Оптимизация ансамблей градиентного бустинга через адаптивное обучение агрегирующих функции

A Preprint

Якушевич Антон Сергеевич
ВМК
МГУ
Москва
s02220301@gse.cs.msu.ru

Сенько Олег Валентинович
ВМК
МГУ
Москва
senkoov@mail.ru

Abstract

В работе исследуется задача оптимизации ансамблей градиентного бустинга за счёт выбора обобщённой агрегирующей функции. Исследование проводится для уменьшения количества деревьев в ансамбле с целью повышения интерпретируемости модели при сохранении её прогностической точности. Для этого предлагается использовать усечённое разложение агрегирующей функции в ряд Тейлора и оптимизировать её коэффициенты с помощью градиентного спуска, что позволяет адаптивно настраивать структуру ансамбля и улучшать баланс между точностью и интерпретируемостью. Экспериментальные результаты показывают, что оптимизация коэффициентов ряда Тейлора позволяет сократить число деревьев в ансамбле при сохранении точности модели, а также повышает устойчивость и эффективность обучения по сравнению с классическим градиентным бустингом.

Keywords Gradient Boosting · Ensemble Methods · Aggregation Function Optimization · Meta-Learning · Taylor Expansion

1 Введение

Градиентный бустинг остаётся одним из наиболее эффективных методов машинного обучения, применяемых в задачах регрессии и классификации. Однако высокая точность ансамбля достигается ценой увеличения числа деревьев, что усложняет интерпретацию модели и приводит к увеличению вычислительных затрат. Поэтому важной задачей является разработка подходов, позволяющих уменьшить размер ансамбля без ухудшения качества прогнозов.

Одним из направлений снижения сложности ансамблей является модификация агрегирующей функции. Наиболее простой подход заключается во введении весов для отдельных деревьев, которые подбираются либо вручную, либо оптимизируются на валидационных данных [Domke, 2012]. Более сложные методы рассматривают автоматическую настройку весов или коэффициентов, что позволяет повысить адаптивность ансамбля [Franceschi et al., 2017]. Подобные подходы расширяют пространство возможных агрегирующих функций и потенциально позволяют достичь более высокой точности. В смежных направлениях рассматривались и дифференцируемые методы оптимизации гиперпараметров [Maclaurin et al., 2015, Lorraine et al., 2019, Shaban et al., 2019], где подбираются параметры регуляризации и обучения, что демонстрирует применимость автоматического дифференцирования к задачам настройки сложных моделей.

Несмотря на успехи, существующие подходы имеют ряд ограничений. Во-первых, во многих исследованиях агрегирующая функция задаётся заранее, как простое суммирование или линейная комбинация, что снижает адаптивность ансамбля [Domke, 2012]. Во-вторых, даже в случаях, когда веса подбираются автоматически, процесс оптимизации часто является дорогостоящим и требует перебора гиперпараметров

или применения эвристик [Franceschi et al., 2017]. В-третьих, существующие подходы не предусматривают влияния агрегирующей функции на процесс обучения базовых моделей, вследствие чего структура ансамбля формируется независимо от выбранного способа агрегирования. Это делает такие методы менее гибкими и усложняет их использование на практике.

В данной работе агрегирующая функция ансамбля представляется в виде усечённого разложения в ряд Тейлора, где коэффициенты ряда рассматриваются как параметры, подлежащие обучению. Для оптимизации коэффициентов агрегирующей функции вводится оператор, который принимает на вход вектор коэффициентов, оценивает качество модели по кросс-валидации, обучая градиентный бустинг с использованием этих коэффициентов, и возвращает полученное значение функции потерь. Это значение используется как целевая функция при обновлении коэффициентов методом градиентного спуска, что позволяет напрямую минимизировать валидационную ошибку и адаптивно подстраивать структуру ансамбля под задачу.

Предложенный метод открывает новый способ интеграции оптимизации агрегирующих функций в процесс обучения ансамблей. В отличие от подходов, где агрегирующая функция задаётся заранее или обучается постфактум, наша методика позволяет адаптивно корректировать её форму одновременно с построением деревьев. Результаты экспериментов подтверждают, что оптимизация коэффициентов агрегирующей функции позволяет добиться меньших значений ошибки при фиксированном числе деревьев по сравнению с классическим градиентным бустингом. Это демонстрирует эффективность предлагаемого подхода и его потенциал для построения более компактных и интерпретируемых ансамблей без потери прогностической мощности. Вклад работы заключается в том, что она расширяет область применения методов оптимизации, показывая, что обучение агрегирующих функций может быть встроено непосредственно в процесс градиентного бустинга, а не рассматриваться как внешний этап.

