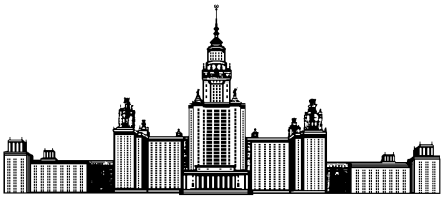


Московский государственный университет имени М. В. Ломоносова
Факультет Вычислительной математики и Кибернетики
Кафедра Математических Методов Прогнозирования

Таскынов Ануар Гульденбекович

Оптимизация параметров решающих деревьев с линейными разделяющими правилами в алгоритме бустинга

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

Научный руководитель:
к.ф.-м.н.
В. В. Китов

Москва, 2017

Содержание

1 Введение

На сегодняшний день редко используются одиночные методы классификации и регрессии и всё большую популярность получают методы, объединяющие несколько алгоритмов. Объединение алгоритмов в так называемый ансамбль даёт большую точность, чем его составляющие, так как ошибки различных базовых алгоритмов будут взаимно компенсироваться.

Одним из популярнейших методов составления ансамблей является бустинг. В бустинге базовые модели строятся последовательно, причём каждая следующая модель пытается исправить ошибки предыдущих моделей. Прогноз бустинга строится с помощью суммы базовых моделей. В качестве базовых моделей обычно используют неглубокие решающие деревья.

Недостатком стандартных решающих деревьев является то, что они рассматривают разбиения параллельно одной из осей координат, так как в каждом узле происходит проверка условия: больше или меньше заданный признак определенного порогового значения. Например (на Рис. 1), если реальная граница между классами линейная, то придется сделать достаточно большое количество разбиений, чтобы построить хорошую модель. В этом случае помогают деревья с линейными разбиениями общего вида, то есть в каждом узле проверяется условие $\langle \mathbf{w}, \mathbf{x} \rangle < w_0$. С одной стороны, это снизит количество разбиений и глубину дерева и позволит более гибко описывать классы объектов. С другой стороны, гибкость вносит большой вклад в переобучение.

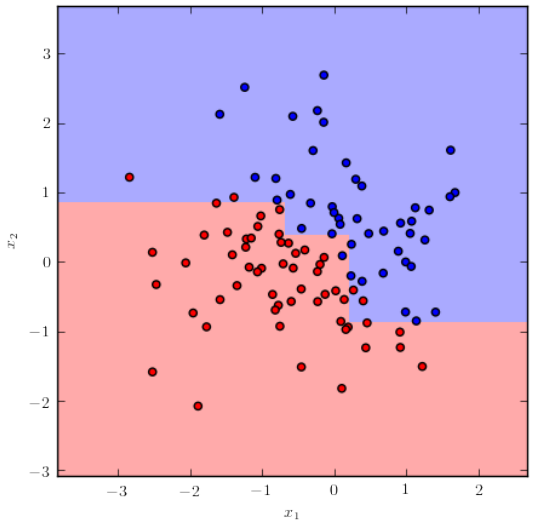
В данной работе будет изучено, какое влияние окажет такой вид деревьев в контексте бустинга, а именно улучшится ли прогноз по сравнению с обычными деревьями. Будет предложено четыре вида построения деревьев с линейными разбиениями:

- RidgeCART – в узле дерева решается задача Ridge-регрессии.
- Continuously Optimized Oblique Tree (CO2) – недифференцируемая функция потерь, которая оптимизируется в обычных деревьях, заменяется на дифференцируемую верхнюю оценку.
- Householder CART (HHCART) – CART с преобразованием Хаусхолдера.
- Random CART – CART со случайными поворотами.

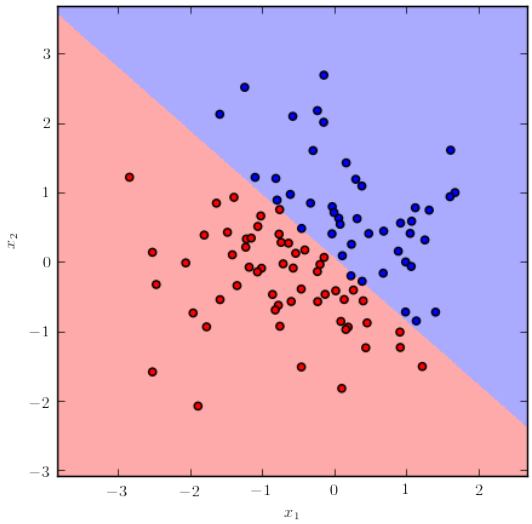
Также будет рассмотрено их применение к градиентному бустингу и будут проведены эксперименты, которые покажут улучшится ли качество прогноза.

1.1 Определения и обозначения

Пусть интересующий объект $\mathbf{x}$ описывается d признаками, тогда его можно представить в виде $\mathbf{x} = (x_1, \ldots, x_d)^T$. Пусть $\{\mathbf{x}_1, \ldots, \mathbf{x}_l\}$ - выборка из l объектов, где $\mathbf{x}_i \in \mathbb{R}^d$, тогда эту выборку можно представить в виде матрицы «объект-признак» $\mathbf{X} \in \mathbb{R}^{l \times d}$. Пусть $\mathbf{y} = (y_1, \ldots, y_l)^T$ - вектор ответов на данной выборке.



а) Обычное решающее дерево, глубина дерева – 3.



б) Дерево с линейным разбиением: глубина дерева – 1.

Рис. 1: Разделяющая кривая для разных видов деревьев.

Базовые алгоритмы будут обозначаться как $b(\mathbf{x})$, а алгоритм ансамбля, состоящего из N базовых алгоритмов, $a_N(\mathbf{x})$. В дальнейшем будут рассматриваться задачи бинарной классификации для бустинга, то есть $y_i \in \{-1, 1\}$ и задача регрессии $y_i \in \mathbb{R}$ для базовых алгоритмов.

