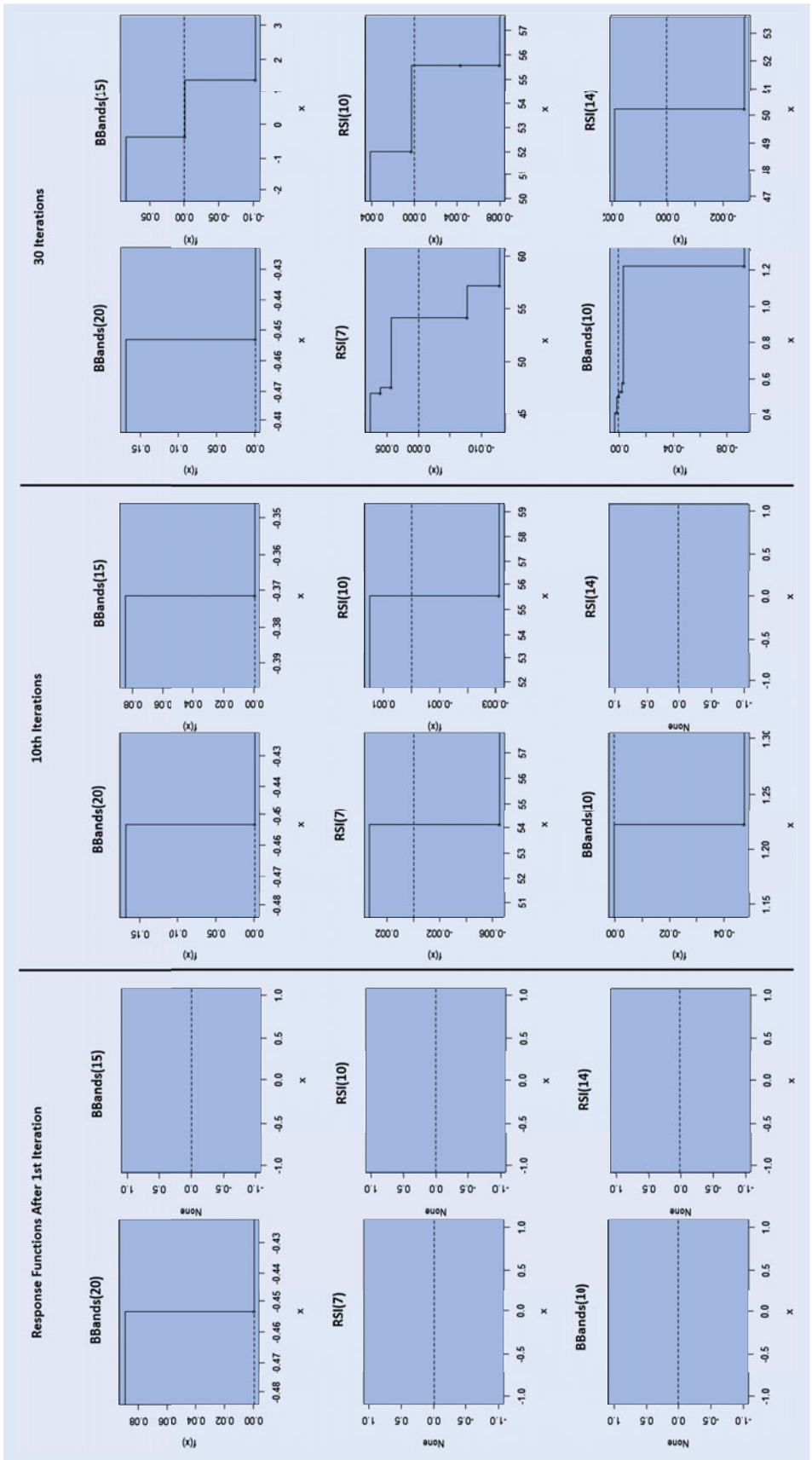

Figure 7. Response functions after different iterations. pdGBM with a base learner of decision stump could be interpreted as a summation of response functions, each of which is a piecewise constant function of one predictor. This figure shows how response functions grow with more iterations. As discussed in section 5.3.1, after an optimal numbers ($M = 30$) of iterations, out of 36 different indicators, BBands(30), BBands(15), RSI(7), RSI(10) and BBands(10) are selected in final response function.