2 Литературный обзор

Ансамблевые методы давно зарекомендовали себя как один из наиболее эффективных классов алгоритмов машинного обучения [Breiman, 1996, Freund and Schapire, 1997]. Среди них особое место занимает градиентный бустинг [Friedman, 2001], который благодаря высокой точности и универсальности получил широкое распространение как в академических исследованиях, так и в индустриальных приложениях [Chen and Guestrin, 2016, Ke et al., 2017]. Однако рост числа деревьев в ансамбле приводит к ухудшению интерпретируемости и повышению вычислительных затрат, что стимулировало поиск методов сокращения сложности без потери качества.

Одним из направлений развития бустинга стали методы регуляризации и усечения ансамблей. Ряд работ показал эффективность ограничений на глубину деревьев и скорости обучения [Zhang and Yu, 2005, Bühlmann and Hothorn, 2007], а также введения различных форм регуляризации [Mason et al., 2000, Friedman, 2002]. Другой класс подходов связан с сокращением ансамбля путём отбора наиболее информативных моделей [Margineantu and Dietterich, 1997, Caruana et al., 2004], что позволяет уменьшить число деревьев при минимальных потерях в точности.

Ключевым элементом ансамблевых методов является агрегирующая функция, которая объединяет предсказания отдельных базовых алгоритмов в итоговый результат. В классических вариантах бустинга используется простая сумма или усреднение выходов деревьев с фиксированными весами [Hastie et al., 2009]. Однако такая схема имеет ограниченную гибкость и не всегда позволяет учесть различный вклад деревьев в итоговую точность. Для повышения адаптивности были предложены методы взвешивания, где каждому дереву приписывается коэффициент, подбираемый либо с помощью оптимизации на валидационных данных, либо через регуляризацию [Domke, 2012, Franceschi et al., 2017]. Подобные подходы позволяют усилить значимость наиболее полезных моделей и, наоборот, снизить влияние переобученных деревьев.

В смежных исследованиях развивались методы stacking, где агрегирующая функция задавалась в виде отдельной модели, обучаемой на предсказаниях базовых алгоритмов [Wolpert, 1992, Ting and Witten, 1999, Maclin and Opitz, 1999]. В простейшем случае это линейная или логистическая регрессия, а в более сложных вариантах — регуляризованные модели или мета-классификаторы, способные учитывать взаимосвязи между деревьями. Такие методы обеспечивали более богатое пространство комбинаций по сравнению с фиксированными весами, но усложняли интерпретацию и повышали риск переобучения.

Параллельно активно развивалось направление дифференцируемой оптимизации гиперпараметров. Ключевая идея была предложена в работах Maclaurin et al. [2015] и Domke [2012], где рассматривалось дифференцирование процесса обучения для вычисления градиентов валидационной ошибки. Дальнейшие исследования сосредоточились на масштабировании этого подхода [Franceschi et al., 2017, Lorraine et al., 2019, Shaban et al., 2019], использовании неявного дифференцирования и аппроксимаций обратного Гессиана [Pedregosa, 2016], а также применении автоматического дифференцирования в широком классе моделей [Baydin et al., 2018]. Эти работы подтвердили возможность точной и вычислительно эффективной оптимизации параметров, выходящих за рамки традиционных методов перебора и байесовской оптимизации [Snoek et al., 2012, Bergstra et al., 2011].

Несмотря на прогресс, существующие методы имеют ограничения: они не обеспечивают достаточной гибкости, поскольку не учитывают влияние формы агрегирующей функции на процесс обучения базовых моделей, а следовательно, не позволяют оптимизировать структуру ансамбля в зависимости от выбранного способа агрегирования. Дифференцируемые методы гипероптимизации сосредоточены преимущественно на настройке параметров обучения и регуляризации, а не на структурных характеристиках агрегирования. Эти ограничения формируют исследовательский разрыв, восполняемый настоящей работой, где агрегирующая функция представляется в виде усечённого разложения в ряд Тейлора, а её коэффициенты оптимизируются напрямую градиентным спуском в ходе построения ансамбля.

