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	

RGBoost: A revised gradient boosting machine [*]

None[†]

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.

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

[*]None 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. 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. Later, by utilizing the chain rule of Frechet derivatives, we calculate the gradient of the empirical loss function and then embed it into boosting algorithm. At last, the simulated data verify the accuracy of our model, the revised displays a faster convergence rate and smaller test error.

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 the 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 our knowledge, a gradient-boosting machine aims to find an element in the hypothesis space that minimizes the empirical loss function. This can be done through a gradient descent algorithm on a functional space. However, gradient descent on function space requires more sophisticated mathematical tools, and thus we start this section by listing some preliminaries 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. 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, we need a 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.

Another important concept is the Frechet derivative. The Frechet derivative is a way to extend the derivative to function spaces. It measures the rate of change of a functional under small perturbations of its arguments. The Frechet derivative is defined as the best linear approximation of the functional at a given point. To be specific, we define the Frechet derivative of f as a map from $\mathcal{X}$ to a linear functional $D_{f,x}$ on $\mathcal{X}$,

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

To get an insight into D_f, we take a multivariate function as an 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 widely used theorem for understanding function spaces' structure. This theorem has many applications in mathematics, physics, and even economics. For example, the Riesz representation theorem in finance implies that a payoff can represent any linear functional on the asset span.

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 essential in our model, we need a better space to discuss it. Note that not all the evaluation functional of a Hilbert space are bounded. To meet the requirement of the Riesz representation theorem requirement, we define the Reproducing Kernel Hilbert space as a space in which every evaluation function is bounded(despite not being 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 bounded, the Riesz Representation Theorem suggests that there exists a unique vector $K_x \in \mathcal{H}$ which satisfies 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, confirming whether the evaluation functional in a Hilbert space is bounded can be challenging. 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 the kernel. As a result, we commonly rely on a symmetric and positive definite function to construct a Hilbert space, 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 the gradient. First, recall the definition of the gradient in calculus. It is a vector that points in the direction of the steepest increase of the function at a particular point. It can be calculated by taking the partial derivative with respect to each coordinate. The inner product of a gradient and the increment can be viewed as the linear approximation of the increment of the function value. This idea can be directly extended to a function space. Specifically, suppose $\mathcal{H}$ is a normed vector 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 $\mathcal{H}$ and the error part must moves faster to zero than h, that is

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

Note that the zero on the 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.

Now, 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 reminds us to define the gradient of function as the unique element in the Hypothesis space that represents the Frechet derivative of the functional.

Suppose f is a 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 an RKHS. Then the gradient of the evaluation functional at f when given x is

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

Proof.

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

The Riesz Representation theorem admits the uniqueness of K_x. □

Suppose $\|h\| = 1$, we have the following:

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

This provides a way to find the local minimum of a functional. Indeed, the equation above suggests that L moves the fastest when walking along the gradient's direction, thus giving us an idea to minimize a functional. We state this in Algorithm 1.

```
Algorithm 1: Function Gradient descend algorithm
  Input:
      • the functional $L : V \to \mathbb{R}$.
      • number of iterations $M \geq 1$
      • leanring rate $\eta \in (0, 1]$
1 Initialize $f_0$
2 while $m \in \{1, \cdots, M\}$ do
3     Calculate the gradient:
          $\nabla L(f_{m-1})$
4     Find the best gradient descent step-size:
          $\rho_m = \arg\min_{\rho \in \mathbb{R}} L(f_{m-1} - \rho \nabla L(f_{m-1})))$
5     Update:
          $f_m \leftarrow f_{m-1} - \eta \rho_m \nabla L(f_{m-1})$
6 end
  Output: $f_M$
```

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

Since the gradient function can only be observed at finite sample points, we focus on the empirical case. Our goal is to find the local minimum 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 starting the gradient descent algorithm, we define another functional $\mathcal{L}'$, which is derived from $\mathcal{L}$:

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

where h_i represents the i-th coordinate of vector h. Next, we will examine the relationship between $\mathcal{L}$ and

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

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

With knowledge of calculus, the gradient of a multivariate function is composed of partial derivatives, as shown in Equation (5). The negative gradient vector in Friedman's boosting algorithm is represented by $\nabla \mathcal{L}'(\hat{y})$. Algorithm 2 outlines the implementation of this algorithm. Note that $\partial_2 L$ refers to the partial derivative of the second component.