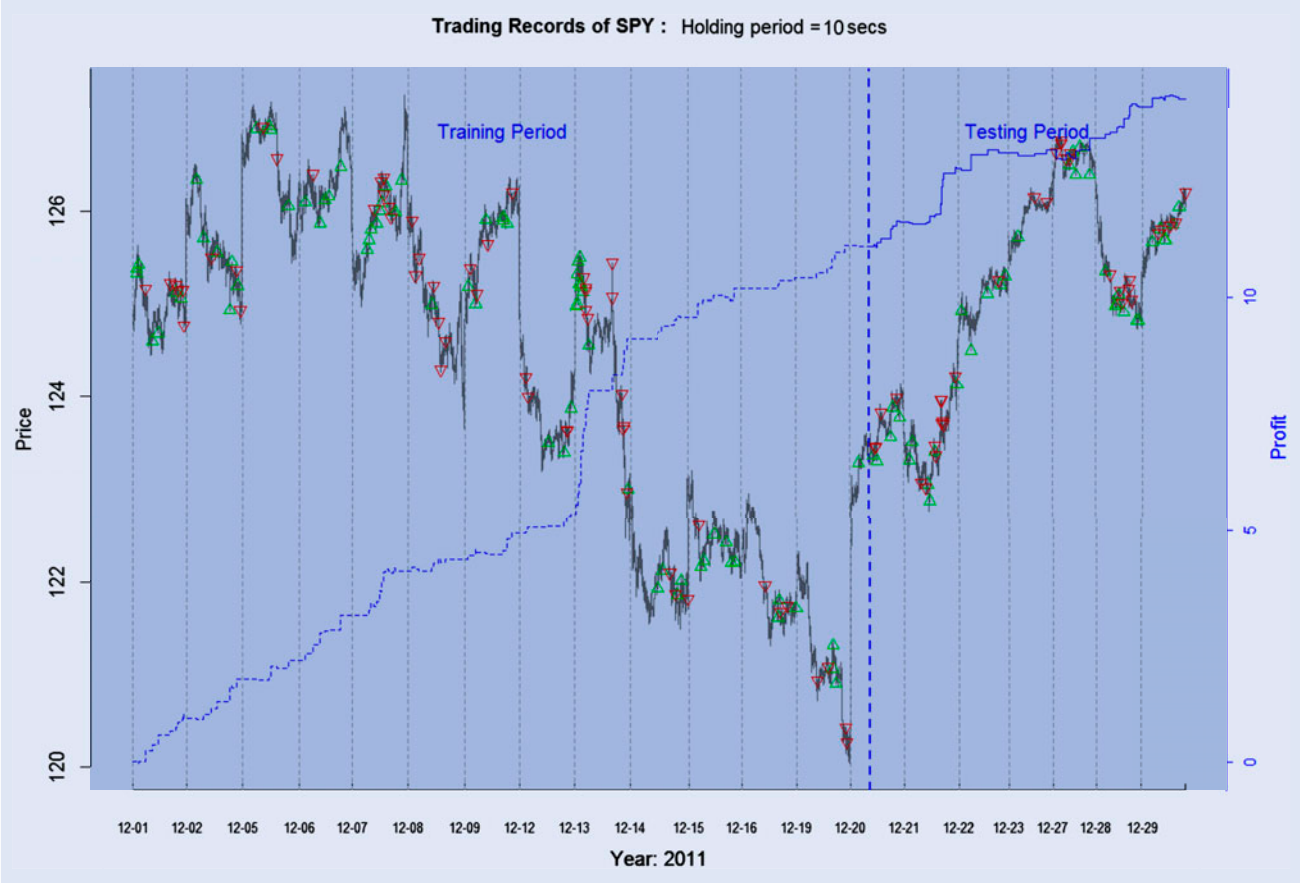

Figure 8. Trading records of pdGBM based on 2011 December SPY data. This figure shows the model performance of the trading strategy in as discussed in section 4. The data in the training period (263 767 records from 1 December 2011 9:30 am to 20 December 2011 11:43 am) are used to fit the model, while the testing period (113 023 records in following trading times in December 2011) is used to evaluate out-of-sample model performance. The grey curve is the time series of SPY's secondly transaction prices. Green arrows indicate time points of buy order, while red arrows are time points of sell orders. All positions are held for exactly 10 s and then liquidated by market order. The blue line is the cumulative profit-and-loss (PnL) plot with the scale on the right axis. In summary, the pdGBM model generates 70 out-of-sample trades, with 0.045% profit per trade, 3.54% total profits and 19.15 annualized Sharpe ratio after \$0.02 $\approx$ 1.5 bps transaction cost.

Table 1. Trading performances of pdGBM for SPY 2011 December.