3 Постановка задачи

3.1 Данные

В работе использовались наборы данных для задач регрессии различной природы, отличающиеся по размерности признакового пространства и объёму выборки:

$$\mathscr{D} = \{(x_i, y_i)\}_{i=1}^{n}$$

где $x_i \in \mathbb{R}^d$ - вектор признаков, а $y_i \in \mathbb{R}$ - целевая переменная.

В экспериментах использовался пул из 9 наборов данных для тестирования моделей, а также расширенный пул из 64 задач регрессии, применявшихся для обучения мета-модели.

3.2 Обобщение градиентного бустинга

Обобщим метод градиентного бустинга. Пусть задана последовательность функций $\{F_N\}_{N=1}^{+\infty}$, где $F_N : \mathbb{R}^N \to \mathbb{R}$. На $(N+1)$-ом шаге обучения вместо стандартной суммы $\sum_{i=1}^{N} T_i(x)$ будем использовать обобщённую функцию ансамбля $F_N\left(T_1(x), T_2(x), \ldots, T_N(x)\right)$, где $\{T_i\}_{i=1}^{N}$ – базовые модели (решающие деревья), обученные на предыдущих шагах.

Задача состоит в нахождении такой последовательности функций F_N, которая при фиксированном числе базовых моделей N обеспечивала бы лучшее качество приближения целевой функции $f(x)$ (относительно заданной функции потерь $\mathscr{L}$) по сравнению с классическим градиентным бустингом, использующим сумму в качестве агрегирующей функции.

3.3 Параметризация агрегирующей функции

Зафиксируем число N и рассмотрим функцию от N переменных $F_N(T_1(x), T_2(x), \ldots, T_N(x))$. Разложим её в ряд Тейлора в окрестности точки $(a_1, a_2, \ldots, a_N)$:

$$F_N(T_1(x), T_2(x), \ldots, T_N(x)) = \sum_{k=0}^{+\infty} \left[\overbrace{\sum_{k_1=0} \sum_{k_2=0} \cdots \sum_{k_N=0}}^{k_1+k_2+\cdots+k_N=k} \frac{1}{k_1! k_2! \ldots k_N!} \frac{\partial^k F_N(a_1, a_2, \ldots, a_N)}{\partial T_1^{k_1} \partial T_2^{k_2} \ldots \partial T_N^{k_N}} \prod_{i=1}^{N} (T_i(x) - a_i)^{k_i} \right]$$

Тогда вместо классической линейной комбинации деревьев в решающем лесе предлагается использовать обобщённую функцию ансамбля, которая является усечённым вариантом ряда Тейлора:

$$\widehat{F}_N(T_1(x), T_2(x), \ldots, T_N(x)) = \sum_{k=1}^{M} \left[\overbrace{\sum_{k_1=0} \sum_{k_2=0} \cdots \sum_{k_N=0}}^{k_1+k_2+\cdots+k_N=k} b_{k_1,k_2,\ldots,k_N} \prod_{i=1}^{N} (T_i(x) - a_i)^{k_i} \right]$$

где $M,\ b_{k_1,k_2,\ldots,k_N},\ a_i$ – гиперпараметры

При этом выбор значений $b_{k_1,k_2,\ldots,k_N}$ и a_i позволяет моделировать различные виды функций F_N при $M \to +\infty$.

Если количество используемых переменных меньше общего числа $L < N$ (например при обучении $(L+1)$-го дерева), то функция принимает вид:

$$\widehat{F}_L(T_1(x), T_2(x), \ldots, T_L(x)) = \sum_{k=1}^{M} \left[\overbrace{\sum_{k_1=0} \sum_{k_2=0} \cdots \sum_{k_L=0}}^{k_1+k_2+\cdots+k_L=k} b_{k_1,k_2,\ldots,k_L,0,\ldots,0} \prod_{i=1}^{L} (T_i(x) - a_i)^{k_i} \right]$$

Здесь нулевые значения индексов $k_{L+1}, \ldots, k_N$ соответствуют игнорированию лишних переменных. Таким образом, при добавлении нового алгоритма в ансамбль старые слагаемые линейной комбинации остаются неизменными, а новые слагаемые просто дополняют разложение. Такой подход обеспечивает стабильность модели и позволяет эффективно наращивать ансамбль без пересчета уже найденных коэффициентов.

