论文编号_____

对外经济贸易大学

University of International Business and Economics

毕业论文

RGBoost: A revised gradient boost machine

学号 _____

姓名 _____

学院 _____

专业 _____

导师 _____

时间 _____ 年 ___ 月 ___ 日

No._____

对外经济贸易大学

University of International Business and Economics

Graduation Thesis

Student ID No. _____

Student Name _____

Department/School _____

Major Field _____

Advisor _____

Date _____

A revised gradient boosting machine with its application in investment.*

Nanyi Zhang†

Abstract

This paper introduces a new gradient boosting algorithm (RGBoost) that modifies the negative gradient in each iteration. We begin by providing a strict definition of the gradient with the assistance of the Riesz representation theorem. We then demonstrate that the gradient vector in the traditional gradient boosting algorithm is biased when the hypothesis is a Reproducing Kernel Hilbert Space (RKHS). Intuitively, the modified gradient vector can more accurately approximate the gradient function. Finally, we illustrate that the revised model significantly outperforms the previous model in simulated data. The model is later applied to quantitative finance.

Keywords: Gradient boosting, Reproducing Kernel Hilbert Space(RKHS), Function Approximation, investment

*Nanyi Zhang is at School of Insurance and Economics, University of International Business and Economics, Beijing 100029, China.

†Corresponding author. Email: nymath@163.com

1 Introduction

Gradient Boosting Machine (GBM) is a machine learning algorithm commonly used for building predictive models in various tasks, including regression and classification. GBM combines weak learners sequentially to create a strong learner, which explains its strong performance across different datasets. The entire process can be viewed as a variational method in which the functional gradient algorithm obtains the goal function. As a result, GBM is widely regarded as one of the best models in statistical learning.

Freund first proposed boosting weak learners (Adaboost), and Schapire [3]. Later, Friedman [4] introduced an algorithm based on gradient boosting. Since then, the concept of gradient boosting has been further developed and expanded upon. In 2001, Breiman [1] introduced a variant of gradient boosting called Random Forest. This model uses a combination of decision trees and random subspace sampling to improve the algorithm's performance. In 2006, Hastie, Tibshirani, and Friedman introduced Gradient Boosted Regression Trees (GBRT), which combine decision trees and gradient boosting to produce a powerful and flexible machine learning algorithm for regression problems. In 2015, Chen [2] introduced a widespread implementation of the Gradient Boosting Machine. XGBoost is based on decision trees and uses a novel regularization technique to prevent overfitting. The algorithm includes several advanced features, such as parallel processing, distributed computing, and early stopping. XGBoost has been widely adopted in industry and research and has won several machine learning competitions. LightGBM is another implementation of the Gradient Boosting Machine introduced by Microsoft in 2017 [6]. LightGBM uses a novel histogram-based approach to speed up the computation of gradients and includes several advanced features, such as categorical feature handling and GPU acceleration. LightGBM has been shown to outperform other popular gradient-boosting algorithms in speed and accuracy. Overall, these related works have helped advance the Gradient Boosting field and make it a popular and powerful machine learning technique.

Due to their simplicity, the traditional boosting algorithm commonly uses decision trees as base learners. However, there is no restriction on the type of base learners that can be used. The gradient boosting algorithm can be viewed as an approximated functional gradient descent process. In this process, a base learner is chosen to approximate the gradient function queried at $(x_1, \cdots, x_n)$ in L^2 norm. Many studies have focused on choosing different types of base learners, such as Generalized Linear Models (GLMs), Kernel functions, splines, and others. Some studies have also attempted to combine different types of base learners in the boosting framework. For example, Sigrist [10], and Hoffmann [5] obtained more negligible bias and error by combining different base learners.

However, there is limited research on modifying the negative gradient vector, a crucial element in each iteration. We argue that the negative gradient vector used by Friedman [4] is biased and propose a better alternative by assuming that the hypothesis space is a Reproducing Kernel Hilbert Space (RKHS). To validate the effectiveness of our model, we begin by introducing the concept of Frechet derivatives, which

are essential in the study of functional analysis. We then delve into the differential of functionals in function space, which plays a critical role in mathematical analysis. By providing a clear definition of the gradient using the Riesz representation theorem, we give a rigorous framework for understanding the concept of the gradient in functional spaces. Finally, by utilizing the chain rule of Frechet derivatives, we calculate the gradient of the empirical loss function and then embed it into boosting algorithm.

This paper is organized as follows. Section 2 introduces some concepts in functional analysis and derives an alternative negative gradient vector. We then discuss the regression range and determine which function our model will approximate. In Section 3, we design the gradient boosting machine and introduce my rngboost package in Python. In Section 4, we verify the accuracy of our model using the sinc(x) dataset. Finally, Section 5 provides a summary of this paper.

2 The model

In this section, we cover some fundamental concepts in functional analysis and provide a more precise definition of the gradient using the Riesz Representation Theorem. Next, we derive the gradient of the empirical loss functional under the assumption that the hypothesis space is a Reproducing Kernel Hilbert Space (RKHS). Our analysis shows that the gradient estimation step in Friedman's boosting algorithm is biased. To address this issue, we suggest using the vector created by assessing the gradient function at sample points as the negative gradient vector in our boosting algorithm.

To the best of our knowledge, the goal of gradient boosting machine is to find an element in the hypothesis space which minimize the empirical loss function, and this can done through gradient descent algorithm on a functional space. However, gradient descend on function space requires more sophosicated mathematical tools, and thus we choose to start this section by listing some preminiaries about functional analysis.

2.1 Preliminaries

To provide a more detailed description of our model, we first introduce some notions in functional analysis. Let $\mathcal{H}$ be a vector space. We define the evaluation functional E_x as the function that maps a functional in $\mathcal{H}$ into its function value at x. Specifically, let

$$E_x : \mathcal{H} \to \mathbb{R}, \quad f \mapsto f(x). \tag{1}$$

Note that $E_x(f+g) = (f+g)(x) = E_x(f) + E_x(g)$, which means that E_x is a linear functional on $\mathcal{H}$.

In Equation 1, we use the notation $\mathcal{H}$ to represent a function space because $\mathcal{H}$ often stands for the hypothesis space or the Hilbert space. A Hilbert space is a complete inner product space. To the best of our knowledge, the norm measures the distance between vectors, while the inner product can describe the correlation between two elements in a space. Since we need to discuss some analytic properties such as continuity, differentiability, or general limits. Thus, we need an complete space, meaning every Cauchy sequence has a limit. This is where a Hilbert space comes in. It combines the abovementioned requirements and is therefore widely used in this context.

Since

$$D_f : \mathrm{int}_{\mathcal{X}} \to B(\mathcal{X}, \mathcal{Y}), \quad x \mapsto D_{f,x}.$$

To get a insight into D_f, we take multivariate function as example. In calculus, we have that

$$D_{f,x_0} = \sum_{i=1}^{n} \partial f_i(x_o) \mathrm{d}x_i.$$

Where

$$\mathrm{d}x_i : \mathbb{R}^n \to \mathbb{R}, \quad h \mapsto h_i.$$

Note that $\mathrm{d}x_i$ is the dual bases on $\mathbb{R}^n$ with respect to standard normal basis, since

$$\mathrm{d}x_i(e_j) = \delta_{ij}.$$

Next we will introduce a theorem which was widely used for understanding the structure of function spaces. This theorem has many applications in mathematics, physics and even economics.

Theorem 1 (Riesz Representation Theorem)**.**
Suppose φ is a bounded linear functional on a Hilbert space V. Then there exists a unique $h \in V$ such that

$$\varphi(f) = \langle f, h \rangle, \quad f \in \mathcal{H}$$

The Riesz representation theorem is a fundamental result in functional analysis that establishes a one-to-one correspondence between linear functionals on a Hilbert space and elements of the space itself. In particular, it states that for any bounded linear functional f on a Hilbert space $\mathcal{H}$, there exists a unique element h in $\mathcal{H}$ such that $\varphi(f) = \langle f, h \rangle$ for all f in $\mathcal{H}$. Here, $\langle \cdot, \cdot \rangle$ denotes the inner product on $\mathcal{H}$.