Также будет применяться следующая запись $\hat{\mathbf{x}}$, которое означает, что исходному объекту был добавлен константный признак. Вектор весов $\hat{\mathbf{w}}$ будет содержать не только веса для признаков, но и сдвиг w_0.

1.2 Градиентный бустинг

Градиентный бустинг - это ансамблевый алгоритм, прогноз которого строится на основании взвешенного голосования базовых алгоритмов:

$$a_N(\mathbf{x}) = \sum_{n=1}^{N} \gamma_n b_n(\mathbf{x}).$$

Алгоритм построения композиции является жадным. Допустим, что построена композиция $a_{N-1}(\mathbf{x})$ из $N-1$ алгоритма, и далее выбирается следующий базовый алгоритм $b_N(\mathbf{x})$ таким образом, чтобы как можно сильнее уменьшить ошибку:

$$\sum_{i=1}^{l} L(y_i, a_{N-1}(\mathbf{x}_i) + \gamma_N b_N(\mathbf{x}_i)) \to \min_{\gamma_N, b_N(\mathbf{x})},$$

где $L(y, z)$ - некоторая дифференцируемая функция потерь, которая выбирается в зависимости от типа задачи. Для решения этой задачи предлагается использовать один шаг градиентного спуска в l-мерном пространстве:

$$s_i = -\frac{\partial L(y,z)}{\partial z}\Big|_{z=a_{N-1}(\mathbf{x}_i)},$$

$$\Big(-\frac{\partial L(y,z)}{\partial z}\Big|_{z=a_{N-1}(\mathbf{x}_i)}\Big)_{i=1}^{l} = -\nabla_s \sum_{i=1}^{l} L(y_i, a_{N-1}(\mathbf{x}_i) + s_i).$$

Дальше алгоритм $b_N(\mathbf{x})$ ищется в так называемом семействе базовых алгоритмов $\mathcal{B}$, с помощью метода наименьших квадратов:

$$b_N(\mathbf{x}) = \arg\min_{b(\mathbf{x})\in\mathcal{B}} \sum_{i=1}^{l} \big(s_i - b(\mathbf{x}_i)\big)^2.$$

После того, как базовый алгоритм найден, можно подобрать коэффициент γ_N при нем по аналогии с наискорейшим градиентным спуском:

$$\gamma_N = \arg\min_{\gamma\in\mathbb{R}} \sum_{i=1}^{l} L(y_i, a_{N-1}(\mathbf{x}_i) + \gamma b_N(\mathbf{x}_i)).$$

На практике градиентный бустинг довольно быстро строит композицию и далее начинает переобучаться, настраиваясь на шум. Для того, чтобы избежать этого используют сокращение шага (shrinking):

$$a_N(\mathbf{x}) = a_{N-1}(\mathbf{x}) + \eta\gamma_N b_N(\mathbf{x}),$$

где $\eta \in (0, 1]$.

2 Методы, находящие коэффициенты для линейного разбиения

В этих методах вектор весов $\mathbf{w}$ будет найден сразу, в отличие от методов следующей главы, где используется линейное преобразование признаков.

2.1 RidgeCART

В основе этого метода стоит решение задачи линейной (а именно гребневой) регрессии. Модель линейной регрессии подразумевает, что ответ y может быть представлен в виде линейной комбинации признаков, то есть

$$y = \langle \hat{\mathbf{w}}, \hat{\mathbf{x}} \rangle^{1}.$$

Решением задачи гребневой регрессии является такой $\hat{\mathbf{w}}^*$, что при $\hat{\mathbf{w}}^*$ достигается минимум следующего функционала при $\hat{\mathbf{w}} \in \mathbb{R}^{d+1}$:

$$\mathcal{Q}(\hat{\mathbf{w}}) = ||\hat{\mathbf{X}}\hat{\mathbf{w}} - \mathbf{y}||^2 + \alpha||\hat{\mathbf{w}}||^2,$$

где $\alpha > 0$ - параметр регуляризации, штрафующий большие значения нормы вектора $||\hat{\mathbf{w}}||$. Функционал выпуклый, поэтому он имеет глобальную точку минимума. Продифференцировав по $\mathbf{w}$ и приравнивая градиент к 0, получаем:

$$\hat{\mathbf{w}}^* = (\hat{\mathbf{X}}^T\hat{\mathbf{X}} + \alpha\mathbf{I})^{-1}\hat{\mathbf{X}}^T\mathbf{y}.$$

В методе RidgeCART в каждом узле решается задача гребневой регрессии, то есть находится оптимальный вектор весов $\hat{\mathbf{w}}^*$. Отметим, что вектор $\mathbf{w}^* = (w_1^*, \ldots, w_d^*)$ указывает на рост функционала $J(\mathbf{x}) = \langle \mathbf{w}^*, \mathbf{x} \rangle$ и учитывая, что $\hat{\mathbf{w}}^*$ - это решение линейной регрессии, то чем больше $J(\mathbf{x})$, тем больше ответ y на этом объекте. Таким образом можно использовать $\langle \mathbf{w}^*, \mathbf{x} \rangle$, как новый признак который является линейной комбинацией других признаков и предикат в узле $\langle \mathbf{w}^*, \mathbf{x} \rangle > t$ будет означать линейное разбиение.

Дальше можно рассмотреть две вариации метода:

- Метод Ridge CART(c)[2]. К исходным d признакам добавляется еще один $\langle \mathbf{w}^*, \mathbf{x} \rangle$. И в новых $d+1$ признаках ищется наилучшее разбиение.
- Метод Ridge CART. Вместо всех d признаков используется только один признак $\langle \mathbf{w}^*, \mathbf{x} \rangle$.

Параметром этого алгоритма является α – параметр регуляризации для гребневой регрессии. В данном методе параметр регуляризации будет единым для всего дерева и будет подбираться по кросс-валидации.