Для упрощения вычислений и повышения устойчивости модели будем нормализовать целевую переменную с помощью z-нормализации. Тогда центр разложения можно выбрать равным нулю: $(a_1, a_2, \ldots, a_N) = \overline{0}$, а в качестве M взять небольшое число.

Везде далее для упрощения восприятия под F_N будет подразумеваться $\widehat{F}_N$.

3.4 Стратегии обучения нового алгоритма

Введём обозначение $F_N(x) = \sum_{k=1}^{M} \overbrace{\sum_{k_1=0} \sum_{k_2=0} \cdots \sum_{k_N=0}}^{k_1+k_2+\cdots+k_N=k} b_{k_1,k_2,\ldots,k_N} T_1^{k_1}(x) T_2^{k_2}(x) \ldots T_N^{k_N}(x)$. Для повышения точности приближения при добавлении нового алгоритма $T_{N+1}(x)$ вводится параметр скорости обучения (learning rate), который вычисляется как $\alpha = \frac{\delta}{N^\theta}$, где N – порядковый номер добавляемого алгоритма в ансамбле, а δ и θ – гиперпараметры. Выбор вектора, на котором будет обучаться новый алгоритм, может осуществляться по одной из трёх стратегий:

1. Нахождение точного корня:

   При добавлении нового алгоритма $T_{N+1}(x)$ в ансамбль требуется, чтобы функция $F_{N+1}(x)$ аппроксимировала целевую переменную y для каждого объекта x из обучающей выборки (т.е. $F_{N+1}(x) \approx y$). Если подставить известные значения $T_1(x), T_2(x), \ldots, T_N(x)$ в выражение для $F_{N+1}(x)$, то задача сводится к решению полиномиального уравнения степени M относительно неизвестного $T_{N+1}(x)$. Для каждого объекта x_i из обучающей выборки необходимо решить уравнение вида $\sum_{k=0}^{M} C_k(x_i) \cdot z_i^k = y_i$, где $C_k(x_i)$ – функции, однозначно задаваемые $T_1(x_i), \ldots, T_N(x_i)$, а z_i представляет собой искомое значение $T_{N+1}(x_i)$. Однако данное уравнение не всегда имеет точное решение; в таких случаях в качестве z_i выбирается точка, в которой полином принимает значение, минимальное по модулю.

   После нахождения значений z_i для всех объектов обучается новый алгоритм T_{N+1} на векторе $\alpha \cdot (z_1, z_2, \ldots, z_N)$. Отметим, что первый алгоритм всегда обучается именно по этой стратегии.

2. Градиент:

   Пусть имеется функция потерь $\mathscr{L}(y, F(x))$, тогда можно записать, что $\mathscr{L}(y, F_{N+1}(x)) = \mathscr{L}(y, F_N(x) + \sum_{k=1}^{M} C_k(x) T_{N+1}^k(x))$, где $C_k(x)$ – функции, однозначно задаваемые $T_1(x), \ldots, T_N(x)$. Разложим $\mathscr{L}$ в ряд Тейлора в точке $T_{N+1}(x) = 0$ (для удобства переименуем $T_{N+1}(x)$ в z):

$$\mathscr{L}(y, F_N(x) + \sum_{k=1}^{M} C_k(x) z^k) = \mathscr{L}(y, F_N(x)) + z \cdot \left. \frac{\partial \mathscr{L}(y, F_N(x) + \sum_{k=1}^{M} C_k(x) z^k)}{\partial z} \right|_{z=0} =$$

$$= \mathscr{L}(y, F_N(x)) + z \cdot C_1(x) \cdot \left. \frac{\partial \mathscr{L}(y, F)}{\partial F} \right|_{F=F_N(x)}$$

где $C_1 = \sum_{k=1}^{M} \overbrace{\sum_{k_1=0} \sum_{k_2=0} \cdots \sum_{k_N=0}}^{k_1+k_2+\cdots+k_N=k-1} b_{k_1,k_2,\ldots,k_N} T_1^{k_1}(x) T_2^{k_2}(x) \ldots T_N^{k_N}(x)$