In a complete normed vector space. The continuity is equal boundedness of a linear functional. Since the evaluation functional mentioned is quite important in our model, we need a better space to discuss it. Note that not all the evaluation fucntional of a Hilbert space is bounded. To meet the requirement of Riesz representation theorem, we define the Reproducing Kernel Hilbert space as a space that every evaluation function is bounded(despite not bounded consistently).

Defintion 2 (RKHS)**.**
Let X be a set and $\mathcal{H}$ a Hilbert space with $\mathcal{H} \subset \mathbb{R}^X$. If the evaluation functional E_x over $\mathcal{H}$ is bounded. (or equivalently, continuous), then we say $\mathcal{H}$ is a Reproducing Kernel Hilbert Space.

Since the evaluation functional on $\mathcal{H}$ is a bounded, the Riesz Representation Theorem suggests that there exists unique vector $K_x \in \mathcal{H}$ which satisfites the following

$$f(x) = E_x(f) = \langle f, K_x \rangle_{\mathcal{H}}.$$

Also

$$K_x(y) = E_y(K_x) = \langle K_x, K_y \rangle_{\mathcal{H}}$$

This allows us to define the reproducing kernel of $\mathcal{H}$ as a function

$$K : X \times X \to \mathbb{R}, \quad (x, y) \mapsto K(x, y) = \langle K_x, K_y \rangle_{\mathcal{H}}.$$

However, in practical applications, it can be challenging to confirm whether the valuation functional in a Hilbert space is bounded. Even if it is bounded, the Riesz representation theorem only informs us of the existence of a kernel, but does not give the analytic expression of kernel. As a result, we commonly rely on a symmetric and positive definite function to construct a Hilbert space, which is known as the Moore-Aronszajn theorem.

Theorem 3 (Moore-Aronszajn)**.**
Suppose K is a symmetric positive definitive function on $X \times X$. Then there is a unique Hilbert Space of functions on X for which K is a reproducing Kernel.

Next, we will give a better definition of gradient. First recall the definition of gradient in calculus, it is a vector that points in the direction of the steepest increase of the function at a particular point and it can calculated by taking the partial derivative with respect to each coordinate. In fact, the inner product of a gradient and the incresement can be viewed as the linear approximation of the increment of function value. This idea can be directly extended to a function space. To be specific, suppose $\mathcal{H}$ is a normed vecotor space and L is a function defined at $x_0 \in \mathcal{H}$. We know that L is Frechet differentiable on $\mathcal{H}$ if and only if

$$L(x_0 + h) = L(x_0) + D_{L,x_0}(h) + e_{x_0}(h), \quad \forall h \in \mathcal{H}. \tag{2}$$

Where D_{L,x_0} is a linear functional on $\mathcal{H}$. Also note that D_L can be viewed as the linear bounded map from $\mathcal{H}$ to $\mathcal{H}^*$ or $D_L \in B(\mathcal{H}, \mathcal{H}^*)$[1]. Also, the error part must moves faster to zero that h, that is

$$\lim_{h \to 0} \frac{e_{x_0}(h)}{\|h\|} = 0.$$

Note that the zero in right side of the above equation is the additive identity$(0 : \mathcal{H} \to \mathbb{R}, \quad f \mapsto 0)$ in $\mathcal{H}$ rather than the real number 0.

The Riesz Representation Theorem ensures a unique vector $g(x_0, L) \in \mathcal{H}$ exists and has the following property:

$$D_{L,x_0}(h) = \langle h, g(x_0, L) \rangle.$$

This remains us to define the gradient of function as the unique element in the Hypothesis space that represents the Frechet derivative of the functional.

[1]The $\mathcal{H}^*$ notation here stands for the dual space of $\mathcal{H}$

Suppose f is multivariate function, according to our definition, the gradient of f at x_0 is the Jacobian matrix of f as x_0.

$$\nabla f(x_0) = \mathbf{J}_f^{x_0} = (\partial_1 f(x_0), \cdots, \partial_n f(x_0)).$$

Next we introduce the gradient of evaluation functional

Theorem 4 (Gradient of the evaluation functional)**.**
Suppose $\mathcal{H}$ is a RKHS. Then the gradient of the evalutaional function at f when given x is

$$\nabla E_x(f) = K_x \in \mathcal{H}.$$

Proof.

$$\begin{aligned} E_x(f+h) &= E_x(f) + E_x(h) \\ &= f(x) + \langle K_x, h\rangle + 0. \end{aligned}$$

The Riesz Representation theorem admits the uniqueness of K_x. □

Suppose $\|h\| = 1$,

$$\begin{aligned} L(f_0 + h) - L(f_0) &\sim \langle \nabla L(f_0), h\rangle \\ &\geq -\|\nabla L(f_0)\|. \end{aligned}$$

Thus the gradient provides a direct way to find the local minimum of a functional. Indeed, the quation above suggests that L moves the fastest when walking along the diection of gradient. This gives us an idea to minimize a functional. We state this at the Algorithm 1.

Algorithm 1: Function Gradient descend algorithm

Input:

- the functinal $L : V \to \mathbb{R}$.
- number of iterations $M \geq 1$
- leanring rate $\eta \in (0, 1]$

Initialize f_0

while $m \in \{1, \cdots, M\}$ **do**

Calculate the gradient:

$$\nabla L(f_{m-1})$$

Find the best gradient descent step-size:

$$\rho_m = \arg\min_{\rho \in \mathbb{R}} L(f_{m-1} - \rho \nabla L(f_{m-1})))$$

Update:

$$f_m \leftarrow f_{m-1} - \eta \rho_m \nabla L(f_{m-1})$$

end

Output: f_M

Algorithm 1 is powerful for solving some variational problem. At each iteration, we first calculate the gradient of loss functional and then find the best gradeint descent step-size using linear search. But calculating the gradient of a functional raises a problem, since the Riesz representation theorem can only ensures the existence of gradient. One solution is to find an element in the hypothesis space which is close to the gradient as close as possible.

Since the gradient function can only be observed as finite sample points, we move our concentration to the empirical case. Suppose we are aiming to find the local minimul of $\mathcal{L}$ where

$$\mathcal{L} : \mathcal{H} \to \mathbb{R}, \quad f \mapsto \frac{1}{n} \sum_{i=1}^{n} L(y_i, f(x_i)). \tag{3}$$

Before staring the gradient descent algorithm, we find that $\mathcal{L}$ yield another functional $\mathcal{L}'$ which is defined by

$$\mathcal{L}' : \mathbb{R}^n \to \mathbb{R}, \quad h \mapsto \frac{1}{n} \sum_{i=1}^{n} L(y_i, h_i), \tag{4}$$

where h_i stands for the i-th coordinate of vector h. Next, we will find out the relationship between $\mathcal{L}$

and $\mathcal{L}'$. To simplify the notation, suppose $\hat{y} = (f(x_1), \cdots, f(x_n))^T$. By definition, we have $\mathcal{L}(f) = \mathcal{L}'(\hat{y})$. Now, we turn to calculate the gradient of $\mathcal{L}$ and $\mathcal{L}'$.

Based on the knowledge of calculus, the gradient of a multivariate function consists of partial derivatives

$$\nabla \mathcal{L}'(\hat{y}) = \begin{bmatrix} \frac{1}{n}\partial_2 L(y_1, f(x_1)) \\ \frac{1}{n}\partial_2 L(y_2, f(x_2)) \\ \vdots \\ \frac{1}{n}\partial_2 L(y_n, f(x_n)) \end{bmatrix} \tag{5}$$

where $\partial_2 L$ stands for the partial deritivative of the second component. $\nabla \mathcal{L}'(\hat{y})$ is viewed as the negative gradient vector in friedman's boosting algorithm, which is presented in algorithm 2.

