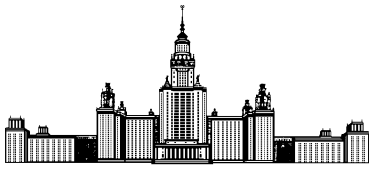
Московский государственный университет имени М. В. Ломоносова



Факультет Вычислительной Математики и Кибернетики

Кафедра Математических Методов Прогнозирования

Выпускная квалификационная работа

«Эволюционный алгоритм подбора гиперпараметров моделей»

Выполнил:

студент 4 курса 417 группы

Абрамов Валентин Андреевич

Научный руководитель:

к.ф.-м.н.

Китов Виктор Владимирович

Москва

2024

Содержание

Введение **4**

Часть 1 **4**

Обзор литературы **4**

Постановка задачи **5**

Существующие методы **6**

Поиск по сетке 6

Случайный поиск 6

Эволюционный алгоритм 7

Successive Halving 9

HyperBand 10

Tree-structured Parzen estimator 11

Предлагаемый метод: EvoHyperBand **13**

Предлагаемый метод: EvoHyperBandMut **15**

Вычислительные эксперименты **17**

Исходные данные и условия эксперимента 17

Оптимизация случайного леса 20

Оптимизация метода опорных векторов 22

Оптимизация градиентного бустинга 22

Анализ влияния параметров метода 23

Выводы **25**

Часть 2 **25**

Алгоритм DEHB **26**

Предлагаемая модификация метода DEHB **26**

Результаты экспериментов **27**

Выводы **29**

Результаты работы **29**

Список литературы **31**

Приложение **33**

Введение

В последние годы машинное обучение стало одной из самых активно развивающихся областей в информационных технологиях. Чтобы достичь оптимального качества моделей машинного обучения, необходимо правильно настроить конфигурацию этих моделей, то есть тщательно подобрать наилучшие гиперпараметры для этих моделей. Примерами гиперпараметров выступает число K в методе K ближайших соседей, сила регуляризации в линейных моделях, число слоев и нейронов на каждом слое в нейронных сетях и т.д.

Гиперпараметры являются важными параметрами моделей машинного обучения, которые не могут быть настроены по обучающей выборке градиентными методами — для этого необходимо оценивать точность модели на отдельной валидационной выборке либо перебирая всевозможные гиперпараметры вручную (что требует экспертизы и занимает много времени), либо используя автоматические методы безградиентной оптимизации. Простейшие методы автоматического выбора гиперпараметров, такие, как перебор по равномерной сетке или случайный поиск, могут быть неэффективными или требовать значительных вычислительных ресурсов, поэтому вместо них используют более сложные алгоритмы, например, эволюционные [1].

Работа состоит из двух частей: в первой предлагается новое исследование – новые алгоритмы оптимизации гиперпараметров, во второй – модификация алгоритма DEHB [2]. Вторая часть была представлена на конференции ИТиММ-2024 в РЭУ имени Г.В. Плеханова.

Часть 1

В данной части рассматриваются существующие методы оптимизации гиперпараметров, работающие как с непрерывными, так и с категориальными параметрами. Предлагается новый метод, проводятся эксперименты для сравнения результатов работы нового метода с другими.

Обзор литературы

Гиперпараметрическая оптимизация — большая и перспективная область исследований. В данный момент в исследованиях поднимаются вопросы переобучения гиперпараметров на валидационную выборку и параллелизации алгоритмов [3]. Для глубоких

нейронных сетей также необходимо подбирать гиперпараметры, такие, как число слоев, скорость обучения и даже связи между слоями [4].

Задача автоматического подбора гиперпараметров активно исследуется. Широко используемые методы перебора значений по сетке и случайного поиска имеют низкую вычислительную эффективность, поскольку не учитывают информацию о качестве ранее протестированных конфигураций (наборов гиперпараметров) для генерации новых конфигураций.

В 2011 году был создан подход SMBO — последовательная оптимизация, основанная на суррогатной функции, то есть функции, приближающей целевую оптимизируемую функцию, но быструю для расчета [5]. В качестве суррогатной функции изначально брали гауссовские процессы, но они подходят не для всех задач, так как предназначены только для непрерывных гиперпараметров [6].

На данный момент одним из лучших методов автоматического подбора гиперпараметров является метод TPE, основанный на байесовском подходе [7]. Также существуют методы, основанные на подходе SMBO, такие, как SMAC, использующий случайный лес в качестве суррогата, или модели, использующие суррогат на основе гауссовских процессов [5, 6, 8, 9].

При оптимизации гиперпараметров также используются эволюционные алгоритмы, так как они позволяют оптимизировать любые функции за счет интеллектуального перебора значений, основанного на таких идеях эволюции, как мутация, скрещивание, селекция и естественный отбор [10, 11].