Возвращаясь к T_{N+1}:

$$T_{N+1}(x_i) = arg\min_T \mathscr{L}(y_i, F_N(x_i) + \sum_{k=1}^{M} C_k(x_i) T^k(x_i)) = arg\min_T \left(T(x_i) \cdot C_1(x_i) \cdot \left. \frac{\partial \mathscr{L}(y,F)}{\partial F} \right|_{F=F_N(x_i)} \right)$$

Отсюда получаем выражение для нового алгоритма $T_{N+1}(x_i) = -\alpha \cdot C_1(x_i) \cdot \left. \frac{\partial \mathscr{L}(y,F)}{\partial F} \right|_{F=F_N(x_i)}$,

где $\alpha \in (0, 1]$

Пример для MSE: $\mathscr{L}(y, F) = \frac{1}{2}(y - F(x))^2$

$$\frac{\partial \mathscr{L}(y,F)}{\partial F} = F(x) - y \Rightarrow T_{N+1}(x_i) = \alpha \cdot C_1(x_i) \cdot (y_i - F_N(x_i))$$

3.5 Стратегия оптимизации коэффициентов аггрегирующей функции

При фиксированном наборе данных $\mathscr{D}$ можно определить оператор $A_{\mathscr{D}}(b) : \mathbb{R}^P \to \mathbb{R}$, который для заданного вектора коэффициентов b выполняет процедуру кросс-валидации, на каждом разбиении обучая градиентный бустинг с указанными коэффициентами и возвращая среднее значение функции потерь, полученное по всем фолдам.

Предположим, что $A_{\mathscr{D}} \in C^1(\mathbb{R}^P)$. В таком случае, для вычисления частных производных можно применить следующий подход:

$$\left. \frac{\partial A_{\mathscr{D}}}{\partial b_i} \right|_{b=b_0} \approx \frac{A_{\mathscr{D}}(b_0 + \epsilon \cdot e_i) - A_{\mathscr{D}}(b_0)}{\epsilon}$$

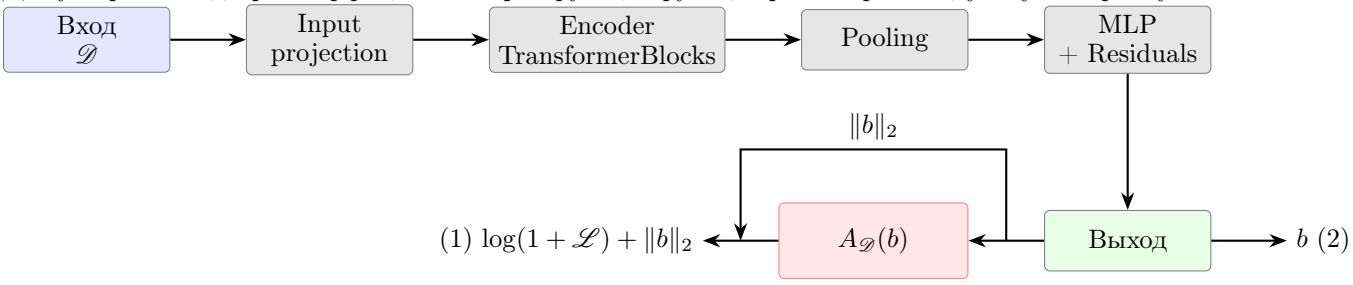
где e_i - это единичный вектор, направленный вдоль i-й координатной оси.

Таким образом, данный оператор может быть оптимизирован методом градиентного спуска:

$$b_i^{k+1} = b_i^k - \alpha \cdot \left. \frac{\partial A_{\mathscr{D}}}{\partial b_i} \right|_{b=b^k}$$

3.6 Мета-модель

Для ускорения подбора коэффициентов агрегирующей функции рассмотрим следующую нейронную сеть:



Процесс обучения модели осуществляется по траектории (1) с использованием алгоритма обратного распространения ошибки (метод вычисления градиента через оператор $A_{\mathscr{D}}(b)$ был представлен ранее). После завершения обучения модель функционирует по траектории (2) для генерации коэффициентов b, которые в дальнейшем используются для обучения градиентного бустинга.