Algorithm 2: Friedman's Gradient Boosting algorithm

Input:

- the functinal $L : V \to \mathbb{R}$.
- number of iterations $M \geq 1$
- leanring rate $\eta \in (0, 1]$

Initialize $F_0 = \arg\min_\rho \sum_{i=1}^n L(y_i, \rho)$

while $m \in \{1, \cdots, M\}$ **do**

Calculate the negative gradient:

$$g_{m,i} = -\left[\frac{\partial L(y_i, F(\mathbf{x}_i))}{\partial F(\mathbf{x}_i)}\right]_{F(\mathbf{x})=F_{m-1}(\mathbf{x})}, \ i = 1, \cdots, n$$

Fit a base estimator:

$$\mathbf{a}_m = \arg\min_{\mathbf{a},\beta} \sum_{i=1}^n \left[g_{m,i} - \beta h(\mathbf{x}_i; \mathbf{a})\right]^2$$

Find the best gradient descent step-size:

$$\rho_m = \arg\min_\rho \sum_{i=1}^n L(y_i, F_{m-1}(\mathbf{x}_i) + \rho h(\mathbf{x}_i; \mathbf{a}_m))$$

Update:

$$F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \rho_m h(\mathbf{x}; \mathbf{a}_m)$$

end

Output: F_M

Here we choose to list the algorithm without changing notations in order to make contract with our

model. You should note that the notation $\frac{\partial L(y_i,F(\mathbf{x}_i))}{\partial F(\mathbf{x}_i)}$ equals $\partial_2 L(y_i, F(x_i))$. In step 4, this algorithm select a base learner to fit the gradient vector in L^2 norm at sample points and then roled as the gradient function in algorithm 1.

2.2 The revised gradient boosting machine

This subsection presents our revised model. It's important to keep in mind that the negative gradient vector in algorithm 2 represents the gradient of the derived empirical loss functional $\mathcal{L}'$, not the gradient of $\mathcal{L}$. Although it may seem impossible to calculate the gradient of the empirical loss functional $\mathcal{L}$ at f, theorem 4 provides an explicit expression of it.

Theorem 5 (The gradient of empirical loss function)**.**
Suppose $\mathcal{H}$ is a RKHS with kernel K defined on it. Then the gradient of the empirical loss function $\mathcal{L}$ can be expressed as

$$\nabla\mathcal{L}(f) = \frac{1}{n}\sum_{i=1}^{n} \partial_2 L(y_i, f(x_i))K_{x_i}.$$

Proof. The proof can be found in the Appendix A. □

Here $\nabla\mathcal{L}(f)$ is a functional on $\mathcal{X}$ and it can only be observed at finite sample points. For example

$$\nabla\mathcal{L}(f)(x_1) = \frac{1}{n}\sum_{i=1}^{n} \partial_2 L(y_i, f(x_i))K(x_i, x_1).$$

And we have the following relationship,

$$\begin{aligned}
\begin{bmatrix} \nabla\mathcal{L}(f)(x_1) \\ \vdots \\ \nabla\mathcal{L}(f)(x_n) \end{bmatrix} &= \begin{bmatrix} \frac{1}{n}\partial_2 L(y_1, f(x_1)), & \cdots, & \frac{1}{n}\partial_2 L(y_n, f(x_n)) \end{bmatrix} \begin{bmatrix} K(x_1,x_1) & \cdots & K(x_1,x_n) \\ \vdots & & \vdots \\ K(x_n,x_1) & \cdots & K(x_n,x_n) \end{bmatrix} \\
&= (\nabla\mathcal{L}'(\hat{y}))^T K.
\end{aligned}$$

We now claim that $\nabla\mathcal{L}'(\hat{y})$ is not always equal to the gradient function $\nabla\mathcal{L}(f)$ which evaluated at sample

points. In fact, they are the same if and only if

$$K(x_i, x_j) = \chi_{\{x_i\}}(x_j) = \delta_{ij},$$

where χ is the characteristic functional of set. Based on the result of theorem 5, we revise step 3 in Friedman's gradient boosting algorithm in algorithm 3.

Algorithm 3: The Revised Gradient boosting algorithm

Data: $\mathcal{D} = \{(y_1, x_1), \cdots, (y_n, x_n)\}$

Input:

- the hypothesis $\mathcal{H}$ and the reproducing kernel K.
- the empirical loss function $\mathcal{L} : \mathcal{H} \to \mathbb{R}$
- number of iterations $M \geq 1$
- leanring rate $\eta \in (0, 1]$

1 Initialize f_0, $\quad f_0(x) = \arg\min_{c \in \mathbb{R}} \frac{1}{n} \sum_{i=1}^{n} L(y_i, c)$ for all $x \in \mathbb{R}^p$.

2 **while** $m \in \{1, \cdots, M\}$ **do**

3 Calculate the negative gradient vector:

$$g_{m,j} = -\frac{1}{n} \sum_{i=1}^{n} \partial_2 L(y_i, f_{m-1}(x_i)) K(x_i, x_j), \quad j = 1, 2, \cdots, n.$$

4 Fit the gradient vector:

$$h_m = \arg\min_{h \in \mathcal{H}, \beta} \sum_{i=1}^{n} (g_{m,i} - \beta h_m(x_i))^2$$

5 Find the best gradient descent step-size:

$$\rho_m = \arg\min_{\rho \in \mathbb{R}} \mathcal{L}(f_{m-1} + \rho h_m)$$

6 Update:

$$f_m \leftarrow f_{m-1} + \eta \rho_m h_m$$

7 **end**

Output: $f_M = f_0 + \sum_{i=1}^{M} \eta \rho_m h_m$

Algorithm 3 is a revised verison of gradient boosting machine. At step 1, we search a constant function minimize the empirical loss function. If we apply the square loss function, this constant must be the sample mean of y. At step 3, we substitude the gradient based on Theorem 5.

2.3 Statistical theory of regression

This subsection illustrates the statistical theory of regression. Just as we have discussed above, the gradient boosting machine can be used to approximate the function which minimize the empirical loss function. So how to choose a loss function and which function will be approximated raise a question and in the subsection, we will summarize the statistical theory of regression and shows that the GBM can be used to approximate the conditional expectation function $m(x) = E(Y|X = x)$ when choosing the L^2 loss function.

To make clear these concepts, we will discuss it in the framework of measure theory. We start with a measurable space, which is a turple of set and a sigma algebra. On a measurable space, we can define a measure, which is a set function that satisfies countable additivity. The integration with respect to measure has better properties than the Riemann integration we learned in calculus. Riemann integration requires partitioning of x, which is intuitive but loses its standard when the function is defined in an abstract space. With the help of integration of measures, we can have a deeper understanding of random variables.

In fact, the integration of a random variable with respect to a probability measure has the following relationship with Lebesgue-Stieltjes integration.

Theorem 6 (Measure transformations)**.**
Suppose $(\Omega, \mathcal{F})$ and $(E, \mathcal{E})$ are two measurable spaces, $f : \Omega \to E$, $g : E \to \mathbb{R}$. Then we have the following

$$\int_{\Omega} g \circ f \mathrm{d}P = \int_{E} g \circ I \mathrm{d}\mu_f, \tag{6}$$

where $I : E \to E, x \mapsto x$.

Proof. The proof can be found in the Appendix. □

The goal of regression is to find the conditional expectation function. Conditional expectation is a fundamental concept in probability theory and statistics. It describes the expected value of a random variable given certain information about another random variable. The definition and properties of conditional expecation can be found in Appendix B. The next theorem states that the conditional expectation with respect to a sigma-algebra generate by a random map can be expressed as a borel-measurable function composition of this random map.

Theorem 7. Suppose $X : \Omega \to \mathbb{R}^p$, X is random vevtor, then there exists a borel-measurable function f $\mathbb{R}^p \to \mathbb{R}$, such that

$$E(Y|\sigma(X)) = f \circ X$$

Proof. The proof can be found in Appendix A. □

The inverse of Theorem 7 is also true and it provide an efficient way to calculate the conditional expectation function provided that the sigma algebra is generated by a random map.

Theorem 8 (The connection between different types of contional expectations)**.**
Suppose $m : \mathbb{R}^p \to \mathbb{R}, \quad x \mapsto E(Y|X = x)$ is a borel-measurable function, then

$$m \circ X = E(Y|\sigma(X)).$$

Proof. The proof can be found in Appendix A. □

To get a deeper understanding of conditional expectation and thus regression, we slightly release the restritions on Y. To be specific, we assume that $Y \in \mathcal{L}^2(\Omega, \mathcal{F}, P)$ and $\mathcal{G}$ is a sub-sigma-algebra of $\mathcal{F}$. Consider the space $\mathcal{L}^2(\Omega, \mathcal{G}, P_{\mathcal{G}})$, which is a closed subspace of $\mathcal{L}^2(\Omega, \mathcal{F}, P)$. Then the Hilbert project theorem suggests that there exists a unique element M in $\mathcal{L}^2(\Omega, \mathcal{G}, P_{\mathcal{G}})$ such that

$$\|Y - M\|_2 \leq \|Y - f\|_2, \quad f \in \mathcal{L}^2(\Omega, \mathcal{G}, P_{\mathcal{G}}).$$

And we can show that M statisfies all the conditions to be a conditonal expecation.