Target Variable: 10 s forward price move
Predictors: $RSI(14k)$, $MACD(12k, 26k, 9)$, $Bbands(20k)$, $k = 0.5, 0.75, 1, \ldots, 10$
Training: (%70 of full sample) 263767 records from 1 December 2011 09:30 am to 20 December 2011 11:43 am
Testing: (%30 of full sample) 113023 records from 2011-12-20 11:44 am to 2011-12-30 4:00 pm

Training period (In-sample)				Testing period (Out-of-sample)			
Sharpe	PnL(%)	*n*.Long	*n*.Short	Sharpe	PnL(%)	*n*.Long	*n*.Short
13.77	11.09	72	55	19.15	3.18	36	34
Performance measurements							
avg-PnL	W Ratio	W/L Ratio	MDD	SSE	R^2	adj-SSE	adj-R^2
0.045	58%	4.43	0.21	44.11	1.4%	0.21	32 %

Notes: This table shows trading records and some common performance measurements generated from Algorithm 2 and figure 8. During the out-of-sample period, pdGBM generates 70 trades, with 36 buy orders (*n*.Long) and 24 short orders (*n*.Short). Winning ratio (W Ratio), which measures the percentage of wining trades out of total trades, is 58%; win to loss ratio (W/L Ratio), which is the ratio of profits from wining trades to the losses from losing trades, is 4.43. Such a high number means we make much more profit in these wining trades than losses in losing trades, though the wining ratio is not very high. Maximum drawn, which measures the peak-to-valley loss, is 0.21% or 6.6% of total profits; sum square error (SSE) of pdGBM predictions on all 113 023 testing samples is 44.11 and the corresponding *R*-square is 1.4%; adjusted sum square error (adj-SSE) defined as the SSE of pdGBM predictions on these 70 data points with trading signals is 0.21, and the corresponding *R*-square is 32%.

With a reasonable trading cost assumption of \$0.02 per trade, LM cannot even generate positive profits in the training period. We also tried other generalized linear regressions and the principle component regression. Unfortunately, none of them could provide positive returns. The biggest difference between pdGBM or GBM with these linear models is that the

Table 2. Monthly out-of-sample trading performances of four different models based on SPY 2011 data.

	pdGBM (Stump)			pdGBM (CART)			GBM			LASSO		
Month	Sharpe	PnL(%)	*n*.Trade	Sharpe	PnL(%)	*n*.Trade	Sharpe	PnL(%)	*n*.Trade	Sharpe	PnL(%)	*n*.Trade
1	10.63	1.74	73	10.63	1.74	73	−2.41	−0.27	60	NA	0.00	0
2	14.34	2.68	68	14.47	2.72	69	9.85	1.68	75	NA	0.00	0
3	18.88	5.46	81	18.52	5.62	87	2.57	0.25	110	−1.71	−0.07	97
4	17.93	2.35	53	18.70	2.29	42	7.10	0.75	39	NA	0.00	0
5	16.34	4.54	70	10.89	4.29	150	2.90	1.28	184	NA	0.00	0
6	8.56	1.45	39	10.58	2.00	62	4.06	0.70	99	−9.15	−1.13	139
7	15.76	3.49	105	11.64	2.05	147	9.26	2.05	220	−5.98	−0.59	22
8	14.94	2.65	96	7.17	2.61	154	−11.72	−6.94	383	NA	0.00	0
9	−7.20	−4.72	262	−4.83	−2.06	279	−11.96	−12.84	710	−2.07	−0.10	9
10	14.36	1.95	40	9.51	2.59	132	17.74	6.25	249	3.60	0.18	24
11	16.10	3.24	34	24.63	4.49	84	4.05	2.33	208	−5.63	−2.12	83
12	19.22	3.18	70	19.03	3.07	101	6.30	1.07	299	−8.14	−2.41	118
Mean		2.33**			2.62**			−0.31			−0.52*	
t statistics		[3.10]			4.61			−0.21			−1.93	
Comparisons with pdGBM (Stump):				−0.74	0.28		−10.18**	−2.64**		−17.47**	−2.85**	
				[−0.59]	[1.00]		[−4.65]	[−2.51]		[−6.70]	[−3.58]	

Notes: This table reports the out-of-sample average monthly returns (PnL), annualized Sharpe ratio (Sharpe) and total number of trades (*n*.Trade) from four different models. pdGBM (Stump) is the standard pdGBM model as introduced in section 3.3; pdGBM (CART) uses the two-level CART tree as base learner; GBM is the standard gradient booting model with L2 loss function and base learner of decision stump (Friedman 2002); LASSO (Tibshirani 1996) is the linear model with L1 penalty for automatic variable selection. For each month, first 70% of samples are used to train the model, while the following 30% samples are used to evaluate out-of-sample model performance. Data samples are 2011 SPY transaction data during trading hours from TAQ database, consolidated by every second. Transaction costs are assumed to be \$0.02 per round trade. *t*-test statistics are used to compare their performances. Test statistics are reported in square brackets and significance at the 1 and 5% levels is given by an ** and an *, respectively.