3.7 Внешний критерий качества

В процессе обучения в качестве функции потерь используется MSE:

$$\mathscr{L}(y, F(x)) = \frac{1}{2}(y - F(x))^2$$

Минимизация данной функции потерь осуществляется на обучающей выборке и определяет процесс построения ансамбля.

Для внешней оценки качества модели на тестовой выборке ($\mathscr{D}$) используется $RMSE$, MAE и R^2:

$$Q_{RMSE} = \sqrt{\frac{1}{|\mathscr{D}|} \sum_{(x,y)\in\mathscr{D}} (y - F_N(x))^2}$$

$$Q_{MAE} = \frac{1}{|\mathscr{D}|} \sum_{(x,y)\in\mathscr{D}} |y - F_N(x)|$$

$$Q_{R^2} = 1 - \frac{\sum_{(x,y)\in\mathscr{D}} (y - F_N(x))^2}{\sum_{(x,y)\in\mathscr{D}} (y - \overline{y})^2}$$

Цель оптимизации заключается в снижении значения функции потерь $\mathscr{L}$ на этапе обучения, что, в свою очередь, позволяет достичь меньших значений внешнего критерия Q при том же числе базовых моделей.

Снижение внутреннего лосса означает, что ансамбль достигает заданного уровня точности при меньшем числе деревьев, сохраняя качество предсказаний на уровне стандартного градиентного бустинга.

Иными словами, если для классического градиентного бустинга с фиксированным числом деревьев N_0 достигается ошибка Q_0, то предлагаемая модель позволяет получить $Q < Q_0$ при том же N_0. Это, в свою очередь, даёт возможность уменьшить число деревьев до $N < N_0$, сохранив ошибку на уровне $Q \approx Q_0$.

3.8 Оптимизационная задача

Оптимизация параметров $b = \{b_{k_1,\ldots,k_N}\}$ в задаче $A_{\mathscr{D}}(b) \to \min\limits_{b}$ осуществляется с помощью градиентного спуска с моментом:

$$\begin{cases} v^{(k+1)} = \mu v^{(k)} - \eta \nabla_b A_{\mathscr{D}}(b^{(k)}) \\ b^{(k+1)} = b^{(k)} + v^{(k+1)} \end{cases}$$

где $\eta > 0$ — шаг обучения, $\mu \in [0, 1)$ — коэффициент момента, $v^{(k)}$ — накопленный импульс изменения параметров.

Обучение мета-модели направлено на минимизацию функционала:

$$\mathscr{J}(\phi) = \log\Big(1 + A_{\mathscr{D}}(\Psi_\phi(\mathscr{D}))\Big) + \|\Psi_\phi(\mathscr{D})\|_2$$

где Ψ_ϕ — нейронная сеть с параметрами ϕ, формирующая вектор коэффициентов $b = \Psi_\phi(\mathscr{D})$, используемых при построении ансамбля. Оператор $A_{\mathscr{D}}(b)$ для фиксированной выборки $\mathscr{D}$ возвращает значение функции потерь, соответствующее обученному ансамблю с коэффициентами b.

Таким образом, задача оптимизации параметров мета-модели может быть записана в виде:

$$\min_{\phi} \ \mathscr{J}(\phi) = \min_{\phi} \Big[\log\Big(1 + A_{\mathscr{D}}(\Psi_\phi(\mathscr{D}))\Big) + \|\Psi_\phi(\mathscr{D})\|_2 \Big]$$

Минимизация функционала $\mathscr{J}(\phi)$ осуществляется с использованием адаптивного алгоритма Adam. Алгоритм обеспечивает устойчивую и быструю сходимость при оптимизации нейронных сетей, что особенно важно при вычислении производных через оператор $A_{\mathscr{D}}(b)$.

На практике оптимизация выполняется итеративно: для каждой задачи из множества $\{\mathscr{D}_i\}$ вычисляется значение функционала $\mathscr{J}(\phi)$, выполняется обратное распространение градиента и обновление параметров ϕ оптимизатором Adam.

4 Эксперименты

Для оценки эффективности предложенного подхода были проведены эксперименты на множестве тестовых задач различной размерности и сложности.

В качестве базовой модели использовался стандартный градиентный бустинг. Так как в существующей литературе отсутствуют работы, направленные на оптимизацию агрегирующей функции в градиентном бустинге, данный вариант служит естественной точкой сравнения.