Thus, regression can be viewed as a projection onto a subspace. Suppose Y is a random variable on a probability space $(\Omega, \mathcal{F}, P)$, X is a random map from Ω to E(Remeber that E often refers to $\mathbb{R}^p$).

$$\begin{aligned} Y &= E(Y|X) + Y - E(Y|X) \\ &= m \circ X + \varepsilon \end{aligned}$$

We know that the conditional expectation enjoys the smallest distance to Y, from this point of view, we need to approximate the conditional expecation using sample data. To be specific, we need to minimize the empirical loss function,

$$\arg\min_{f \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} (y_i - f(x_i))^2.$$

There are many ways to maximize the the empirical loss function. The easiest way is parameterize. For example, we can fix the form of the hypothesis as

$$\mathcal{H} = \left\{ f : \mathbb{R}^p \to \mathbb{R}, \quad x \mapsto c + \beta^T x | where\ \beta \in \mathbb{R}^p, c \in \mathbb{R} \right\}.$$

In this case, the problem is converted to

$$\arg\min_{\beta \in \mathbb{R}^{p+1}} \frac{1}{n} \sum_{i=1}^{n} (y_i - \beta^T \tilde{x}_i)^2.$$

This process is called linear regression. In fact

$$\hat{\beta} = (X^T X)^{-1} X^T Y.$$

The hypothesis can also be a RKHS. Assume that $K : \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R}$ is a positive definite kernel function. Then there exists reproducing kernel Hilbert space(RKHS) $\mathcal{H}$ with an innner product such that K_x belongs to $\mathcal{H}$ for all $x \in \mathbb{R}^p$. The representer theorem [9] then states that there is a uniuqe minimizer of the form

$$f = \sum_{i=1}^{n} \alpha_i K_{x_i}$$

3 GBM Design

In this section, we list the alternative loss functions and base learners, and describe their Applicable situations. Later we provide some improvement approaches to GBM, which contains combine different type of base learners, regularzation, subsample and others. For more information about the GMB design, we refer to NateKin [7].

To design a GBM for a specific task, it is necessary to determine the available options for the empirical loss function $\mathcal{L}$ and the hypothesis $\mathcal{H}$ (which is the set of base learners), as these choices will greatly impact the GBM's performance.(just as we discussed in last section, the conditional expectation function minimize the L^2 loss.)

3.1 Loss functions

The loss function is a critical element of the gradient boosting machine, as it quantifies the discrepancy between the anticipated and actual values of the target variable. The selection of loss functions (functionals) depends on the objective of our model. For instance, if the goal is to obtain the conditional expectation of the response given X, the square loss function (L^2 loss) is preferable since the conditional expectation can be considered as the projection of the response, thus attaining the smallest L^2 distance to the response. To use a loss function in practice, one must specify both the loss and the corresponding negative gradient. (In our Python module, the loss function is defined as an abstract class with attributes "loss" and "negative gradient".) Other loss functions are provided in Schmid et al., 2011 [8].

The types of loss functions can be roughly divided according to the type of response as follows:

1. **Continuous response:**
    - L^2 loss (Conditional Expectation)
    - L^1 loss (Conditional Median)
    - Quantile loss (Conditional Quantile)
    - ZeroOne loss (Conditional Mode).
2. **Categorical response:**
    - Exponential loss
    - Cross-Entropy loss
3. **Other response:**
    - MLE loss

- Custom loss

The custom loss function is a commonly used concept in quantitative finance and investment analysis, particularly in stock selection. At each rebalance date, we have a multitude of factors, including historical prices, financial ratios, and technical indicators, which are denoted by $F_t \in \mathbb{R}^{m\times N}$, where m is the number of factors and N is the number of stocks. Our goal is to combine these factors into a vector $\alpha_t \in \mathbb{R}^N$ and then use a given strategy to obtain indicators like Sharpe ratio or excess alpha. This process can be viewed as a customer loss function. Specifically, we can represent the result at time t as

$$\begin{aligned} \text{result}_t &= \text{Strategy}_t \circ f \circ F_t \\ &= \text{Strategy}_t \circ \alpha_t. \end{aligned}$$

In this situation, the custom loss functional can be expressed as

$$\mathcal{L} : \mathcal{H} \to \mathbb{R}, \quad f \mapsto \frac{1}{T}\sum_{i=1}^{T} \text{Strategy}_t \circ f \circ F_t.$$

Theorem 9. (L^2 loss, general version)
Suppose $(\Omega, \mathcal{F}, P)$ is a probability space, $Y : \Omega \to \mathbb{R}$ and $X : \Omega \to \mathbb{R}^p$. The the solution of the following

$$\arg\min_{f\in\mathcal{H}} \|Y - f \circ X\|_2^2 = \arg\min_{f\in\mathcal{H}} \int_\Omega (Y - f \circ X)^2 \mathrm{d}P$$

is the function $m : \mathbb{R}^p \to \mathbb{R}, \quad x \mapsto E(Y|X = x)$.

Note that the notation $X = x$ stands for the set $\{\omega : X(\omega) = x\}$ 定理26提供了一个比较漂亮的结果, 但这很难在实际中应用，我们先建立总体到样本的关系

$$\begin{aligned} Y \in \mathcal{L}^1(\Omega) \quad &\Longrightarrow \quad (y_1, y_2, \cdots, y_n)' \in \mathbb{R}^n. \\ f \circ X \in \mathcal{L}^1(\Omega) \quad &\Longrightarrow \quad (f(x_1), f(x_2), \cdots, f(x_n))' \in \mathbb{R}^n. \end{aligned}$$

上述对应关系给了我们一个近似的思路，即利用$\mathbb{R}^n$中的两个向量来近似刻画随机变量之间的距离

Theorem 10. (L^2 loss, empirical version)
Suppose $\mathcal{D} = \{(y_i, x_i) : i = 1, 2, \cdots, n\}$ is a sample from (Y, X). Then the conditional expectation of Y given X can be approximated by

$$\arg\min_{f\in\mathcal{H}} \frac{1}{2n}\sum_{i=1}^{n}(y_i - f(x_i))^2.$$

3.2 Base Learners

The base learn also plays importstant roles in model design since the GBM can be seen as the linear combination of base learners. In this section, we will introduce some base estimators which was widely used in practice.

Initially, we define the base-learners as a group of functions that map $\mathbb{R}^p$ to $\mathbb{R}$. It is not necessary for it to be a Hilbert space, nor even a vector space. Therefore, we refer to it as a collection of functions instead of a function space. In practice, there are three types of base-learners.

Type	**Base-Learner**
Linear models	Affine functions
Smooth models	P-splines
	Kernel functions
Decision trees	Decision tree

Table 1: Base Learners

The most commonly used base-learners is the decision trees. The decision tree can be viewed as the simple-measurable function and can be used to approximate borel-measurable functions in L^p norm and this makes its universality. The decision tree algorithm works by recursively splitting the data into subsets based on the values of one of the input features until a stopping criterion is met. The result is a tree-like structure where each internal node represents a decision based on the value of a feature, and each leaf node represents a final decision or prediction. One advantage of decision trees is that they are easy to interpret and visualize, making them a useful tool for understanding how a model makes decisions. They are also able to handle both categorical and numerical data, and can be used with both small and large datasets.

3.3 Combine different base learners

There are no restrictions on the selection of base learners when building a GBM model, meaning that several classes of base learners can be included at the same time. Many studies have focused on choosing different types of base learners, such as Generalized Linear Models (GLMs), Kernel functions, splines, and others. Some studies have also attempted to combine different types of base learners in the boosting framework. For example, Sigrist [10] and Hoffmann [5] obtained smaller bias and error by combining different types of base learners. We summarize these ideas in Algorithm 4.