Table 3. Trading performances under different transaction cost assumptions.

	Cost = \$0.02			Cost = \$0.03			Cost = \$0.04		
Month	Sharpe	PnL(%)	*n*.Trade	Sharpe	PnL(%)	*n*.Trade	Sharpe	PnL(%)	*n*.Trade
1	10.63	1.74	73	7.93	0.79	34	4.99	0.01	34
2	14.34	2.68	68	11.68	1.07	31	8.89	0.02	31
3	18.88	5.46	81	11.53	2.05	52	9.14	0.03	55
4	17.93	2.35	53	6.39	0.47	35	1.77	0.00	35
5	16.34	4.54	70	11.52	2.30	19	11.53	0.12	18
6	8.56	1.45	39	9.23	1.21	15	8.60	0.07	15
7	15.76	3.49	105	11.66	0.95	51	6.20	0.01	51
8	14.94	2.65	96	0.00	0.00	0	0.00	0.00	0
9	−7.20	−4.72	262	6.37	0.66	15	0.00	0.00	0
10	14.36	1.95	40	0.00	0.00	0	0.00	0.00	0
11	16.10	3.24	34	15.36	2.98	31	14.16	0.09	31
12	19.22	3.18	70	17.80	1.37	36	14.61	0.03	37
Mean		2.33**			1.15**			0.03**	
t statistics		[2.70]			[3.66]			[2.25]	

Notes: This table reports the out-of-sample average monthly returns (PnL), annualized Sharpe ratio (Sharpe) and total number of trades (*n*.Trade) from pdGBM model under different round trade transaction costs. Data samples are 2011 SPY transaction data during trading hours from TAQ database. For each month, first 70% of samples are used to train the model, while the following 30% samples are used to evaluate out-of-sample model performance. Test statistics are reported in square brackets and significance at the 1 and 5% levels is given by an ** and an *, respectively.

previous two models are capable of fitting nonlinear patterns between predictors and the target variable. This again provides evidence of the importance of introducing nonlinear models to solve predictive problems in similar financial problems.

6.2. Comparison of different base learners

Table 2 also compares results from pdGBM using different base learners. Our standard pdGBM uses a decision stump, which is the simplest regression tree with two terminal nodes. As we discussed in section 3.3, pdGBM using a decision stump is already capable of capturing most nonlinear patterns and provides interpretable response function. To check if it is necessary to use a base learner with more complex structure, we also implemented a pdGBM model with a two-level CART tree (Breiman *et al*. 1984) as a comparison.

Results in table 2 show that the difference in either monthly return or Sharpe ratio is insignificant by statistical t-test. Compared with pdGBM using a decision stump, the output function from pdGBM using a two-level CART tree includes many interacted terms from different predictors. These complex structures are dangerous enough to have potential overfitting and noninterpretable issues. Therefore, we suggest using the

Table 4. Trading performance from different fixed holding periods.

	$H = 10$ s			$H = 30$ s			$H = 60$ s			$H = 120$ s			$H = 300$ s		
Month	Sharpe	PnL(%)	*n*.Trade	Sharpe	PnL(%)	*n*.Trade	Sharpe	PnL(%)	*n*.Trade	Sharpe	PnL(%)	*n*.Trade	Sharpe	PnL(%)	*n*.Trade
1	10.63	1.74	73	8.98	1.99	89	16.29	1.60	28	−2.41	−2.05	377	−19.88	−66.67	2362
2	14.34	2.68	68	6.22	2.61	197	12.47	3.14	55	9.85	1.48	131	−4.97	−34.42	3665
3	18.88	5.46	81	5.24	6.77	531	20.28	4.59	79	2.57	−1.87	267	−4.91	−7.23	208
4	17.93	2.35	53	1.01	2.92	1047	10.48	1.57	67	7.10	−4.38	219	9.56	6.65	127
5	16.34	4.54	70	3.00	4.46	293	5.22	2.55	306	2.90	−0.65	312	−10.11	−18.36	555
6	8.56	1.45	39	11.22	1.48	94	−4.42	−4.07	313	4.06	−31.16	1279	3.86	72.77	3109
7	15.76	3.49	105	−4.42	−4.07	313	11.22	1.48	94	9.26	−7.05	407	−4.77	−19.61	3279
8	14.94	2.65	96	5.22	2.55	306	3.00	4.46	293	−11.72	9.70	593	−15.92	−928.85	30 929
9	−7.20	−4.72	262	10.48	1.57	67	1.01	2.92	1047	−11.96	29.96	1550	−3.87	−91.03	5916
10	14.36	1.95	40	20.28	4.59	79	5.24	6.77	531	17.74	1.90	452	−9.70	−287.93	9048
11	16.10	3.24	34	12.47	3.14	55	6.22	2.61	197	4.05	9.35	287	−3.47	−41.71	6249
12	19.22	3.18	70	16.29	1.60	28	8.98	1.99	89	6.30	0.65	50	−5.49	−68.42	7552
Mean		2.33**			2.47**			2.47**			0.49			−123.74	
t statistics		[2.89]			[3.19]			[2.83]			[0.10]			[−1.34]	