2.2 Continuously Optimized Oblique Tree (CO2)

Введём функционал, который будет оптимизироваться в узле дерева, следующим образом:

$$\mathcal{L}(\hat{\mathbf{w}}, \theta_{0,1}; (\hat{\mathbf{X}}, \mathbf{y})) = \sum_{i=1}^{l} \Big(\big[\hat{\mathbf{w}}^T\hat{\mathbf{x}}_i < 0\big] l(\theta_0, y_i) + \big[\hat{\mathbf{w}}^T\hat{\mathbf{x}}_i \geq 0\big] l(\theta_1, y_i) \Big), \tag{1}$$

[1]Здесь предполагается, что в $\hat{\mathbf{x}}$ добавлен константный признак. То есть $\hat{\mathbf{x}} = (1, x_1, \ldots, x_d) \in \mathbb{R}^{d+1}$, соответственно $\hat{\mathbf{w}} = (w_0, w_1, \ldots, w_d)$.

[2]Здесь имеется (c) обозначает compared with CART.

где $\theta_{0,1}$ – среднее значение ответа на левом и правом поддереве соответственно, а $l(\theta, y) = (\theta - y)^2$. Для того чтобы разбиение было оптимально нужно проминимизировать функционал $\mathcal{L}(\hat{\mathbf{w}}, \theta_{0,1}; (\hat{\mathbf{X}}, \mathbf{y}))$ по $\theta_{0,1}$ и $\hat{\mathbf{w}}$. Заметим, что функционал является разрывной функцией относительно $\hat{\mathbf{w}}$, поэтому поиск оптимального значения затруднителен.

В статье [6] была предложена идея использовать верхнюю гладкую оценку на слагаемые (1):

$$\big[\hat{\mathbf{w}}^T\hat{\mathbf{x}}_i < 0\big]l(\theta_0, y_i) + \big[\hat{\mathbf{w}}^T\hat{\mathbf{x}}_i \geq 0\big]l(\theta_1, y_i) \leq \max\Big(-\hat{\mathbf{w}}^T\hat{\mathbf{x}}_i + l(\theta_0, y_i), \hat{\mathbf{w}}^T\hat{\mathbf{x}}_i + l(\theta_1, y_i)\Big) - |\hat{\mathbf{w}}^T\hat{\mathbf{x}}_i|. \tag{2}$$

Верхняя оценка является непрерывной, поэтому ее легче оптимизировать. Для того чтобы доказать эту оценку достаточно рассмотреть два случая:

1. Если $\hat{\mathbf{w}}^T\hat{\mathbf{x}}_i < 0$, то:
$$l(\theta_0, y_i) \leq \max\Big(l(\theta_0, y_i), 2\hat{\mathbf{w}}^T\hat{\mathbf{x}}_i + l(\theta_1, y)\Big).$$
2. Если $\hat{\mathbf{w}}^T\hat{\mathbf{x}}_i \geq 0$, то:
$$l(\theta_1, y_i) \leq \max\Big(-\hat{\mathbf{w}}^T\hat{\mathbf{x}}_i + l(\theta_0, y_i), l(\theta_1, y_i)\Big).$$

Заметим, что исходный функционал никак не зависел от нормировки $\hat{\mathbf{w}}$, то есть $\mathcal{L}(\hat{\mathbf{w}}, \theta_{0,1}) = \mathcal{L}(a\hat{\mathbf{w}}, \theta_{0,1})$. Верхняя оценка, напротив, при уменьшении нормы вектора весов становиться всё больше. Докажем это в следующем неравенстве:

$$\begin{aligned} &\max\Big(\hat{\mathbf{w}}^T\hat{\mathbf{x}}_i + l(\theta_0, y_i), \hat{\mathbf{w}}^T\hat{\mathbf{x}}_i + l(\theta_1, y_i)\Big) - |\hat{\mathbf{w}}^T\hat{\mathbf{x}}_i| \geq \\ &\geq \max\Big(-a\hat{\mathbf{w}}^T\hat{\mathbf{x}}_i + l(\theta_0, y), a\hat{\mathbf{w}}^T\hat{\mathbf{x}}_i + l(\theta_1, y)\Big) - a|\hat{\mathbf{w}}^T\hat{\mathbf{x}}_i|, \end{aligned} \tag{3}$$

при $a > 1$. Для доказательства этого неравенства также можно рассмотреть два случая:

1. Когда $\hat{\mathbf{w}}^T\hat{\mathbf{x}}_i < 0$, то неравенство можно переписать в следующем виде:
$$\max\Big(l(\theta_0, y_i), 2\hat{\mathbf{w}}^T\hat{\mathbf{x}}_i + l(\theta_1, y_i)\Big) \geq \max\Big(l(\theta_0, y_i), 2a\hat{\mathbf{w}}^T\hat{\mathbf{x}}_i + l(\theta_1, y_i)\Big),$$
что является очевидным.
2. Когда $\hat{\mathbf{w}}^T\hat{\mathbf{x}}_i \geq 0$, то:
$$\max\Big(-2\hat{\mathbf{w}}^T\hat{\mathbf{x}}_i + l(\theta_0, y_i), l(\theta_1, y_i)\Big) \geq \max\Big(-2a\hat{\mathbf{w}}^T\hat{\mathbf{x}}_i + l(\theta_0, y_i), l(\theta_1, y_i)\Big),$$
что также очевидно.