Algorithm 4: The Combined Gradient boosting algorithm

Data: $\mathcal{D} = \{(y_1, x_1), \cdots, (y_n, x_n)\}$

Input:

- the hypothesis $\mathcal{H}_k$ and the reproducing kernel K.
- the empirical loss function $\mathcal{L} : \mathcal{H} \to \mathbb{R}$
- number of iterations $M \geq 1$
- leanring rate $\eta \in (0, 1]$

1 Initialize f_0, $\quad f_0(x) = \arg\min_{c \in \mathbb{R}} \frac{1}{n} \sum_{i=1}^{n} L(y_i, c)$ for all $x \in \mathbb{R}^p$.

2 **while** $m \in \{1, \cdots, M\}$ **do**

3 Calculate the negative gradient vector:

$$g_{m,j} = -\frac{1}{n} \sum_{i=1}^{n} \partial_2 L(y_i, f_{m-1}(x_i)) K(x_i, x_j), \quad j = 1, 2, \cdots, n.$$

4 Fit the gradient vector:

$$h_m^{(k)} = \arg\min_{h \in \mathcal{H}_k, \beta} \sum_{i=1}^{n} (g_{m,i} - \beta h(x_i))^2$$

5 Find the best base learner and step size:

$$\rho_m, \ k^* = \arg\min_{\rho, k} \mathcal{L}(f_{m-1} + \rho h_m^{(k)})$$

6 Update:

$$f_m \leftarrow f_{m-1} + \eta \rho_m h_m^{(k^*)}$$

7 **end**

Output: $f_M = f_0 + \sum_{i=1}^{M} \eta \rho_m h_m^{(k)}$

At each step, we use different types of learners to approximate the revised negative gradient vector. Then, in step 5, we choose the one that minimizes the empirical loss at this step. Finally, the output model is a linear combination of different types of base learners.

3.4 Regularization

3.4.1 subsample

Subsampling is a technique used in machine learning to improve the overall performance of models by reducing overfitting. This involves randomly selecting a subset of the training data for each iteration of the learning algorithm. By doing so, the variance of the model can be reduced, and it can prevent learning unnecessary noise in the data. The gradient descent algorithm based on subsample is commonly referred to as the stochastic descent algorithm (SGD). Empirical evidence has shown that the model trained by SGD

often yields better results, which is presented in algorithm 5. Suppose we have M batches such that:

$$\bigcup_{t=1}^{M} B_t = \{1, 2, \cdots, n\}.$$

Algorithm 5: The Revised Stochastic Gradient boosting algorithm

Data: $\mathcal{D} = \{(y_1, x_1), \cdots, (y_n, x_n)\}$

Input:

- the hypothesis $\mathcal{H}$ and the reproducing kernel K.
- the empirical loss functional $\mathcal{L} : \mathcal{H} \to \mathbb{R}$
- number of iterations $M \geq 1$
- leanring rate $\eta \in (0, 1]$
- stochastic batches: $B_t, t = 1, \cdots, M$.

Initialize f_0, $\quad f_0(x) = \arg\min_{c \in \mathbb{R}} \frac{1}{n} \sum_{i=1}^{n} L(y_i, c)$ for all $x \in \mathbb{R}^p$.

while $m \in \{1, \cdots, M\}$ **do**

Calculate the gradient vector:

$$g_{m,j} = -\frac{1}{|B_m|} \sum_{i \in B_m} \partial_2 L(y_i, f_{m-1}(x_i)) K(x_i, x_j), \quad j \in B_m.$$

Fit the gradient vector:

$$h_m = \arg\min_{h \in \mathcal{H}, \beta} \sum_{i \in B_m} (g_{m,i} - \beta h_m(x_i))^2$$

Find the best gradient descent step-size:

$$\rho_m = \arg\min_{\rho \in \mathbb{R}} \mathcal{L}(\mathbf{f}_{m-1} + \rho h_m)$$

Update:

$$f_m \leftarrow f_{m-1} + \eta \rho_m h_m$$

end

Output: $f_M = f_0 + \sum_{i=1}^{M} \eta \rho_m h_m$

The difference is that, we apply a stochastic batch B_m to calculate the negative gradient.

3.4.2 step size

The step-size, or learning rate, is a critical hyperparameter in the gradient descent algorithm. It determines the magnitude of the update to the model parameters during each iteration. A large step-size can cause the algorithm to diverge, while a small step-size can result in slow convergence. The learning rate can also be thought of as the weight of the base learners. If the learning rate is too high, the algorithm will focus too

much on the front part, which can reduce performance.

3.5 RGBoost

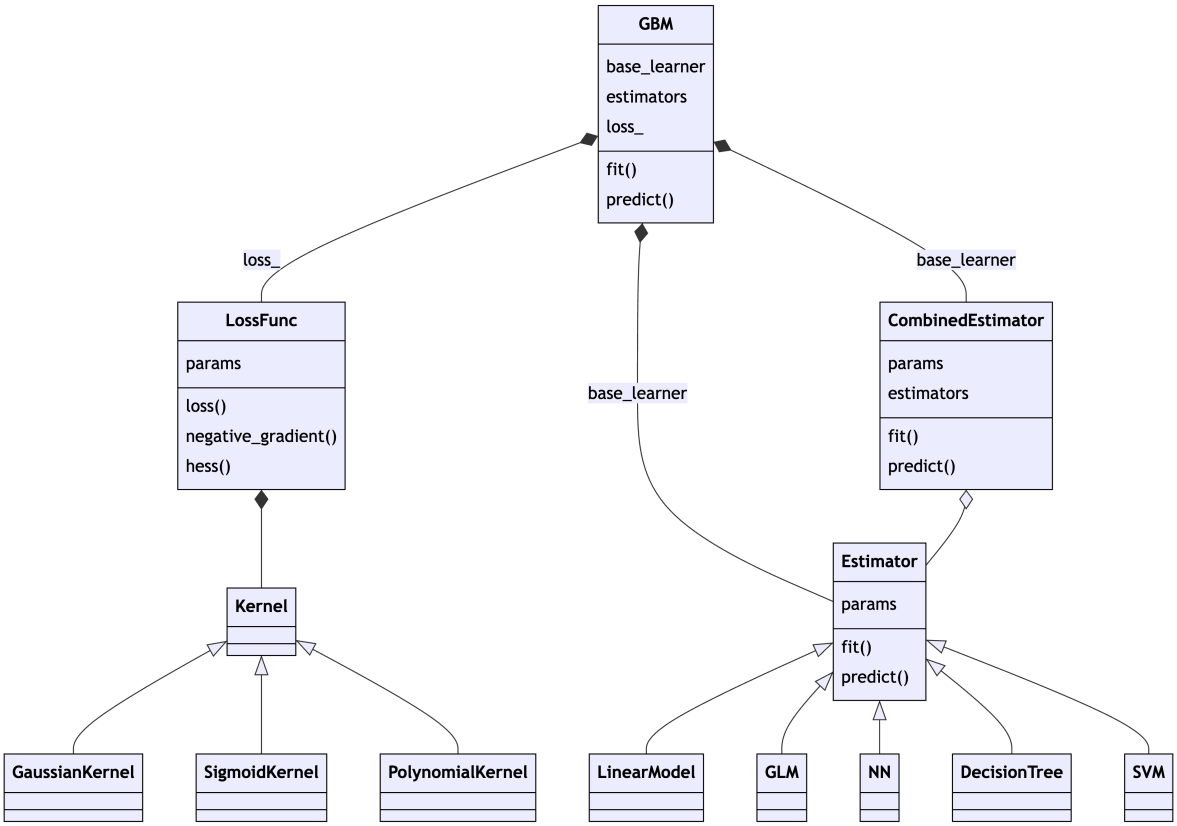

Figure 1: framework of rgboost

This section introduces the RGBoost[2]. module, which is based on the Python class GradientBoostingRegressor (Classifier) in sklearn.ensemble and can produce consistent results. While the base learner of GradientBoostingRegressor is limited to decision trees, RGBoost supports the combination of multiple base learners, including the combination function proposed by Sigrist[10]. Additionally, RGBoost incorporates the gradient correction scheme proposed in this paper.

One feature of our module is the versatility of estimator class. In fact, any model with attribute "fit" and "negative gradient" can be acted as an estimator. This property enable us to boost the models like neural network, SVM, GLM and even a boosted model. In addition, we define the collections of different types of Estimator classes as CombinedEstimator class. The CombinedEstimator class also has the "fit" and "negative gradient" and thus can be embeded into GBM class.