Notes: This table reports the out-of-sample average monthly returns (PnL), annualized Sharpe ratio (Sharpe) and total number of trades (*n*.Trade) from pdGBM model with different pre-fixed holding periods. Test statistics are reported. Results show that 10-, 30- or 60- s forward price movements are predictable by pdGBM model, with 11 out of 12 months are profitable. Holding period of 120 s or 300 s still generates positive average monthly returns. However, *t*-test statistics are not significant for these two cases. Data samples are 2011 SPY transaction data during trading hours from TAQ database. For each month, first 70% of samples are used to train the model, while the following 30% samples are used to evaluate out-of-sample model performance.

Table 5. Comparisons of two different performance-driven loss functions.

	PnL Loss			SR Loss		
Month	Sharpe	PnL(%)	*n*.Trade	Sharpe	PnL(%)	*n*.Trade
1	10.63	1.74	73	10.29	1.13	34
2	14.34	2.68	68	13.98	1.38	31
3	18.88	5.46	81	13.70	2.57	52
4	17.93	2.35	53	9.96	0.82	35
5	16.34	4.54	70	11.76	2.51	18
6	8.56	1.45	39	9.76	1.36	15
7	15.76	3.49	105	14.80	1.47	55
8	14.94	2.65	96	0.00	0.00	0
9	−7.20	−4.72	262	24.94	1.04	40
10	14.36	1.95	40	14.36	1.95	40
11	16.10	3.24	34	16.10	3.24	34
12	19.22	3.18	70	21.09	1.73	36
Mean		1.97			1.66	
Standard error		2.86*			0.97**	
t statistics		[2.28]			[5.68]	
Comparisons with pdGBM (Stump):				0.78	−0.67	
				[0.02]	[−1.12]	

Notes: This table reports the out-of-sample average returns in dollar value (PnL), annualized Sharpe ratio (Sharpe) and total number of trades (*n*.Trade) from two different versions of pdGBM. One is using PnL loss function (PnL Loss) as defined in (3); another is using Sharpe ratio loss function (SR Loss) as defined in (4). Data samples are 2011 SPY transaction data during trading hours from TAQ database. For each month, first 70% of samples are used to train the model, while the following 30% samples are used to evaluate out-of-sample model performance. Transaction costs are assumed to be \$0.02 per round trade. *t*-test statistics are used to compare their performances. Test statistics are reported in square brackets and significance at the 1 and 5% levels is given by an ** and an *, respectively.

simplest decision stump as base learner in the pdGBM model. Whenever it becomes necessary, we could introduce reasonable interacted terms as pre-defined fuzzy logics (Chen *et al.* 2013) and use them as predictors directly.

Remark 5 Our implementation of this regression tree was through the rpart package in R language. This package is flexible in controlling tree splitting criteria and many other control parameters. For more details, refer to Therneau *et al.* (1997).

6.3. Robustness to transaction costs

Table 3 reports performances of the pdGBM model under different transaction cost assumptions. For a very liquid product like SPY, the transaction cost is usually lower than 1 bid-ask spread, which is roughly \$0.02 per round trade. Results in table 3 show that even with a transaction cost assumption as high as \$0.04 per trade, pdGBM is still profitable with 3% monthly returns on average. We can also observe that larger transaction cost assumptions reduce the number of trades, from 991 total

trades with an assumption of \$0.02 cost to 307 trades with an assumption of \$0.04 cost.