Это можно проинтерпретировать следующим образом: чем больше норма весов, тем верхняя оценка становится ближе к исходному функционалу и тем сложнее его оптимизировать. В статье предлагается ввести ограничения на $\hat{\mathbf{w}}$, чтобы верхняя оценка не слишком сильно отличалась от $\mathcal{L}(\hat{\mathbf{w}}, \theta_{0,1})$. Тем самым предлагается решить следующую задачу условной оптимизации:

$$\begin{cases} \mathcal{L}'(\hat{\mathbf{w}}, \theta_{0,1}) \to \min\limits_{\hat{\mathbf{w}}, \theta_{0,1}} \\ ||\hat{\mathbf{w}}||^2 \leq \nu \end{cases}, \tag{4}$$

где $\nu \in \mathbb{R}^+$, а $\mathcal{L}'(\hat{\mathbf{w}}, \theta_{0,1})$ это верхняя оценка, то есть:

$$\mathcal{L}'(\hat{\mathbf{w}}, \theta_{0,1}; (\hat{\mathbf{X}}, \mathbf{y})) = \sum_{i=1}^{l} \max\Big(-\hat{\mathbf{w}}^T\hat{\mathbf{x}} + l(\theta_0, y), \hat{\mathbf{w}}^T\hat{\mathbf{x}} + l(\theta_1, y)\Big) - |\hat{\mathbf{w}}^T\hat{\mathbf{x}}|. \tag{5}$$

Algorithm 1 ***Построение узла CO2.***

Вход: $(\mathbf{X}^t, \mathbf{y}^t)$ – объекты, попавшие в данный узел t, ν – параметр регуляризации, τ – количество итераций в методе субградиентного стохастического спуска, η – шаг градиентного спуска.

Выход: Узел дерева CO2.

Инициализировать $\hat{\mathbf{w}}$ обычным разбиением дерева;
Инициализировать θ_0, θ_1 на основе разбиения $\hat{\mathbf{w}}$ и $(\mathbf{X}^t, \mathbf{y}^t)$;
пока не сошелся функционал 5
    $\hat{\mathbf{w}}^{old} := \hat{\mathbf{w}}$;
    для $t = 1, \ldots, \tau$:
        Сэмплировать $(\hat{\mathbf{x}}, y)$ из $(\hat{\mathbf{X}}^t, \mathbf{y}^t)$;
        $s := \text{sgn}\big(\langle \hat{\mathbf{w}}^{old}, \hat{\mathbf{x}} \rangle\big)$;
        если $-\hat{\mathbf{w}}^T\hat{\mathbf{x}} + l(\theta_0, y) \geq \hat{\mathbf{w}}^T\hat{\mathbf{x}} + l(\theta_1, y)$ **то**
            $\hat{\mathbf{w}} := \hat{\mathbf{w}} + \eta(1 + s)\hat{\mathbf{x}}$;
            $\theta_0 := \theta_0 - \eta\frac{\partial l(\theta_0, y)}{\partial \theta}$;
        иначе
            $\hat{\mathbf{w}} := \hat{\mathbf{w}} - \eta(1 - s)\hat{\mathbf{x}}$;
            $\theta_1 := \theta_1 - \eta\frac{\partial l(\theta_1, y)}{\partial \theta}$;
        если $||\hat{\mathbf{w}}||^2 > \nu$ **то**
            $\hat{\mathbf{w}} := \sqrt{\nu}\frac{\hat{\mathbf{w}}}{||\hat{\mathbf{w}}||}$
В соответствии с вектором весов $\hat{\mathbf{w}}$ вызвать этот метод для левого и правого поддеревьев.

Отметим, что (5) является выпукло-вогнутой функцией относительно $\hat{\mathbf{w}}$ (первое слагаемое является выпуклой частью, второе - вогнутой) и выпуклая относительно θ_0 и θ_1. Оптимизация такой задачи также затруднительна, однако функционал

уже можно дифференцировать (точнее, находить субдифференциал). Авторы статьи предложили следующую идею оптимизации (сам алгоритм приведен 1):

1. Фиксация субградиента вогнутого слагаемого, то есть $-|\hat{\mathbf{w}}^T\hat{\mathbf{x}}|$ при текущей оценке $\hat{\mathbf{w}}$. Пусть $\hat{\mathbf{w}}^{olf}$ - это приближение вектора весов с предыдущей итерации.

2. Субрадиентный спуск относительно выпуклой задачи оптимизации (в целях оптимизации времени использовался стохастический вариант):

$$\sum_{i=1}^{l} \max\Big(-\hat{\mathbf{w}}^T\hat{\mathbf{x}}_i + l(\theta_0, y_i), \hat{\mathbf{w}}^T\hat{\mathbf{x}}_i + l(\theta_1, y_i)\Big) - sgn\big(\hat{\mathbf{w}}^{old}\hat{\mathbf{x}}_i\big)\hat{\mathbf{w}}^T\hat{\mathbf{x}}_i \to \min_{\hat{\mathbf{w}},\theta_{0,1}}.$$

3. Если норма весов превысила ν, то перенормируем $\hat{\mathbf{w}}$.

Авторы статьи использовали метод CO2 в качестве базовых алгоритмов в случайном лесе. При этом $l(\theta, y)$ – это логарифмическая функция потерь. CO2 Forest сравнивался с Random Forest и показал значительно лучшее качество и для достижения этого качества требовалось меньшее количество деревьев и деревья были менее глубокими. В данной работе будет изучено влияние CO2 дерева на точность бустинга.

3 Методы, использующие переход в новое признаковое пространство

Рассмотрим теперь методы, которые будут «косвенно» находить линейные разбиения. Будем линейными преобразованиями изменять базис исходного пространства на другой. Для этого будут перебираться ортогональные матрицы: $\mathbf{Q}^T\mathbf{Q} = \mathbf{Q}\mathbf{Q}^T = \mathbf{I}$, $\mathbf{Q} \in \mathbb{R}^{d\times d}$.