Algorithm 2: Friedman's Gradient Boosting algorithm

```
   Input:
     • the functional $L : V \to \mathbb{R}$.
     • number of iterations $M \geq 1$
     • leanring rate $\eta \in (0, 1]$
1  Initialize $F_0 = \arg\min_\rho \sum_{i=1}^n L(y_i, \rho)$
2  while $m \in \{1, \cdots, M\}$ do
3      Calculate the negative gradient:
           $g_{m,i} = -\left[\frac{\partial L(y_i, F(\mathbf{x}_i))}{\partial F(\mathbf{x}_i)}\right]_{F(\mathbf{x})=F_{m-1}(\mathbf{x})}, \; i = 1, \cdots, n$
4      Fit a base estimator:
           $\mathbf{a}_m = \arg\min_{\mathbf{a},\beta} \sum_{i=1}^n [g_{m,i} - \beta h(\mathbf{x}_i; \mathbf{a})]^2$
5      Find the best gradient descent step-size:
           $\rho_m = \arg\min_\rho \sum_{i=1}^n L(y_i, F_{m-1}(\mathbf{x}_i) + \rho h(\mathbf{x}_i; \mathbf{a}_m))$
6      Update:
           $F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \rho_m h(\mathbf{x}; \mathbf{a}_m)$
7  end
   Output: $F_M$
```

In order to maintain consistency with our model's notation, we will list the algorithm without changing any symbols. However, note that the notation $\frac{\partial L(y_i, F(\mathbf{x}_i))}{\partial F(\mathbf{x}_i)}$ is equivalent to $\partial_2 L(y_i, F(x_i))$.

In step 4 of the algorithm, a base learner is chosen to fit the gradient vector in L^2 norm at sample points. This learner can be viesw as the gradient function in algorithm 1.

2.2 The revised gradient boosting machine

This section showcases our revised model. It's vital to remember that the negative gradient vector in algorithm 2 represents the gradient of the derived empirical loss functional $\mathcal{L}'$, not the gradient of $\mathcal{L}$. While it may appear challenging to calculate the gradient of the empirical loss functional $\mathcal{L}$ at f, theorem 4 offers an explicit expression for 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 the 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 $f_0$,    $f_0(x) = \arg\min_{c\in\mathbb{R}} \frac{1}{n} \sum_{i=1}^{n} L(y_i, c)$ 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 version of the gradient boosting machine. In Step 1, we search for a constant function that minimizes the empirical loss function. If we apply the square loss function, this constant must be the sample mean of y. In Step 3, we substitute the gradient based on Theorem 5.

2.3 Statistical theory of regression

This subsection explains the statistical theory of regression. As previously discussed, the gradient boosting machine (GBM) can be used to approximate the function that minimizes the empirical loss function. Therefore, the choice of loss function and function to be approximated deserves discussion. We summarize the statistical theory of regression and demonstrate that the GBM can approximate the conditional expectation function $m(x) = E(Y|X = x)$ by using the L^2 loss function.

To clarify these concepts, we will discuss them within the framework of measure theory. We begin with a measurable space comprising a set and a sigma-algebra defined on it. A measure can be defined on a measurable space, which is a set function that satisfies countable additivity. Integration with respect to measure has superior properties to Riemann integration, which we learned about in calculus. Although Riemann integration is intuitive, it loses its applicability when the function is defined in an abstract space. However, Lebesgue integration offers numerous advantages over Riemann integration. One of its primary benefits is its generality, as it can integrate a broader range of functions, including those not Riemann integrable. Lebesgue integration is also more flexible, enabling the integration of functions with respect to more general measures rather than just the Lebesgue measure. Furthermore, Lebesgue integration has superior convergence properties to Riemann integration, making it particularly suitable for numerous analysis and probability theory applications. The integration of a random variable with respect to a probability measure has a 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. □

Theorem 6 has many applications in probability theory. For example, we can calculate the expectation of $g(X)$ by using the formula $E(g(X)) = \int_{\mathbb{R}} g(x)\mathrm{d}F_X(x)$.

The objective of regression is to identify the conditional expectation function, which is a key concept in probability theory and statistics. It represents the anticipated value of a random variable based on certain information provided by another random variable. Appendix B provides a definition of conditional expectation, as well as its properties. According to the following theorem, the conditional expectation with respect to a sigma-algebra generated 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. $\square$

The converse of Theorem 7 is also true, and it provides an efficient method for calculating the conditional expectation function so long as the sigma algebra is generated by a random map, since $E(Y|X = x)$ can often be easily obtained.