It is interesting to see that returns under a cost assumption of \$0.03 has a 0.0434% profit per trade, which is higher than the 0.0283% profit per trade from an assumption of \$0.02 cost. One simple reason is that the pdGBM model incorporates transaction costs into the loss function and adjusts trading decisions accordingly. Though we assume a constant transaction cost, the cost from real trading might be totally different. That means if we are assuming a \$0.03 cost per trade while real cost turns to be less than \$0.03 per trade, then the realized trading performance will be much better than directly using a \$0.02 cost assumption. This is another new feature of pdGBM that a conservative transaction cost assumption might be able to filter out relative weak trading signals. Certainly, an over-conservative assumption of transaction cost will kill most of our trades and profits.

6.4. Results from different holding periods

Our simple trading strategy based on pdGBM assumes a pre-fixed time-horizon for holding a trading position. Previous results already show the predictability of short time-horizon (10-s) price movements. This section analyses results from different pre-selected holding periods in table 4. It is clear that 10-, 30- or 60-s forward price movements are all predictable by the pdGBM model, with 11 out of 12 months profitable and with more than 2% monthly returns on average. Holding period of 120 s or 300 s seems to be too long for pdGBM to predict. However, this does not mean price movements over these longer holding periods are not predictable by the pdGBM model. We need to keep in mind that we just used three most popular types of technical indicators without any sophisticated trading signal constructions. Methods of constructing more useful predictors from market information, sentiment information and cross-sectional information will be studied in the future work.

6.5. Different performance-driven loss functions

Finally, as we discussed before, different performance-driven loss functions could be incorporated into the pdGBM model, including pure cumulative returns like PnL, risk-adjusted returns like Sharpe ratio or other measurements. In table 5, we report results from the PnL and Sharpe ratio loss functions as defined in (3) and (4). It is clear that both models generate significantly positive profits, with more than 1.6% monthly returns on average. Statistical *t*-test was used to compare out-of-sample Sharpe ratio and PnL from pdGBM models based on these two different loss functions. From these results, the pdGBM model based on PnL loss provides higher averaged monthly returns, and the pdGBM model based on Sharpe ratio loss generates a higher Sharpe ratio on average. However, hypothesis tests show that these differences are not significant at a 5% level. The reason may be that both of them measure investment returns but from different angles: one in absolute measurement and another in risk-adjusted measurement.

7. Conclusion

In this article, we introduced an innovative nonlinear predictive model, pdGBM, to forecast short-horizon price movements and outlined a simple high-frequency systematic trading strategy based on the model predictions. Empirical results have shown that the pdGBM model provides superior performance compared to the GBM with L2 loss and linear models. For investors with different investment philosophies or risk appetites, the pdGBM model and its corresponding trading system could be equipped with customized performance functions.

Given this established framework, our research may be extended in several ways. One direction for further research is to bring in more predictors like other technical indicators, sentiment information and cross-sectional information. Additionally, as widely accepted and studied in the literature (e.g. Wahal and Yavuz 2013), some style variables like size and the book-to-market ratio could also be useful in predicting stock returns. Therefore, there is a huge pool of potential predictors for discovery. When there is a large number of potential predictors, it comes with an interesting question that whether predictor pre-screening will be helpful to avoid overfitting and improve model accuracy. These ideas have recently been studied in the statistical literature (e.g. Fan and Lv 2008, Fan *et al*. 2010, Bühlmann *et al*. 2013). These might be useful add-on features of the pdGBM model.

Another possible extension of this research is portfolio optimization on top of our trading strategies with different trading set-ups. As we have shown in section 6.4, different pre-selected holding periods like 10-, 30- or 60-s are all profitable with a trading strategy based on pdGBM. Combining strategies with different trading horizons might diversify risk and improve the overall risk-adjusted returns.

Acknowledgements

The authors would like to thank Ionut Florescu for organizing the 5th Annual Modeling High Frequency Data in Finance Conference in 2013. We are also grateful for valuable comments and suggestions from an anonymous referee. All errors and omissions are the sole responsibility of the authors.

Disclosure statement

No potential conflict of interest was reported by the authors.

Disclaimer

The first author and the third author were both working at J.P. Morgan & Co. when the first version of this paper was prepared. Any comments or statements made herein do not necessarily reflect those of Quest Partners LLC or J.P. Morgan & Co.

References

Aït-Sahalia, Y. and Yu, J., High frequency market microstructure noise estimates and liquidity measures. *Ann. Appl. Stat.*, 2009, **3**, 422–457.

Allen, F. and Karjalainen, R., Using genetic algorithms to find technical trading rules. *J. Financ. Econ.*, 1999, **51**, 245–271.

Almgren, R. and Chriss, N., Optimal execution of portfolio transactions. *J. Risk*, 2001, **3**, 5–40.