[2]rgboost is available on https://github.com/nymath

4 Application

In this section, we apply the gradient enhancer to function approximation. To be specific, we use synthetic dataset simulated from sinc(x)[3]. The simulated sinc(x) dataset is a common benchmark dataset used to evaluate the performance of regression models. It consists of 200 data points evenly spaced between -10 and 10, with a noisy sinc(x) function as the target variable. The sinc(x) function is defined as sinc$(x) = \frac{\sin(x)}{x}$. The noisy function is obtained by adding Gaussian noise with mean 0 and standard deviation 0.1 to the true function. The goal of the regression task is to learn the underlying function from the noisy data. The simulated sinc(x) dataset is often used to test the accuracy and generalization ability of different regression models, including gradient boosting models.

4.1 Function Approximation

In figure, 首先是对比在GBDT在测试集上的表现, 基于Revised negative gradient改进的模型, 在迭代的前期展现出更快的下降速度，同时能够达到更小的测试误差。对于Kernel-based learners，因为其更适合用于平滑问题

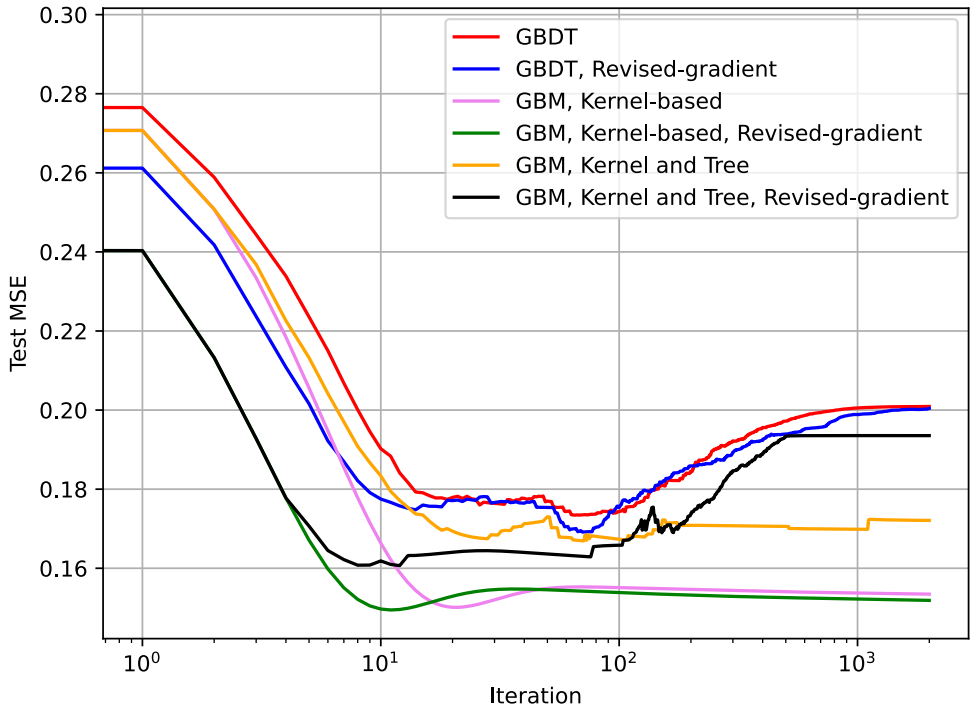

Figure 2: Test MSE versus the number of iteration for comparing GBM with/without revised gradient.

[3]See https://www.rdocumentation.org/packages/qrnn/versions/1.1.3/topics/sinc

We use gaussian kernels for illustration.

$$K(x,y) = \mathrm{e}^{-\frac{\|x-y\|^2}{\sigma^2}}, \quad \sigma > 0.$$

We found that

$$\lim_{\sigma \to 0^+} \mathrm{e}^{-\frac{\|x-y\|^2}{\sigma^2}} = \chi_{\{x\}}(y).$$

Which is consistent of Friedman's result. 图中展示了拟合的过程, 可以发现，在sinc(x)数据集中，基于梯度修正的GBM表现总是好于原始模型, 由于该数据集样本数较小, 所以在迭代次数超过100时将会出现过拟合的情形. 同时我们发现, 在平滑问题上, baselearn选择kernel functions的表现会明显好于GBDT, 这也可以从后续的图中看出来.

4.2 Event-Driven Backtesting

In this subsection, we will introduce some concepts in quantitative finance. For those who wish to learn more, we recommend visiting https://www.quantstart.com/. Established in 2012, QuantStart is an online resource for mathematical finance articles and tutorials on derivatives pricing, created to assist aspiring quants in obtaining a role in quantitative finance.

There are two primary frameworks in quantitative finance: vectorized backtester and event-driven backtester. While both are popular, event-driven systems provide several advantages over a vectorized approach:

- **Code Reuse**: An event-driven backtester can be used for historical backtesting and live trading with minimal component switching. In contrast, vectorized backtesters require all data to be available simultaneously for statistical analysis.
- **Lookahead Bias**: Event-driven backtesters eliminate lookahead bias by treating market data receipt as an "event" that must be acted upon. As a result, it is possible to "drip feed" an event-driven backtester with market data, replicating how an order management and portfolio system would behave.
- **Realism**: Event-driven backtesters allow for significant customization over how orders are executed and transaction costs are incurred. Essential market and limit orders, as well as market-on-open (MOO) and market-on-close (MOC) orders, can be easily handled by constructing a custom exchange handler.

Although event-driven systems have numerous benefits, they have two significant drawbacks compared to simpler vectorized systems. Firstly, they are more complicated to implement and test due to having more "moving parts," resulting in a greater risk of introducing bugs. Proper software testing methodology, such as test-driven development, can be employed to mitigate this. Secondly, they are slower to execute than

a vectorized system because optimal vectorized operations cannot be used when performing mathematical calculations.

Regenerate response

Pytrade is a event-driven trading system developed us. 借鉴Uqer，zipline，backtesting，qstrader等多个回测框架，我编写了一个基于事件驱动的量化回测系统，还原了优矿部分 功能，极大程度的避免了量化研究中的前向偏差问题。目前该框架支持策略开发，因子投研，参数优化等功能。在下单方面， 系统充分考虑市场微观结构，支持市价单，限价单，止损单等多种订单。在事件驱动方面，系统利用事件队列的结构，能够处 理新闻数据等另类信息，并同时生成交易信号。

4.3 Stock selection based on gradient boosting

In stock selection, we need to choose those stocks whose price will rise significantly in the future. 在监督学习中, 对数据进行标注至关重要, 我们按照不同的标注方式，将模型分为分类, 回归两类. 在分类问题中, 我们对股票未来一段时间内的收益率进行排序, 前百分之30标记为正, 后百分之30标记为负,

$$
\begin{aligned}
B_t^+ &= \left\{ i : CrossQuantile(\frac{C_{t+k}^{(i)} - C_t^{(i)}}{C_t^{(i)}}) > 0.7 \right\} \\
B_t^- &= \left\{ i : CrossQuantile(\frac{C_{t+k}^{(i)} - C_t^{(i)}}{C_t^{(i)}}) < 0.3 \right\}
\end{aligned}
$$

$$
y_t^{(i)} = \chi_{B_t^+}(i) - \chi_{B_t^-}(i), \quad i \in B_t^+ \cup B_t^-.
$$

而我们认为未来一段时间内收益率分位数为在0.3至0.7之间的数据为噪声, 故将其删除.

风险度量指标 Suppose r_t is the daily(hourly ...) return, then

- Mean and Volatility
$$
mean(r) = \frac{1}{T} \sum_{t=1}^{T} r_t,
$$
- Maxdrawdown:
- Sharpe Ratio:
- Alpha / Excess alpha:
- Beta:
- Turnover Rate:

而在回归问题中, 我们就有了更多数据标注的方案.