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

$$m \circ X = E(Y|\sigma(X)). \tag{7}$$

Proof. The proof can be found in Appendix A. $\square$

To gain a deeper understanding of conditional expectation and regression, we relax the restrictions on Y slightly. Specifically, we assume that $Y \in \mathcal{L}^2(\Omega, \mathcal{F}, P)$ and $\mathcal{G}$ is a sub-sigma-algebra of $\mathcal{F}$. We can then consider the space $\mathcal{L}^2(\Omega, \mathcal{G}, P_{\mathcal{G}})$, which is a closed subspace of $\mathcal{L}^2(\Omega, \mathcal{F}, P)$. According to the Hilbert project theorem, there exists a unique element M in $\mathcal{L}^2(\Omega, \mathcal{G}, P_{\mathcal{G}})$ such that

$$\|Y - M\|_2 \leq \|Y - f\|_2, \quad \forall f \in \mathcal{L}^2(\Omega, \mathcal{G}, P_{\mathcal{G}}), \tag{8}$$

and

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

The partial average of M is equal to that of Y, making it sufficient to serve as a conditional expectation. In this sense, we can see the conditional expectation as a projection onto the subspace created by $\mathcal{G}$. Figure 1 reveals the relationship of Y, $E(Y|\mathcal{G})$ and $E(Y)$. Note that the variance decomposition formula is the Pythagoras theorem in a Hilbert space.

$$Var(Y) = Var(E(Y|X)) + E(Var(Y|X)). \tag{9}$$

The regression model can be structed as follows, suppose we have a random variable Y defined on a probability space $(\Omega, \mathcal{F}, P)$, and a random map X from Ω to a vector space E (often denoted as $\mathbb{R}^p$). We

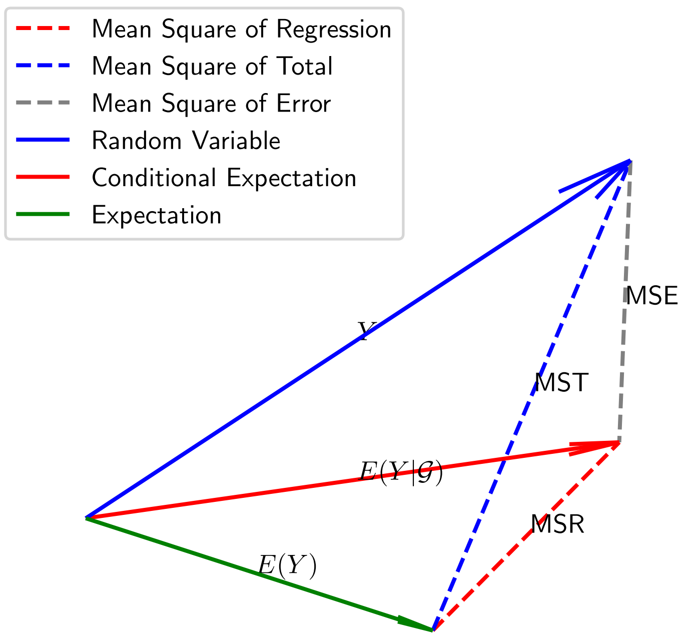

Figure 1: Conditional Expectation as a projection.

can express Y as the sum of its conditional expectation $E(Y|X)$ and the error term $Y - E(Y|X)$, which can be written as $m \circ X + \varepsilon$.

To estimate the conditional expectation from sample data, we need to minimize the empirical loss function:

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

where y_i is the value of Y and x_i is the value of X in the i-th sample, and $\mathcal{H}$ is a suitable class of functions. The goal is to find a function f that minimizes the difference between the predicted values and the actual values, as measured by the mean squared error.