Пусть в новом признаковом пространстве найдено оптимальное разбиение (i, t), где i – это номер признака, а t – порог. При выполненных условиях на матрицу $\mathbf{Q}$, найдем коэффициенты весов $\mathbf{w}$ в исходном пространстве. В новом пространстве признаков $\tilde{\mathbf{w}} = \big(0, \ldots, \underbrace{1}_{i}, \ldots, 0\big)^T \in \mathbb{R}^d$, $\tilde{\mathbf{x}} = \mathbf{Q}^T\mathbf{x}$, а решающее правило выглядит так: $\big\langle\tilde{\mathbf{w}}, \tilde{\mathbf{x}}\big\rangle \geq t$. Напомним, что уравнение гиперплоскости выглядит следующим образом: $\big\langle\mathbf{w}, \mathbf{x}\big\rangle = \big\langle\mathbf{w}, \mathbf{x}_0\big\rangle$, где $\mathbf{x}_0$ принадлежит гиперплоскости. В нашем случае $\tilde{\mathbf{x}}_0 = \big(0, \ldots, \underbrace{t}_{i}, \ldots, 0\big)^T$. Таким образом гиперплоскость в новом признаковом пространстве можно записать, как: $\big\langle\tilde{\mathbf{w}}, \tilde{\mathbf{x}}\big\rangle = \big\langle\tilde{\mathbf{w}}, \tilde{\mathbf{x}}_0\big\rangle$. Далее, $\mathbf{w} = \mathbf{Q}^{-1}\tilde{\mathbf{w}} = \mathbf{Q}^T\tilde{\mathbf{w}} = \mathbf{Q}_i$, также $\mathbf{x}_0 = t\mathbf{Q}_i$ где $\mathbf{Q}_i$ – i-я строка матрицы $\mathbf{Q}$. Уравнение гиперплоскости в исходном пространстве: $\big\langle\mathbf{w}, \mathbf{x}\big\rangle = \big\langle\mathbf{w}, \mathbf{x}_0\big\rangle = \big\langle\mathbf{Q}_i, t\mathbf{Q}_i\big\rangle = t$. Итого, исходное решающее правило с симметричной, ортогональной матрицей поворота $\mathbf{Q}$ выглядит следующим образом:

$$\big\langle\mathbf{w}, \mathbf{x}\big\rangle \geq t,$$

где $\mathbf{w} = \mathbf{Q}_i$. Такая оптимизация уменьшает затраты по памяти для хранения дерева: вместо матрицы размера $d \times d$ нужно хранить только i-ю строку матрицы.

3.1 Householder CART (HHCART)

Построение данного вида решающего дерева основано на преобразовании Хаусхолдера [2]. Относительно нормированного вектора $\mathbf{u} \in \mathbb{R}^d$ преобразование Хаусхолдера выглядит следующим образом:

$$\mathcal{H}(\mathbf{x}) = \mathbf{x} - 2\langle \mathbf{u}, \mathbf{x} \rangle \mathbf{u}.$$

Или в эквивалентной матричной формулировке: матрица Хаусхолдера имеет вид:

$$\mathcal{H}(\mathbf{x}) = \mathbf{H}\mathbf{x},$$

где $\mathbf{H} = \mathbf{I} - 2\mathbf{u}\mathbf{u}^T$, $\mathbf{I} \in \mathbb{R}^{d \times d}$ – единичная матрица. Одними из замечательных свойств матрицы является ее ортогональность и симметричность: $\mathbf{H}^T\mathbf{H} = \mathbf{I}$, $\mathbf{H} = \mathbf{H}^T$.

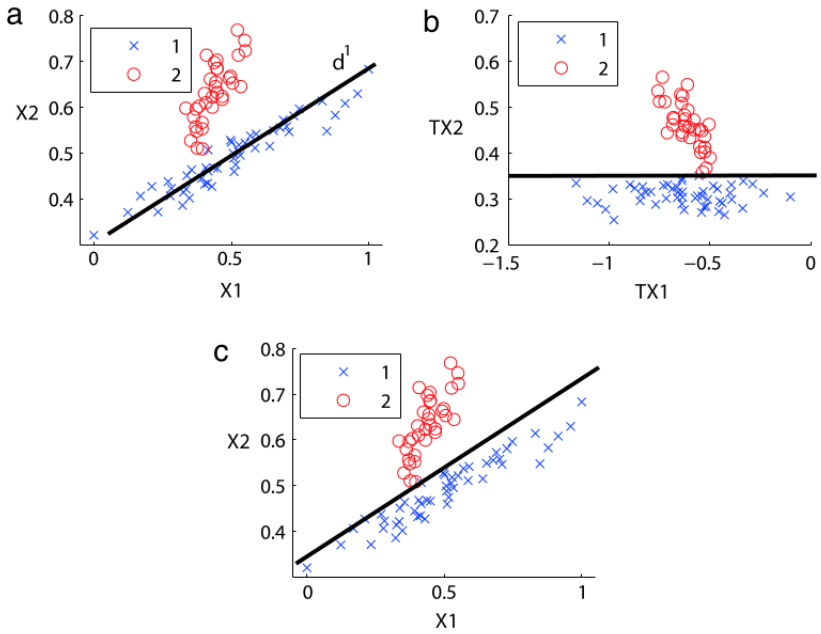


Рис. 2: Результат преобразования Хаусхолдера.

Преобразование Хаусхолдера имеет следующий геометрический смысл: это отражение точки $\mathbf{x}$ относительно гиперплоскости, заданной вектором $\mathbf{u}$.

Пусть $\mathbf{X}^c$ – это объекты класса c, $\mathbf{\Sigma}^c = \frac{1}{l^c - 1}(\mathbf{X} - \boldsymbol{\mu}^c)^T(\mathbf{X} - \boldsymbol{\mu}^c)$, $\mathbf{\Sigma}^c \in \mathbb{R}^{d \times d}$ – матрица ковариации для объектов класса c, l^c – количество объектов класса c, $\boldsymbol{\mu}_c \in \mathbb{R}^d$ – вектор среднего значения класса c.