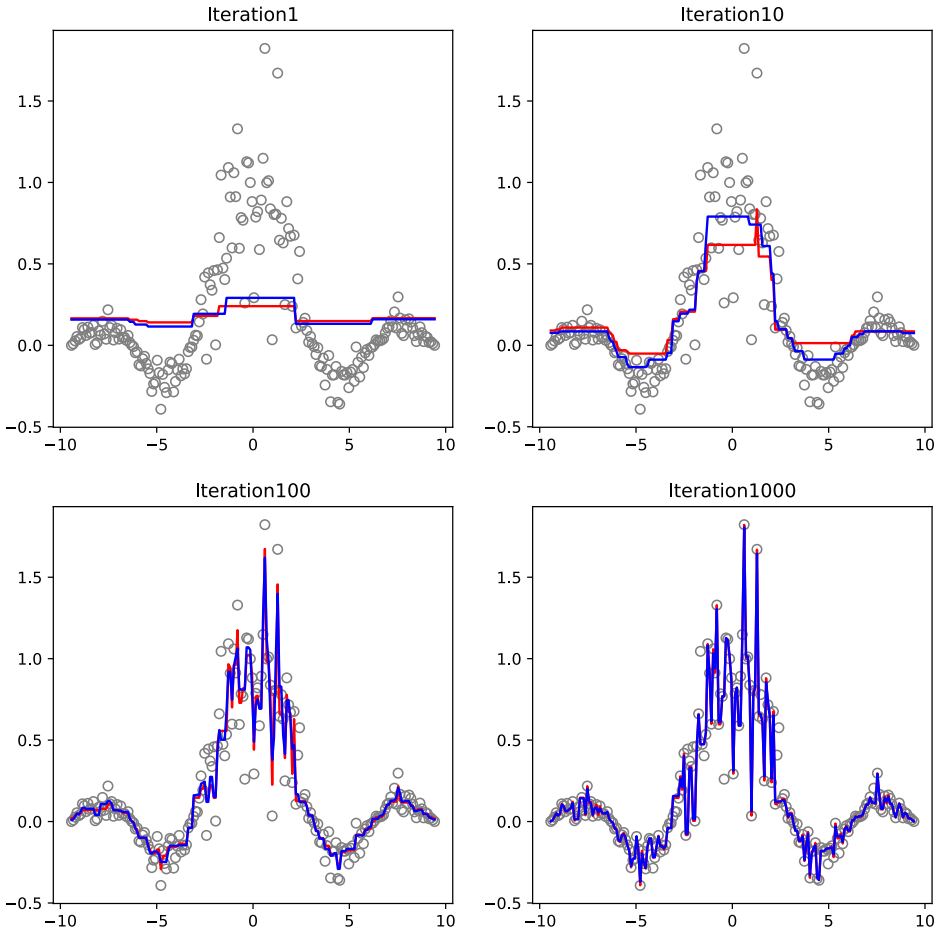
Iteration1
Iteration10
Iteration100
Iteration1000
1.5
1.0
0.5
0.0
−0.5
−10
−5
0
5
10

Figure 3: 123

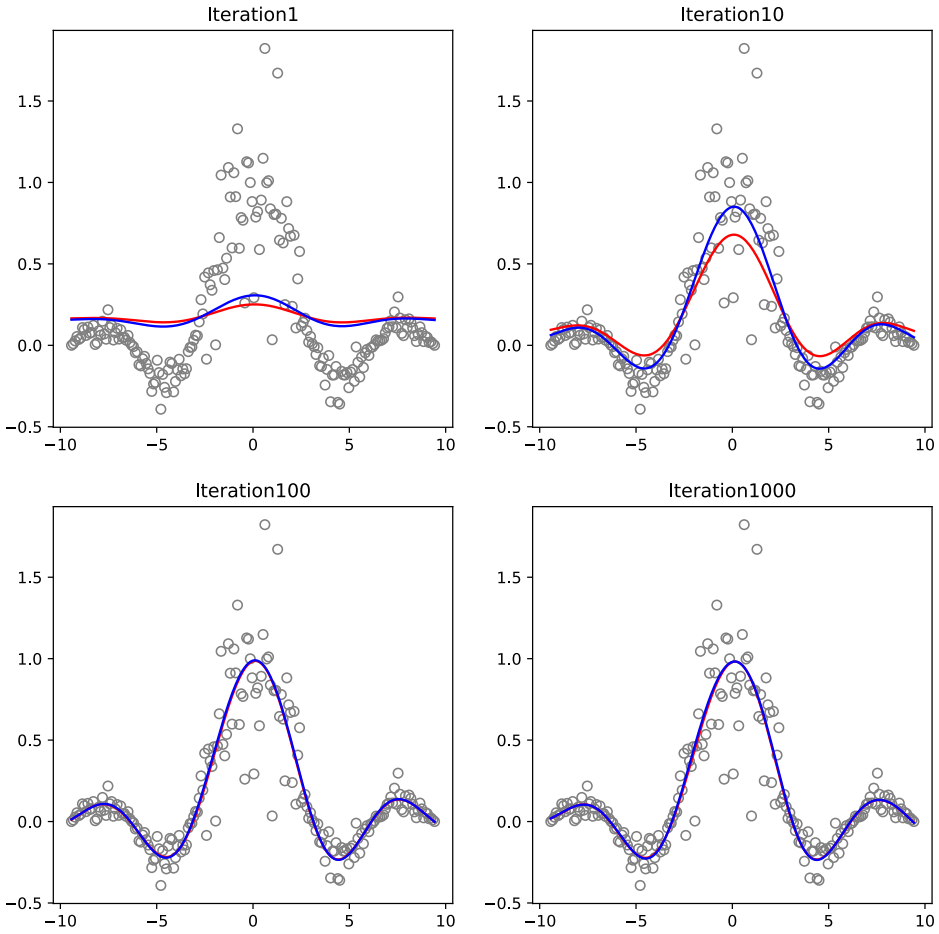
Iteration1
Iteration10
Iteration100
Iteration1000
1.5
1.0
0.5
0.0
−0.5
−10
−5
0
5
10

Figure 4: 123

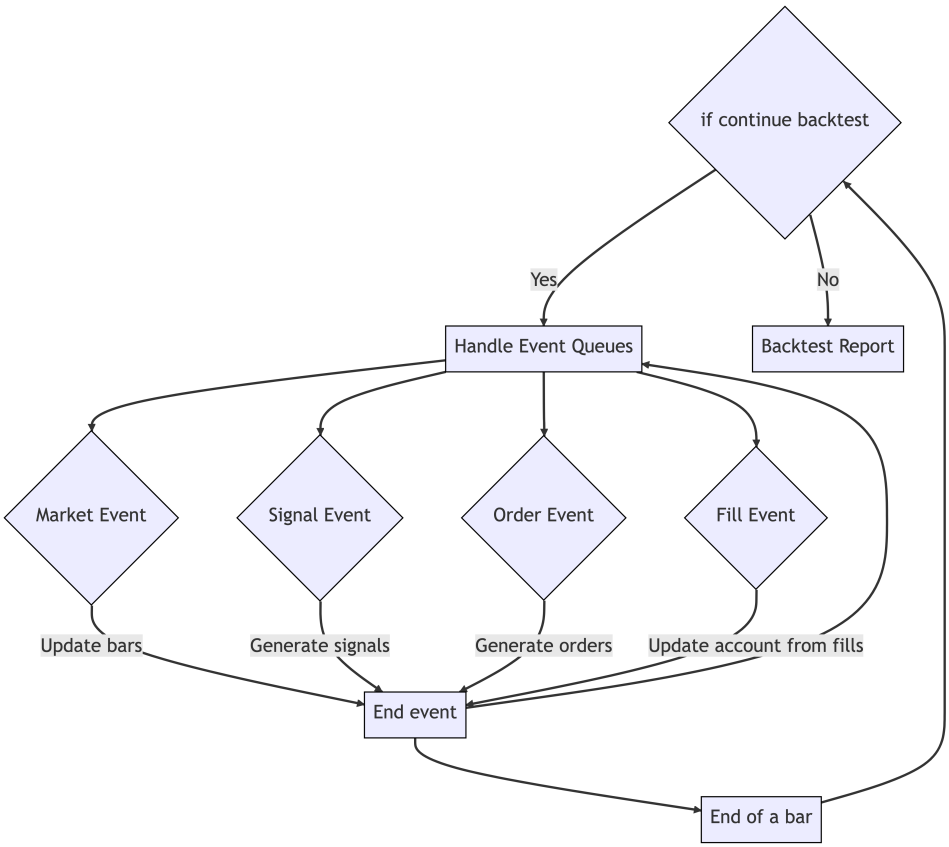

Figure 5: Event-Driven Trading System

5 Conclusion

This paper presents a new gradient boosting algorithm called RGBoost, which enhances the accuracy of the gradient function approximation by modifying the negative gradient at each iteration. We define the gradient function strictly using the Riesz representation theorem and demonstrates that the gradient vector in the conventional gradient boosting algorithm is biased if the hypothesis is a Reproducing Kernel Hilbert Space (RKHS). The updated model performs significantly better than the previous model in simulated data and is subsequently applied in quantitative finance. The paper also includes an overview of related works in gradient boosting and ensemble learning, and describes the self-developed RGBoost module in Python.