There are many ways to optimize the empirical loss function. One of the simplest ways is to parameterize it. For instance, we can set the form of the hypothesis as follows:

There are various methods to optimize the empirical loss function. One of the simplest ways is to parameterize it. For example, we can define the form of the hypothesis as follows:

$$\mathcal{H} = \left\{ f : \mathbb{R}^p \to \mathbb{R}, \quad x \mapsto c + \beta^T x | \text{where } \beta \in \mathbb{R}^p, c \in \mathbb{R} \right\}. \tag{10}$$

In this case, the problem can be converted to:

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

where $\mathcal{H}$ is often referred to as the collection of affine functions, and the β that minimize the square loss is

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

The hypothesis space can also be represented as a reproducing kernel Hilbert space (RKHS). Assuming that $K : \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R}$ is a positive definite kernel function, there exists an inner product in $\mathcal{H}$ such that K_x belongs to $\mathcal{H}$ for all $x \in \mathbb{R}^p$. According to the representer theorem [9], there is a unique minimizer in the form of

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

Moreover, the GLM can be utilized as a Hypothesis. In this scenario, $\mathcal{H}$ can be expressed as follows:

$$\mathcal{H} = \left\{ f : \mathbb{R}^p \to \mathbb{R}, \quad x \mapsto g^{-1} \circ (c + \beta^T x) | \text{where } \beta \in \mathbb{R}^p, c \in \mathbb{R} \right\}, \tag{11}$$

Here, g^{-1} refers to the inverse of the canonical link function and varies for different cases.

Table 1: The Canonical Link Function of GED.

Distribution	Mean Function	Canonical Link Function
$N(\mu, \sigma^2)$	$b' : \theta \mapsto \theta$	$g : \mu \mapsto \mu$
$Pois(\lambda)$	$b' : \theta \mapsto \mathrm{e}^\theta$	$g : \mu \mapsto \ln \mu$
$Bin(1, p)$	$b' : \theta \mapsto \frac{1}{1+\mathrm{e}^{-\theta}}$	$g : \mu \mapsto \ln \frac{\mu}{1-\mu}$

3 GBM Design

In this section, we will list the alternative loss functions and base learners, and explain when they can be used. Later, we will discuss some improvement approaches to GBM, such as combining different types of base learners, regularization, subsampling, and others. For more information about the design of GBM, we refer to NateKin [7].

To create a GBM for a particular task, it's important to choose the appropriate options for the empirical loss function $\mathcal{L}$ and the hypothesis $\mathcal{H}$ (which is the set of collections of base learners), because these choices have a significant impact on the GBM's performance. (As we discussed in the previous section, the conditional expectation function minimizes 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 framework allows for the use of custom loss functions. One example of their application is in quantitative finance and investment analysis, particularly in stock selection. At each rebalance date, we have a multitude of factors denoted by $F_t \in \mathbb{R}^{m \times N}$, where m is the number of factors and N is the number of stocks. These factors include historical prices, financial ratios, and technical indicators. 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 of future retures. This can be viewed as a custom 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

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

Here we use the L^2 loss function for illustration.

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

Theorem 9 provides ideas for choosing the loss function, but in practical applications, we also need to estimate based on the sample. Therefore, we propose the following method of converting from the population to the sample:

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

that is, to use two vectors in $\mathbb{R}^n$ to approximate the distance between random variables.