Кроме того, рассматривалась модификация FGBReg (Functional Gradient Boosting Regressor) — базовый вариант предлагаемой модели, в котором агрегирующая функция представлена в виде усечённого ряда Тейлора. При этом коэффициенты задаются по правилу $b_{k_1,\dots,k_N} = \frac{1}{\sum_{i=1}^{N} k_i}$. Данная версия служит контрольной, позволяя оценить влияние процедуры оптимизации коэффициентов на итоговое качество модели.

Для проверки эффективности оптимизации агрегирующей функции рассматривались две стратегии обучения коэффициентов b:

- FGBReg + GD. Для каждой задачи из тестового пула исходная выборка делилась на две равные части. На первой половине данных вектор коэффициентов b оптимизировался с помощью градиентного спуска, после чего на второй половине вычислялась тестовая ошибка с использованием кросс-валидации.
- FGBReg + NN. Для ускорения и обобщения процесса подбора коэффициентов b была обучена нейронная сеть, описанная ранее. Обучение мета-модели выполнялось на пуле из 64 различных задач. После завершения обучения качество модели оценивалось на тех же 9 тестовых наборах данных с использованием кросс-валидации.

Результаты экспериментов представлены в Таблицах 1–3. Для каждого набора данных приведены значения метрик $RMSE$, MAE и R^2 для стандартного градиентного бустинга (GB), предложенной модели с фиксированными коэффициентами (FGBReg), а также для вариантов с оптимизацией b с помощью градиентного спуска (FGBReg + GD) и нейронной сети (FGBReg + NN). Во всех экспериментах число деревьев в ансамбле фиксировалось на уровне $N = 20$.

Набор данных	GB	FGBReg	FGBReg + GD	FGBReg + NN
task1	93.69	95.72	86.78	99.43
task6	123.8	119.7	107.4	132.8
task7	8.424	8.380	8.366	8.378
task10	0.177	0.178	0.233	0.161
task16	41.03	33.54	32.05	35.99
task18	68.85	77.60	74.45	81.50
task41	0.401	0.364	0.356	0.345
task901	309.0	361.7	367.1	655.7
task970	40688	46121	45923	51427

Таблица 1: $RMSE$

Набор данных	GB	FGBReg	FGBReg + GD	FGBReg + NN
task1	67.09	72.21	62.40	79.40
task6	78.33	89.09	76.50	107.2
task7	6.857	6.788	6.786	6.816
task10	0.165	0.166	0.222	0.151
task16	32.83	25.95	24.84	28.70
task18	51.55	57.69	53.00	64.39
task41	0.211	0.194	0.197	0.202
task901	189.1	244.5	249.6	424.6
task970	28994	32799	32810	34680

Таблица 2: MAE

Набор данных	GB	FGBReg	FGBReg + GD	FGBReg + NN
task1	0.704	0.711	0.755	0.695
task6	0.654	0.723	0.762	0.661
task7	0.319	0.335	0.336	0.340
task10	-0.863	0.207	-0.108	0.303
task16	-11.28	-10.42	-11.61	-13.33
task18	0.681	0.642	0.603	0.617
task41	-0.503	-0.199	-0.133	-0.057
task901	0.919	0.894	0.891	0.716
task970	0.728	0.695	0.694	0.634

Таблица 3: R^2

Из представленных таблиц видно, что оптимизация коэффициентов агрегирующей функции с помощью градиентного спуска во многих задачах приводит к снижению ошибок $RMSE$ и MAE, а также к увеличению значения R^2 по сравнению с моделью с фиксированными коэффициентами. На части датасетов нейронная мета-модель также показывает улучшение качества, хотя её результаты характеризуются большей вариативностью.

При этом в отдельных задачах базовый градиентный бустинг остаётся лучшей моделью, что подчёркивает зависимость эффективности оптимизации коэффициентов от свойств конкретного набора данных. Тем не менее, общие результаты демонстрируют, что адаптивная настройка агрегирующей функции позволяет улучшать качество ансамбля при фиксированном числе деревьев и предоставляет дополнительный механизм контроля сложности модели.

Таким образом, предложенные методы демонстрируют потенциал к уменьшению числа базовых моделей в ансамбле при сохранении или улучшении точности по сравнению со стандартным градиентным бустингом.