Almgren, R., Thum, C., Hauptmann, E. and Li, H., Direct estimation of equity market impact. *Risk*, 2005, **18**, 58–62.

Andersen, T., Bollerslev, T., Diebold, F. and Labys, P., Modeling and forecasting realized volatility. *Econometrica*, 2003, **71**, 579–625.

Avellaneda, M. and Stoikov, S., High-frequency trading in a limit order book. *Quant. Finance*, 2008, **8**, 217–224.

Barndorff-Nielsen, O.E., Hansen, P.R., Lunde, A. and Shephard, N., Realized kernels in practice: Trades and quotes. *Econ. J.*, 2009, **12**, C1–C32.

Bouchaud, J.P., Mézard, M., Potters, M., et al., Statistical properties of stock order books: Empirical results and models. *Quant. Finance*, 2002, **2**, 251–256.

Breiman, L., Bagging predictors. *Machine Learn.*, 1996, **24**, 123–140.

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

Brennan, M.J. and Copeland, T.E., Stock splits, stock prices, and transaction costs. *J. Financ. Econ.*, 1988, **22**, 83–101.

Bühlmann, P., Rütimann, P., van de Geer, S. and Zhang, C.H., Correlated variables in regression: Clustering and sparse estimation. *J. Stat. Plan. Inference*, 2013, **143**, 1835–1858.

Bühlmann, P. and Yu, B., Boosting with the L 2 loss: Regression and classification. *J. Am. Stat. Assoc.*, 2003, **98**, 324–339.

Chen, Z., Cahan, R. and Luo, Y., The evolution of market timing. Deutsche Bank Quantitative Strategy, 2013.

Cont, R., Kukanov, A. and Stoikov, S., The price impact of order book events. *J. Financ. Econ.*, 2013, **12**, 47–88.

Creamer, G., Model calibration and automated trading agent for Euro futures. *Quant. Finance*, 2012, **12**, 531–545.

Dempster, M. and Leemans, V., An automated FX trading system using adaptive reinforcement learning. *Expert Syst. Appl.*, 2006, **30**, 543–552.

Doumpos, M., Zopounidis, C. and Pardalos, P.M., *Financial Decision Making Using Computational Intelligence*, Vol. 70, 2012 (Springer: New York).

Fan, J. and Lv, J., Sure independence screening for ultrahigh dimensional feature space. *J. Roy. Stat. Soc.: Ser. B (Stat. Meth.)*, 2008, **70**, 849–911.

Fan, J. and Lv, J., A selective overview of variable selection in high dimensional feature space. *Stat. Sinica*, 2010, **20**, 101–148.

Fan, J., Song, R., et al., Sure independence screening in generalized linear models with NP-dimensionality. *Ann. Stat.*, 2010, **38**, 3567–3604.

Freund, Y. and Schapire, R., A decision-theoretic generalization of on-line learning and an application to boosting. In *Proceedings 2nd European Conference on Computational Learning Theory*, pp. 23–37, 1995 (Springer: Berlin).

Friedman, J.H., Greedy function approximation: A gradient boosting machine. *Ann. Stat.,* 2001, **29**, 1189–1232.

Friedman, J., Stochastic gradient boosting. *Comput. Stat. Data Anal.*, 2002, **38**, 367–378.

Friedman, J., Hastie, T. and Tibshirani, R., Special invited paper. Additive logistic regression: A statistical view of boosting. *Ann. Stat.*, 2000, 337–374.

Ganchev, K., Nevmyvaka, Y., Kearns, M. and Vaughan, J., Censored exploration and the dark pool problem. *Commun. ACM*, 2010, **53**, 99–107.

Harris, L., *Trading and Exchanges: Market Microstructure for Practitioners*, 2002 (Oxford University Press: New York).

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

Holland, J., *Adaptation in Artificial and Natural Systems*, 1975 (The University of Michigan Press: Ann Arbor, MI).

Iba, W. and Langley, P., Induction of one-level decision trees. *Proceedings of the Ninth International Workshop on Machine Learning*, pp. 233–240, 1992 (Morgan Kaufmann: San Mateo, CA).

In, F. and Kim, S., The hedge ratio and the empirical relationship between the stock and futures markets: A new approach using wavelet analysis. *J. Bus.*, 2006, **79**, 799–820.

Johnson, B., *Algorithmic Trading & DMA: An Introduction to Direct Access Trading Strategies*, 2010 (4Myeloma Press: London).

Kissell, R., The expanded implementation shortfall: Understanding transaction cost components. *J. Trading*, 2006, **1**, 6–16.