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 importtant 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 2: 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 $f_0$,   $f_0(x) = \arg\min_{c \in \mathbb{R}} \frac{1}{n} \sum_{i=1}^{n} L(y_i, c)$ 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: $B_t, t = 1, \cdots, M$.
1 Initialize $f_0$,    $f_0(x) = \arg\min_{c \in \mathbb{R}} \frac{1}{n} \sum_{i=1}^{n} L(y_i, c)$ for all $x \in \mathbb{R}^p$.
2 while $m \in \{1, \cdots, M\}$ do
3     Calculate the gradient vector:
          $g_{m,j} = -\frac{1}{|B_m|} \sum_{i \in B_m} \partial_2 L(y_i, f_{m-1}(x_i)) K(x_i, x_j), \quad j \in B_m.$
4     Fit the gradient vector:
          $h_m = \arg\min_{h \in \mathcal{H}, \beta} \sum_{i \in B_m} (g_{m,i} - \beta h_m(x_i))^2$
5     Find the best gradient descent step-size:
          $\rho_m = \arg\min_{\rho \in \mathbb{R}} \mathcal{L}(\mathbf{f}_{m-1} + \rho h_m)$
6     Update:
          $f_m \leftarrow f_{m-1} + \eta \rho_m h_m$
7 end
Output: $f_M = f_0 + \sum_{i=1}^{M} \eta \rho_m h_m$
```

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

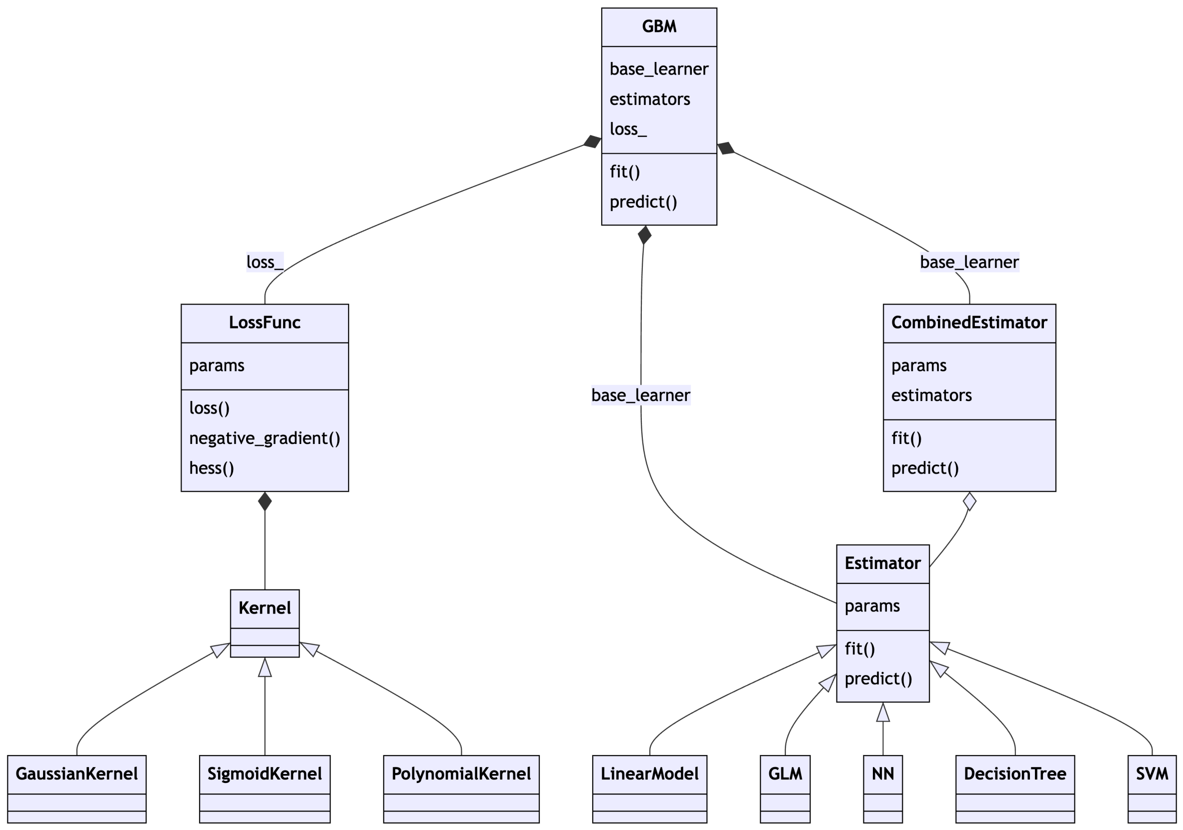

Figure 2: framework of rgboost

This section introduces the RGBoost[1]. 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.

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

4 Application