A Appendix

A.1 Proof of Theorem 5

Proof. From the chain rules(theorem **??**) we have the following

$$\begin{aligned} D_{\mathcal{L},f} &= \frac{1}{n}\sum_{i=1}^{n} D_{L\circ E_{x_i},f} \\ &= \frac{1}{n}\sum_{i=1}^{n} D_{L,f(x_i)} \circ D_{E_{x_i},f}. \end{aligned}$$

In addition,

$$\begin{aligned} D_{\mathcal{L},f}(h) &= \frac{1}{n}\sum_{i=1}^{n} D_{L,f(x_i)} \circ D_{E_{x_i},f} \circ h \\ &= \frac{1}{n}\sum_{i=1}^{n} \partial_2 L(y_i, f(x_i))\langle K_{x_i}, h\rangle \\ &= \langle \frac{1}{n}\sum_{i=1}^{n} \partial_2 L(y_i, f(x_i))K_{x_i}, h\rangle. \end{aligned}$$

Now the Riesz representation theorem suggests that the gradient of $\mathcal{L}$ at f is the first element in the inner product notation. □

A.2 Proof of Theorem 6

Proof. First suppose that $g \circ f$ is a simple measurable function on Ω, that is

$$g \circ f = \sum_{k=1}^{n} c_k \chi_{F_k}, \quad F_k \in \mathcal{F}.$$

Easy to verify that

$$g \circ f \circ f^{-1} = \sum_{k=1}^{n} c_k \chi_{f(F_k)},$$

Also, we have

$$\begin{aligned}
\int_{\mathbb{R}} g \circ I \mathrm{d}\mu_f &= \int_{\mathbb{R}} \sum_{k=1}^{n} c_k \chi_{f(F_k)} \mathrm{d}\mu_f \\
&= \sum_{k=1}^{n} c_k \mu_f \circ f(F_k) \\
&= \sum_{k=1}^{n} c_k P \circ f^{-1} \circ f \circ F_k \\
&= \sum_{k=1}^{n} c_k P(F_k) \\
&= \int_{\Omega} g \circ f \mathrm{d}P.
\end{aligned}$$

Since the simple-measuralbe functions are dense in $\mathcal{L}^1(\Omega, \mathcal{F}, P)$, we claim that equation 6 holds if $g \circ f \in \mathcal{L}^1(\Omega, \mathcal{F}, P)$. □

A.3 Proof of Theorem 8

Proof. From theorem 14 we know that $m \circ X$ is $\mathcal{F}$-measurable. Next we need to verify whether the partial average of $m \circ X$ equals to that of Y.

$$\begin{aligned}
\int_A m \circ X \mathrm{d}P &= \int_{\Omega} \chi_A m \circ X \mathrm{d}P \\
&= \int_{\mathbb{R}^p} \chi_{X(A)}(x) m(x) \mathrm{d}\mu_X(x) \\
&= \int_{\mathbb{R}^p} \chi_{X(A)}(x) m(x) f_X(x) \mathrm{d}\lambda^p(x) \\
&= \int_{\mathbb{R}^p} \chi_{X(A)}(x) f_X(x) \frac{\int_{\mathbb{R}} y f(x,y) \mathrm{d}\lambda(y)}{f_X(x)} \mathrm{d}\lambda^p(x) \\
&= \int_{\mathbb{R}^p} \int_{\mathbb{R}} \chi_{X(A)}(x) y f(x,y) \mathrm{d}\lambda(y) \mathrm{d}\lambda^p(x) \\
&= \int_{\mathbb{R}^{p+1}} \chi_{X(A)}(x) y \mathrm{d}\mu_{X,Y}(x,y) \\
&= \int_A Y \mathrm{d}P,
\end{aligned}$$

where the second and last equaty follows from Theorem 6. According to the definition of conditional expectation, we know that $m \circ X$ is the conditional expectation of Y with respect to the sigma algebra generated by X. □

B Properties of Conditional Expectation

Defintion 11 (The conditional Expectation with respect a sigma-algebra)**.**
Suppose $\mathcal{G}$ is sub-sigma-algebra of $\mathcal{F}$, Y is random variable. A random variable M is called the conditional expectation if

1. M is $\mathcal{G}$-measurable.
2. $\forall A \in \mathcal{G}$, $\int_A X \mathrm{d}P = \int_A M \mathrm{d}P$.

M is often denoted by $E(X|\mathcal{G})$.

Theorem 12. Radon-Nikodym Theorem Suppose μ is a sigma-finite measure on a measurable space $(X, \mathcal{S})$. Suppose ν is a sigma-finite on $X, \mathcal{S}$ such that $\nu << \mu$. Then there exists $h \in \mathrm{L}^1(\mu)$ such that

$$\mathrm{d}\nu = h \mathrm{d}\mu.$$

Defintion 13. The conditional Expectation with respect a random map
Suppose X is measurable map from Ω to $\mathcal{E}$(e.g. $\mathbb{R}^p$) and Y is a random variable, then the conditional expecation of Y with respect to X is defined as

$$E(Y|X) = E(Y|X^{-1}(\mathcal{E})).$$

Theorem 14. Suppose $X : \Omega \to E$, then $Y : \Omega \to \mathbb{R}$ is $\sigma(X)$-measurable if and only if there exists $h : E \to \mathbb{R}$, $h \in \mathcal{E}$, such that

$$Y = h \circ X.$$

Example 15. Suppose $E \subset \mathbb{R}^n$, $(\Omega, \mathcal{F}, P)$ is a probability space, $X : \Omega \to E$(a random vector).

References

[1] L. Breiman. Random forests. *Machine learning*, 45:5–32, 2001.

[2] T. Chen, T. He, M. Benesty, V. Khotilovich, Y. Tang, H. Cho, K. Chen, R. Mitchell, I. Cano, T. Zhou, et al. Xgboost: extreme gradient boosting. *R package version 0.4-2*, 1(4):1–4, 2015.

[3] Y. Freund, R. E. Schapire, et al. Experiments with a new boosting algorithm. In *icml*, volume 96, pages 148–156. Citeseer, 1996.

[4] J. H. Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[5] F. Hoffmann. Combining boosting and evolutionary algorithms for learning of fuzzy classification rules. *Fuzzy Sets and Systems*, 141(1):47–58, 2004.

[6] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30, 2017.

[7] A. Natekin and A. Knoll. Gradient boosting machines, a tutorial. *Frontiers in neurorobotics*, 7:21, 2013.

[8] M. Schmid, T. Hothorn, K. O. Maloney, D. E. Weller, and S. Potapov. Geoadditive regression modeling of stream biological condition. *Environmental and Ecological Statistics*, 18:709–733, 2011.

[9] B. Schölkopf, R. Herbrich, and A. J. Smola. A generalized representer theorem. In *Computational Learning Theory: 14th Annual Conference on Computational Learning Theory, COLT 2001 and 5th European Conference on Computational Learning Theory, EuroCOLT 2001 Amsterdam, The Netherlands, July 16–19, 2001 Proceedings 14*, pages 416–426. Springer, 2001.

[10] F. Sigrist. Ktboost: Combined kernel and tree boosting. *Neural Processing Letters*, 53(2):1147–1160, 2021.

致谢

感谢国家，感谢家人，感谢老师，感谢同学。大学四年里中一直铭记着叶老师的三句话，自学能力，信息搜集能力，信息分辨能力。自己也在本科期间学会了以前不曾设想的知识。希望在硕士阶段能够保持，更近一步。