Рассмотрим, как происходит разбиение в узле. В алгоритме за основу берется лучшее разбиение, которое параллельно осям признакового пространства. Далее авторы предложили два варианта:

- Метод HHCART(D). Для матрицы ковариации $\mathbf{\Sigma}^c$ ищется собственный вектор $\boldsymbol{\xi}_c$, отвечающий наибольшему собственному значению λ_c. Если одновременно

для всех $i = 1, \ldots, d$: $||\mathbf{e}_i - \boldsymbol{\xi}_c|| > \tau$, где $\mathbf{e}_i = (0, \ldots, \underbrace{1}_{i}, \ldots, 0)^T \in \mathbb{R}^d$, то строится матрица Хаусхолдера с $\mathbf{u} = \frac{\mathbf{e}_i - \boldsymbol{\xi}_c}{||\mathbf{e}_i - \boldsymbol{\xi}_c||}$. Все объекты переводятся в новое признаковое пространство $\boldsymbol{\mathcal{X}}^c = \mathbf{X}^c\mathbf{H}$. Затем из всех классов c находится преобразование Хаусхолдера, которое наилучшим образом разделяет выборку, попавшую в узел. Это разбиение сравнивается с обычным разбиением и из них выбирается наилучшее.

- Метод HHCART(A). Для матрицы ковариации $\boldsymbol{\Sigma}^c$ ищутся все собственные вектора $\left((\boldsymbol{\xi}^{1c}, \lambda^{1c}), \ldots, (\boldsymbol{\xi}^{dc}, \lambda^{dc})\right)$. Затем для каждого из собственных векторов $\boldsymbol{\xi}^{jc}$, где $j : \lambda^{jc} \neq 0$, строится преобразование способом, описанным выше и из всех таких j и классов c ищется наилучшее разбиение в преобразованном пространстве признаков, результат сохраняется и сравнивается с обычным разбиением и из них выбирается лучшее.

Преобразование в методе HHCART имеет геометрическую интерпретацию: это отражение собственного вектора $\boldsymbol{\xi}^c$ в вектор $\mathbf{e}_i$. Можно доказать этот факт геометрически: так как два вектора $\boldsymbol{\xi}^c$ и $\mathbf{e}_i$ нормированы, то в проекции на плоскость, задаваемой этими двумя векторами начало координат и концы этих векторов будут задавать равнобедренный треугольник. Проекция гиперплоскости, задаваемой вектором $\frac{\mathbf{e}_i - \boldsymbol{\xi}^c}{||\mathbf{e}_i - \boldsymbol{\xi}^c||}$ будет являться перпендикуляром к основанию треугольника. Таким образом данные два вектора будут отражаться друг в друга. На Рис. 2 показан результат действия матрицы Хаусхолдера (изображение взято со статьи [1]).

По результатам экспериментов статьи [1], HHCART(A) и HHCART(D) работают примерно одинаково. В статье был описан метод для классификации и его никак нельзя применить для задач регрессии.

Рассмотрим, как можно применить данный алгоритм для регрессии. Для этого необходимо решить задачу гребневой регрессии:

$$||\hat{\mathbf{X}}\hat{\boldsymbol{\beta}} - \mathbf{y}||^2 + \alpha||\hat{\boldsymbol{\beta}}||^2 \to \min_{\hat{\boldsymbol{\beta}} \in \mathbb{R}^{d+1}}.$$

Как было показано ранее решением является $\hat{\boldsymbol{\beta}}^* = (\hat{\mathbf{X}}^T\hat{\mathbf{X}} + \alpha\mathbf{I})^{-1}\hat{\mathbf{X}}^T\mathbf{y}$. Далее применить преобразование Хаусхолдера с $\mathbf{u} = \frac{\mathbf{e}_i - \boldsymbol{\beta}^*}{||\mathbf{e}_i - \boldsymbol{\beta}^*||}$, после чего находим оптимальное i как в методе HHCART(D). Отметим, что $\boldsymbol{\beta}^*$ – направление роста функции $J(\mathbf{x}) =< \boldsymbol{\beta}^*, \mathbf{x} >$, так как кроме этого $\boldsymbol{\beta}^*$ – является решением задачи регрессии, то он показывает на рост ответов $\mathbf{y}$.

Отметим, что при этом появляется параметр регуляризации α, который будет встречаться на всех узлах при построении дерева. Для того чтобы не усложнять модель принимаем, что α будет единым для всего дерева и этот параметр будет настраиваться кросс-валидацией.

Algorithm 2 ***Построение узла HHCART для задачи регрессии.***

Вход: $(\mathbf{X}_t, \mathbf{y}_t)$ – объекты, попавшие в данный узел t; $\tau > 0$ – положительная константа; $\alpha > 0$ – коэффициент регуляризации для гребневой регрессии.

Выход: Внутренний узел HHCART.

$impurity := 0$;
$h_t := \emptyset$;
$h_t^{best} :=$ обычное разбиение дерева;
$h_t := h_t^{best}$;
$impurity :=$ критерий информативности для h_t^{best};
Решить задачу гребневой регрессии: $\hat{\boldsymbol{\beta}}^* := (\hat{\mathbf{X}}_t^T \hat{\mathbf{X}}_t + \alpha \mathbf{I})^{-1} \hat{\mathbf{X}}_t^T \mathbf{y}_t$
если $||\mathbf{e}_1 - \boldsymbol{\beta}^*|| \leq \tau$ **или** $||\mathbf{e}_2 - \boldsymbol{\beta}^*|| \leq \tau$ **или** ... **или** $||\mathbf{e}_d - \boldsymbol{\beta}^*|| \leq \tau$ **то**
    $\mathbf{H}_t^{jc} := \mathbf{I}$;
иначе
    Построить матрицу Хаусхолдера $\mathbf{H}_t$ используя $\boldsymbol{\beta}^*$ и $\mathbf{e}_i$, где $i = \arg\max_j ||\mathbf{e}_j - \boldsymbol{\beta}^*||$;
$\mathcal{X}_t := \mathbf{X}_t \mathbf{H}_t$;
Найти наилучшее разбиение в новом пространстве, h_t';
если $impurity(h_t') > impurity$ **то**
    $h_t := h_t'$;
    $impurity := impurity(h_t')$;
В соответствии с разбиением h^t вызвать этот метод для правого и левого поддеревьев;

3.2 Random CART (RandCART)

В работах [3], [4] было показано применение случайных поворотов в ансамблях, где базовые алгоритмы обучались на $\mathcal{X} = \mathbf{XQ}$, где $\mathbf{Q}$ - случайная, ортогональная, симметричная матрица. Можно применить ту же идею для построения деревьев, где в каждом узле будет производится случайный поворот.