In this section, we apply the gradient boosting machine to function approximation. To be specific, we use synthetic dataset simulated from $\text{sinc}(x)$[2]. The simulated $\text{sinc}(x)$ 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 $\text{sinc}(x)$ function as the target variable. The $\text{sinc}(x)$ function is defined as $\text{sinc}(x) = \frac{\sin(x)}{x}$. 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 $\text{sinc}(x)$ dataset is often used to test the accuracy and generalization ability of different regression models, including gradient boosting models.

4.1 Function Approximation

In this subsection, we apply the revised gradient boosting machine into simulated data application. We first split the sincx dataset into training and testing sets. The model was then trained using the training set and evaluated using the testing set.

The result is shown in figure 3. Our first step was to compare the performance of GBDT on the testing set. The model using the revised gradient approach (indicated by the blue curve in the learning curve) showed a faster initial descent rate and a smaller testing error. For Kernel-based learners(here we use gaussian kernels for illustration), the model using gradient improvement still significantly reduced testing error in the early iterations, but gradually converged after 200 iterations. Lastly, we combined RKHS with decision trees, but the results indicated that this combination did not always improve the testing error of the model. This is easily understood because our aim is to fit a smooth function, and the RKHS-based gradient boosting model has a natural advantage in this regard. Therefore, the combined model did not perform as well as the individual models, which was expected. Next, Figure 3 and Figure 4 depict the process of model fitting. As we can observe from the figures, the revised gradient boosting algorithm (denoted by the blue curve) can better approximate the sample points and achieve good results with few iterations, while keeping other parameters constant. It is also important to note that due to the small sample size, the model quickly begins to overfit. Moreover, when the base learner is an RKHS, the improved model can also accelerate the convergence speed of the model.

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

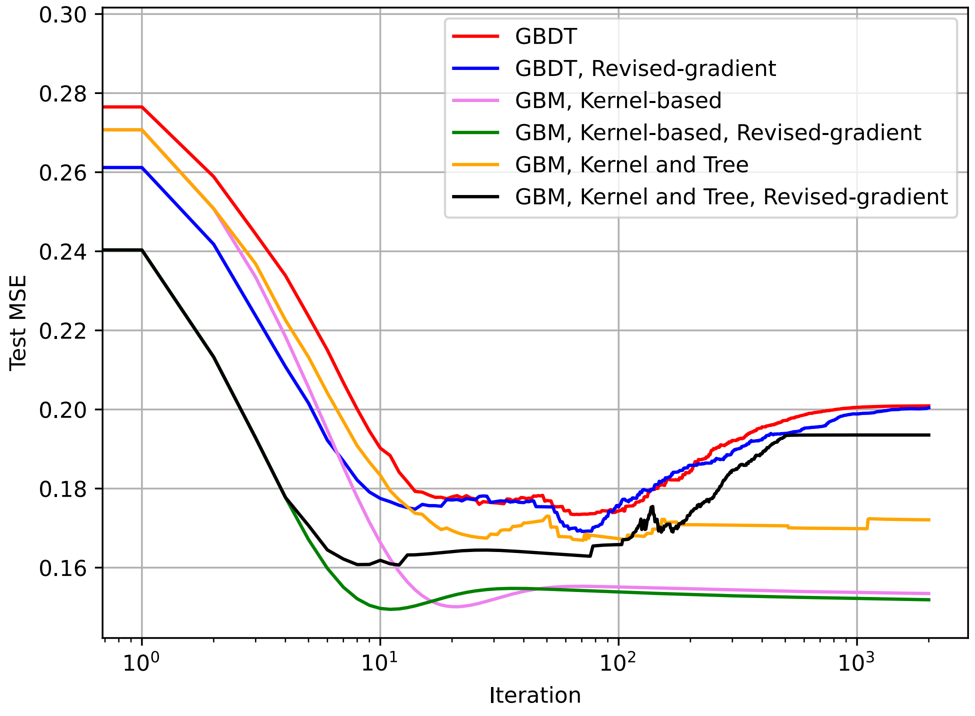

Figure 3: Test MSE versus the number of iteration for comparing GBM with/without revised gradient.

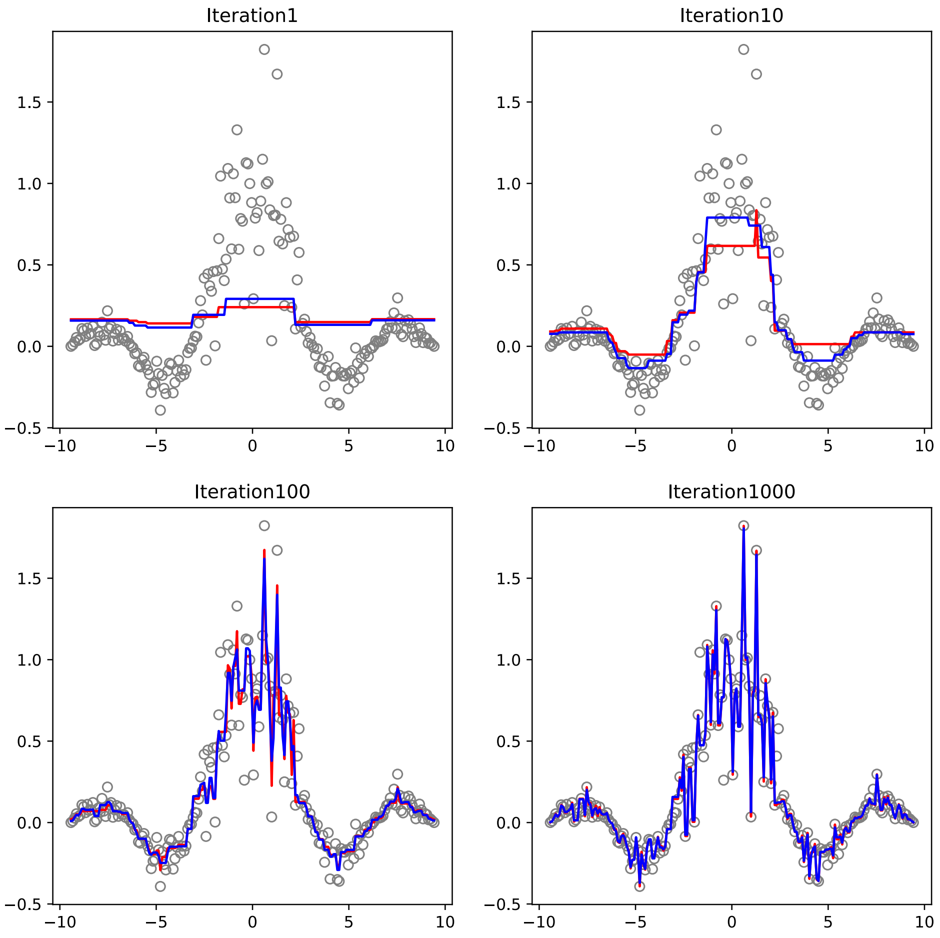

Figure 4: Training process of GBDT.

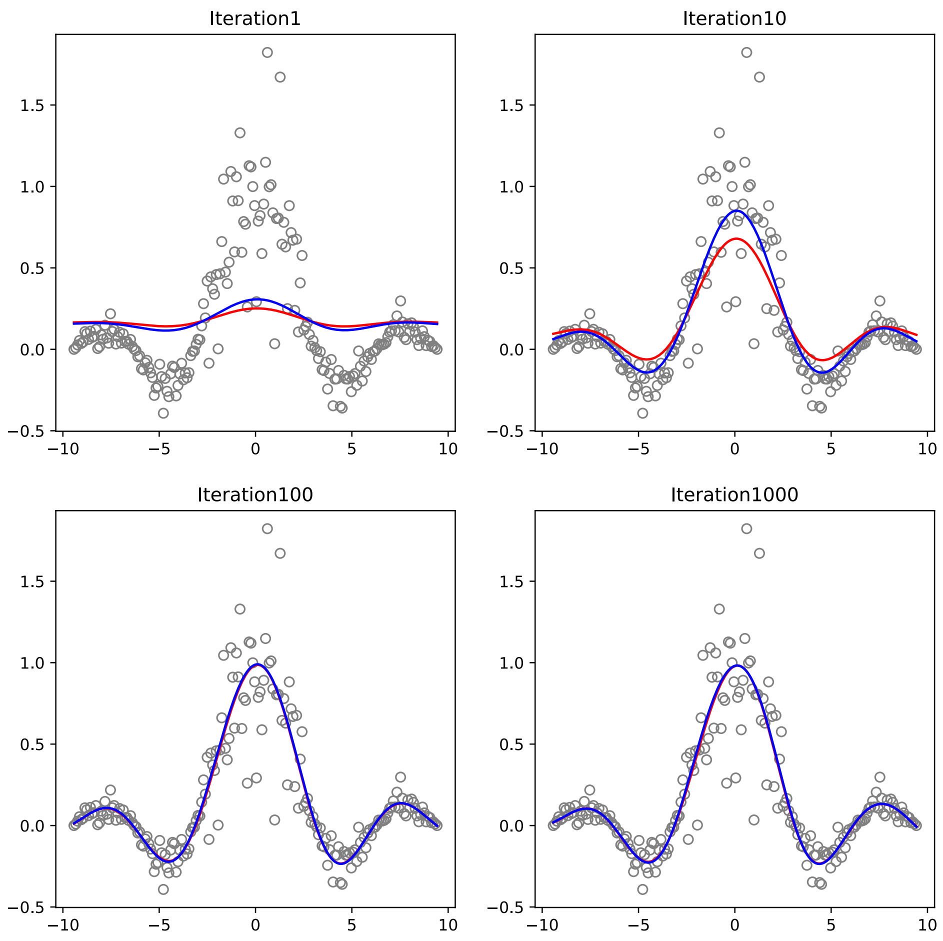

Figure 5: Training process of GBM based on RKHS.

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 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 $h \in \mathcal{L}^1(\mu)$ 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

$$Y = h \circ X.$$

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。