Kissell, R. and Glantz, M., *Optimal Trading Strategies: Quantitative Approaches for Managing Market Impact and Trading Risk*, 2003 (McGraw-Hill: New York).

Kordas, G., Smoothed binary regression quantiles. *J. Appl. Econ.*, 2006, **21**, 387–407.

Koulajian, N. and Czkwianianc, P., Black box trend following-lifting the veil. *AlphaQuest CTA Res. Ser.,* 2010, **1**, 1–16.

Koulajian, N., Czkwianianc, P. and Zhou, N., Return enhancing style drifts in futures/FX-based momentum portfolios. *CTA Intell.*, 2014, **21**, 24–25.

Kovac, P., *Flash Boys: Not So Fast: An Insider's Perspective on High-frequency Trading*, 2014 (Directissima Press: Engelska).

Kulkarni, A., Application of neural networks to stock market prediction. Technical Report, 1996.

LeBaron, B., An evolutionary bootstrap method for selecting dynamic trading strategies. In *Decision Technologies for Computational Finance. Proceedings of the Fifth International Conference Computational Finance*, Vol. 2, pp. 141–160, 1998 (Kluwer: Dordrecht).

Lewis, M., *Flash Boys*, 2014 (WW Norton & Company: New York).

Lo, A.W., MacKinlay, A.C. and Zhang, J., Econometric models of limit-order executions. *J. Financ. Econ.*, 2002, **65**, 31–71.

Lo, A., Mamaysky, H. and Wang, J., Foundations of technical analysis: Computational algorithms, statistical inference, and empirical implementation. *J. Finance*, 2002, **55**, 1705–1770.

Marzi, H. and Turnbull, M., Use of neural networks in forecasting financial market. In *Proceedings of the IEEE International Conference on Granular Computing*, Fremont, pp. 516–521, 2007.

Moody, J. and Saffell, M., Learning to trade via direct reinforcement. *IEEE Trans. Neural Netw.*, 2001, **12**, 875–889.

Nehren, D., Fellah, D., Ruiz-Mata, J. and Qin, Y., Dynamic density estimation of market microstructure variables via auxiliary particle filtering. *J. Trading*, 2012, **7**, 55–64.

O'Hara, M., *Market Microstructure Theory*. Vol. 108, 1995 (Blackwell: Cambridge, MA).

Pindyck, R.S. and Rubinfeld, D.L., *Econometric Models and Economic Forecasts*. Vol. 2, 1981 (McGraw-Hill: New York).

Samuelson, P.A., Asset allocation could be dangerous to your health. *J. Portfolio Manage.*, 1990, **16**, 5–8.

Satchell, S. and Timmermann, A., An assessment of the economic value of non-linear foreign exchange rate forecasts. *J. Forecasting*, 1995, **14**, 477–497.

Sharpe, W. F., The Sharpe ratio. *J. Portfolio Manage.,* 1994, **21**(1), 49–58.

Tay, F. and Cao, L., Application of support vector machines in financial time series forecasting. *Omega*, 2001, **29**, 309–317.

Therneau, T.M. and Atkinson, E.J., An introduction to recursive partitioning using the RPART routines. Technical Report, 61, Mayo Clinic, Rochester, 1997.

Tibshirani, R., Regression shrinkage and selection via the lasso. *J. Roy. Stat. Soc. Ser. B (Methodol.)*, 1996, **58**(1), 267–288.

Todorov, V., Estimation of continuous-time stochastic volatility models with jumps using high-frequency data. *J. Econ.*, 2009, **148**, 131–148.

Wahal, S. and Yavuz, M.D., Style investing, comovement and return predictability. *J. Financ. Econ.*, 2013, **107**, 136–154.

White, H., Learning in artificial neural networks: A statistical perspective. *Neural Comput*., 1989, **1**, 425–464.

Zhang, L., Mykland, P. and Aït-Sahalia, Y., A tale of two time scales: Determining integrated volatility with noisy high-frequency data. *J. Am. Stat. Assoc*., 2005, **100**, 1394–1411.

Zhang, D. and Zhou, L., Discovering golden nuggets: Data mining in financial application. *IEEE Trans. Syst. Man Cybern. Part C: Appl. Rev.,* 2004, **34**, 513–522.

Zheng, S., QBoost: Predicting quantiles with boosting for regression and binary classification. *Expert Syst. Appl*., 2012, **39**, 1687–1697.

Zhou, N., Volatility and jumps in high frequency financial data: Estimation and testing. PhD Thesis, University of Pittsburgh, 2011.