5 Выводы

В работе был предложен обобщённый подход к построению ансамблей на основе градиентного бустинга, в котором классическая сумма базовых моделей заменяется параметризованной агрегирующей функцией в виде усечённого разложения в ряд Тейлора. Такой подход позволяет гибко моделировать взаимодействие между базовыми алгоритмами и, как следствие, повышать качество аппроксимации при фиксированном числе деревьев в ансамбле.

Результаты экспериментов показали, что предложенный подход способен снижать значение ошибки по сравнению с классическим градиентным бустингом. Особенно заметное улучшение наблюдается для варианта с оптимизацией коэффициентов методом градиентного спуска, что подтверждает целесообразность замены фиксированных коэффициентов на адаптивные. Использование нейронной мета-модели также демонстрирует потенциал, однако требует дополнительной настройки архитектуры и процедуры обучения для повышения устойчивости результатов.

Таким образом, предложенная концепция функционального градиентного бустинга открывает перспективы для дальнейшего развития ансамблевых методов, в частности — в направлении автоматического выбора агрегирующих функций и переноса знаний между задачами.

Список литературы

Justin Domke. Generic methods for optimization-based modeling. Journal of Machine Learning Research, 13: 203–233, 2012.

Luca Franceschi, Michele Donini, Paolo Frasconi, and Massimiliano Pontil. Forward and reverse gradient-based hyperparameter optimization. In Proceedings of the 34th International Conference on Machine Learning, 2017.

Dougal Maclaurin, David Duvenaud, and Ryan P. Adams. Gradient-based hyperparameter optimization through reversible learning. In Proceedings of the 32nd International Conference on Machine Learning, 2015.

Jonathan Lorraine, Paul Vicol, and David Duvenaud. Optimizing millions of hyperparameters by implicit differentiation. In Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics, 2019.

Amirreza Shaban, Ching-An Cheng, Nolan Hatch, and Byron Boots. Truncated back-propagation for bilevel optimization. In Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics, 2019.

Leo Breiman. Bagging predictors. Machine Learning, 24(2):123–140, 1996.

Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. Journal of Computer and System Sciences, 55(1):119–139, 1997.

Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. The Annals of Statistics, 29(5):1189–1232, 2001.

Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 785–794, 2016.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. In Advances in Neural Information Processing Systems 30, 2017.

Tong Zhang and Bin Yu. Boosting with early stopping: Convergence and consistency. The Annals of Statistics, 33(4):1538–1579, 2005.

Peter Bühlmann and Torsten Hothorn. Boosting algorithms: Regularization, prediction and model fitting. Statistical Science, 22(4):477–505, 2007.

Llew Mason, Jonathan Baxter, Peter Bartlett, and Marcus Frean. Boosting algorithms as gradient descent. In Advances in Neural Information Processing Systems 12, 2000.

Jerome H. Friedman. Stochastic gradient boosting. Computational Statistics & Data Analysis, 38(4):367–378, 2002.

Dragos D. Margineantu and Thomas G. Dietterich. Pruning adaptive boosting. In Proceedings of the 14th International Conference on Machine Learning, pages 211–218, 1997.

Rich Caruana, Alexandru Niculescu-Mizil, Geoff Crew, and Alex Ksikes. Ensemble selection from libraries of models. In Proceedings of the 21st International Conference on Machine Learning, 2004.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer, 2009.

David H. Wolpert. Stacked generalization. Neural Networks, 5(2):241–259, 1992.

K. M. Ting and I. H. Witten. Issues in stacked generalization. Journal of Artificial Intelligence Research, 10: 271–289, 1999.

R. Maclin and D. Opitz. Popular ensemble methods: An empirical study. Journal of Artificial Intelligence Research, 11:169–198, 1999.

Fabian Pedregosa. Hyperparameter optimization with approximate gradient. In Proceedings of the 33rd International Conference on Machine Learning, 2016.

Atilim Gunes Baydin, Barak A. Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. Automatic differentiation in machine learning: A survey. Journal of Machine Learning Research, 18:1–43, 2018.

Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. Practical bayesian optimization of machine learning algorithms. In Advances in Neural Information Processing Systems, 2012.

James Bergstra, Rémi Bardenet, Balázs Kégl, and Yoshua Bengio. Algorithms for hyper-parameter optimization. In Advances in Neural Information Processing Systems, 2011.