Генерация матрицы случайного поворота:

1. Генерируется матрица $\mathbf{A} \in \mathbb{R}^{d \times d}$ из нормального многомерного распределения $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ с нулевым средним $\boldsymbol{\mu}$ и единичной матрицей ковариации $\boldsymbol{\Sigma}$, то есть каждый столбец независимо сэмплируется из нормального распределения.

2. Над матрицей $\mathbf{A}$ производится QR-разложение. Напомним, что QR-разложение матрицы – это представление матрицы в виде ортогональной матрицы $\mathbf{Q} \in \mathbb{R}^{d \times d}$ и верхнетреугольной матрицы $\mathbf{R} \in \mathbb{R}^{d \times d}$. Матрица $\mathbf{Q}$ является искомой матрицей случайного поворота.

Можно рассмотреть две вариации метода:

1. Метод RandCART. Генерируется матрица поворота, происходит поворот посредством этой матрицы в новое признаковое пространство и на нем ищется наилучшее разбиение.

2. Метод RandCART(c). Ищется наилучшее разбиение в исходном признаковом пространстве, затем оно сравнивается с наилучшим разбиением, полученным матрицей поворота.

Ниже описано построение узла дерева RandCART(c).

Algorithm 3 ***Построение узла RandCART(c).***

Вход: $\mathbf{X}_t$ – объекты, попавшие в данный узел; $\mathbf{y}_t$ – ответы на этих объектах.
Выход: Обученное дерево с линейными разбиениями на узлах.

если листовая вершина **то**
  вернуть листовую вершину;
h_{best} := наилучшее разбиение, параллельное осям признакового пространства;
I^{best} := критерий информативности, отвечающий разбиению h_{best};
Генерация матрицы поворота $\mathbf{Q}$;
$\mathcal{X}_t := \mathbf{XQ}$;
h_t^{rand} := наилучшее разбиение в новом признаковом пространстве;
I^{rand} := критерий информативности, отвечающий разбиению h_{rand}
если $I^{rand} < I^{best}$ **то**
  $I^t := I^{rand}$;
  $h^t := h^{rand}$;
иначе
  $I^t := I^{best}$;
  $h^t := h^{best}$;
В соответствии с разбиением h^t вызвать этот метод для правого и левого поддеревьев;

4 Эксперименты на данных

4.1 Условия экспериментов.

Данные были загружены с репозитория UCI [5]. Собрано 12 задач бинарной классификации. Все категориальные признаки были преобразованы согласно one-hot-encoding, вещественные признаки отмасштабированы так, что они имеют нулевое среднее и единичное стандартное отклонение. Объекты, имеющие хотя бы один пропуск были удалены из выборки. Подробное описание в таблице 1. В таблице указано итоговое количество признаков.

Каждая выборка было поделена следующим образом: 75% - обучающая выборка, 25% - тестовая. На обучающей выборке была проведена кросс-валидация по глубине дерева в каждом алгоритме и по количеству самих деревьев. В кросс-валидации глубина дерева варьировалась от 1 до 7. Валидация по количеству деревьев проводилась следующим образом: если через k добавленных базовых алгоритмов $b_n(\mathbf{x})$ не изменяется качество на валидационном множестве, то выбирается количество алгоритмов,

ID	Название датасета:	Количество объектов	Количество признаков
1	ionosphere	351	33
2	throat surgery	470	37
3	wisconsin breast cancer	569	30
4	Indian liver	579	11
5	credit approval	653	46
6	Australian credit approval	690	38
7	blood transfusion	748	4
8	Pima Indians diabetes	768	8
9	mammographic mass	831	14
10	banknote authentication	1372	4
11	EEG Eye State	14980	14
12	adult	30162	102

Таблица 1: Датасеты.

которое было оптимально. В экспериментах $k = 900$, максимальное количество деревьев равно 3000, в датасетах EEG Eye State, adult количество деревьев было равно 5000. В методах HHCART, CO2, RidgeCART также проводилась кросс-валидация по параметрам регуляризации по сетке [0.001, 0.1, 1.0, 10.0, 100.0]. Темп обучения (shrinkage) во всех экспериментах жестко равен 0.1. Метрика качества – accuracy (доля правильных ответов прогноза). Baseline-решением является обычный градиентный бустинг.

4.2 Результаты экспериментов

По кросс-валидации находились параметры методов, которые предоставляют максимальную точность, а затем данный метод применялся на тестовой выборке. В таблице 2 показана точность алгоритмов на тестовой выборке. Видно, что применение линейных разбиений в узлах деревьев значительно улучшает качество прогноза. Все модификации бустинга оказались лучше baseline-решения, что доказывает целесообразность использования линейных решающих правил в узлах. Минусом предложенных методов является их долгий этап обучения.

Приведем график зависимости ошибки на тестовой и обучающей выборках в зависимости от глубины дерева на датасете mammographic mass. Результат на Рис. 3. Как видно из экспериментов, при увеличении глубины дерева увеличивается ошибка как и на валидации так и на тестовой выборке. Это означает, что в бустинге над деревьями с линейными решающими правилами целесообразно выбирать неглубокие деревья.

Как видно из 3 лучше всего на всех датасетах показали себя методы HHCART Boosting, Random CART(c) Boosting, Ridge CART(c) Boosting. Это показывает, что

ID	Boosting	HHCART Boosting	Random CART Boosting	Random CART(c) Boosting	Ridge CART Boosting	Ridge CART(c) Boosting	CO2 Boosting
1	90.91	89.77	**92.05**	86.36	85.23	82.95	88.64
2	80.51	**83.9**	**83.05**	**83.9**	**83.9**	**83.9**	**83.9**
3	90.91	**96.5**	**96.5**	**96.5**	**97.2**	**95.8**	**93.71**
4	66.21	**72.41**	**70.34**	**73.1**	**68.28**	**72.41**	**73.79**
5	82.32	**84.76**	**86.59**	**84.15**	**87.8**	**85.37**	**85.37**
6	80.35	**85.55**	**86.13**	**83.82**	**82.66**	**84.97**	**85.55**
7	76.47	**79.14**	74.33	**78.07**	74.33	**77.01**	74.87
8	75.52	**78.12**	**79.69**	**79.17**	**76.56**	**79.69**	**78.12**
9	78.85	**80.77**	**80.77**	**81.25**	**81.73**	**79.33**	**80.29**
10	99.13	**100.0**	**99.71**	**100.0**	**99.71**	**100.0**	98.83
11	88.79	**95.19**	84.62	**93.54**	**96.56**	**94.15**	**91.98**
12	86.12	**86.8**	84.76	**87.09**	**86.96**	**86.72**	85.69

Таблица 2: Сравнение методов градиентного бустинга между собой. Жирным выделены методы, которые оказались лучше baseline-решения.

HHCART Boosting	Random CART Boosting	Random CART(c) Boosting	Ridge CART Boosting	Ridge CART(c) Boosting	CO2 Boosting
11	9	**11**	10	**11**	8

Таблица 3: Количество датасетов, на которых предложенные алгоритмы сработали лучше, чем градиентный бустинг. Жирным выделены методы, которые оказались лучше всего на всех датасетах.

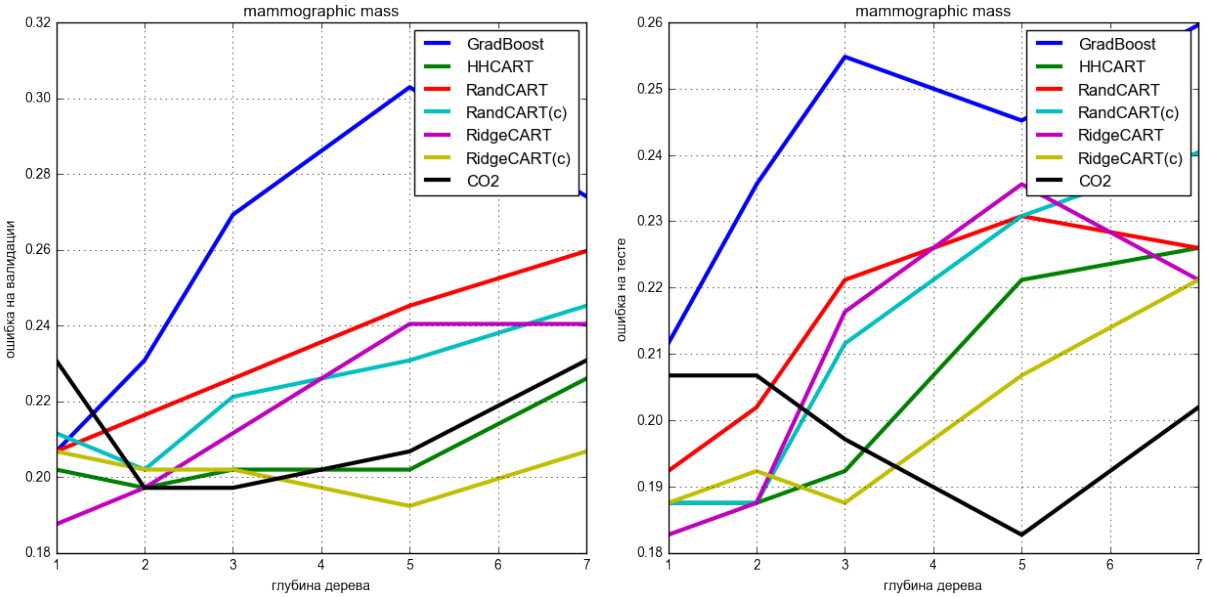
нужно не избавляться от старых признаков и искать оптимальное разбиение на основе них.

5 Заключение

Главным результатом исследований стало то, что применение линейных разбиений в узлах дерева в контексте бустинга улучшает качество прогноза.

Были достигнуты также следующие результаты:

1. предложено обобщить методы HHCART(A), HHCART(D) на случай задачи регрессии, с заменой собственного вектора на веса, полученные из гребневой регрессии.



a) Результат на валидации. b) Результат на тестовой выборке.

Рис. 3: Зависимость ошибки классификации от глубины дерева.

2. реализованы методы RidgeCART, RidgeCART(c), RandCART, RandCART(c), CO2, HHCART на языке python и код выложен в открытый доступ. Ссылка: https://github.com/Apogentus/advanced_decision_trees.

3. указанные методы применены в алгоритме градиентного бустинга и показали более высокую точность на большинстве задач, чем градиентный бустинг над обычными деревьями. Лучше всего показали себя градиентные бустинги над HHCART, Random CART(c), Ridge CART(c).

В качестве продолжения работы можно рассмотреть применение данных решающих деревьев в контексте XGBoost.

Список литературы

[1] *Wickramarachchi D.C.; Robertson B.L.; Reale, M.; Price, C.J.; Brown, J.* (2015) HHCART: An oblique decision tree. Computational Statistics and Data Analysis, p. 12-23.

[2] *Householder, A. S.* (1958) Unitary Triangularization of a Nonsymmetric Matrix. Journal ACM, p. 339-342.

[3] *Blaser, R.; Fryzlewicz, P.* (2015). Random Rotation Ensembles. // Journal of Machine Learning Research 2 (2015) 1-15

[4] *Китов В.В.* (2016) Исследование точности метода градиентного бустинга со случайными поворотами.

[5] UCI Machine Learning Repository [Электронный ресурс]. Режим доступа: https://archive.ics.uci.edu/ml/, свободный.

[6] *M. Norouzi, M. D. Collins, D. J. Fleet, and P. Kohli.* (2015) CO2 forest: Improved random forest by continuous optimization of oblique splits.