Хорошее качество обеспечивает метод Successive Halving (SH) – он позволяет найти лучший набор гиперпараметров в условиях ограниченности вычислительных мощностей [12]. На основе SH созданы алгоритмы HyperBand, автоматически выделяющий ресурсы для нескольких запусков SH, BOHB, добавляющий в HyperBand суррогат на основе подходов TPE, и DEHB, добавляющий в HyperBand идею дифференциальной эволюции [2, 13, 14].

Постановка задачи

Задача гиперпараметрической оптимизации состоит в том, чтобы имея обучающую выборку $X \in \mathbb{R}^{N \times d}, y \in \mathbb{R}^N$, пространство гиперпараметров $\Theta \in \mathbb{R}^k$, модель $f(\theta) : \mathbb{R}^d \to \mathbb{R},\ \theta \in \Theta,\ f \in \mathbb{H}$ и функцию ошибок $\mathcal{L}(f, X, y) : \mathbb{H} \times \mathbb{R}^{N \times d} \times \mathbb{R}^N \to \mathbb{R}$, найти θ^* такую,

что выполнено

$$\theta^* = \arg\min_{\theta} \mathcal{L}(f(\theta), X, y)$$

Таким образом, зная модель f, пространство гиперпараметров Θ, набор данных X, y и функцию ошибок $\mathcal{L}$, необходимо найти набор гиперпараметров θ^*, при котором значение функции ошибок $\mathcal{L}$ наименьшее при обучении модели f на данных X, y.

Предполагается, что набор гиперпараметров состоит из непрерывных и категориальных переменных, функция $\mathcal{L}$ имеет бюджет вычислений. Им может быть ограничение на максимальное число итераций при обучении нейронной сети, количество деревьев в случайном лесе и градиентном бустинге и т. п.

Существующие методы

Поиск по сетке

Поиск по сетке работает следующим образом:

1. Задаются наборы значений параметров.
2. Для каждой комбинации параметров вычисляется значение целевой функции.

У этого подхода есть множество известных проблем:

1. Необходимость выбора сетки значений вручную.
2. Медленное время работы.
3. В случае, когда один гиперпараметр влияет сильнее, чем другие, перебор неэффективен из-за многократного использования одного значения существенного гиперпараметра.

Случайный поиск

Случайный поиск работает следующим образом:

1. Наборы параметров случайно сэмплируются из некоторого распределения.
2. Для каждого набора параметров вычисляется значение целевой функции.

Случайный поиск избавлен от проблем поиска по сетке, но также имеет недостатки:

1. Перебор не использует вычисленные значения функции ошибок.

2. Поиск ненаправленный, то есть наборы гиперпараметров сэмплируются полностью случайно.

Эволюционный алгоритм

Для краткости будем называть наборы гиперпараметров кандидатами. Классический эволюционный алгоритм работает следующим образом:

1. Создается набор кандидатов размера N, называемый популяцией.

2. Повторяются следующие действия:

   (a) Вычисляется значение оптимизируемой функции для каждого кандидата.

   (b) Отбирается M лучших по значению функции кандидатов.

   (c) Создаются новые $N - M$ кандидатов путем скрещивания 2 случайных кандидатов из M лучших и мутации.

Начальные кандидаты сэмплируются из равномерного распределения или из распределений, выбираемых пользователем. В качестве операции скрещивания используется биномиальное скрещивание – выбор значения параметра одного из кандидатов-родителей с вероятностью 0.5. Мутация – замена параметра на случайное значение с вероятностью p. В таком подходе N, M, p – параметры эволюционного алгоритма, эволюция происходит, пока не будет достигнуто определенное число итераций, время работы или пока лучшее достигнутое значение оптимизируемой функции не перестанет изменяться в течение некоторого заранее определенного числа итераций. Результат работы такого алгоритма - лучший кандидат по значению функции.

Algorithm 1 Эволюционный алгоритм

Ввод: N – размер популяции, M – количество выживших на каждом этапе, p – вероятность мутации, f – оптимизируемая функция, распределения гиперпараметров $p_j(\theta)(j = 1, \ldots, d)$, условие остановки.

Вывод: θ^* – набор гиперпараметров с наименьшим значением f из всех обработанных наборов.

$\Theta \leftarrow \{\theta^k \mid k = 1, \ldots, N\},\ \theta_j^k \sim p_j(\theta)\ \forall j = 1, \ldots, d$
while <не выполено условие остановки> **do**
  $F \leftarrow \{f(\theta) \mid \forall \theta \in \Theta\}$
  $\Theta \leftarrow \{M$ лучших по значению f наборов гиперпараметров$\}$
  $\Theta_{new} \leftarrow \emptyset$
  while $|\Theta_{new}| < N - M$ **do**
    $\theta_1, \theta_2 \leftarrow \Theta$ – случайный выбор без возвращения двух родителей
    $\theta_{new} \leftarrow Crossover(\theta_1, \theta_2)$ – скрещивание
    $\theta_{new} \leftarrow Mutation(\theta_{new}, p, p_j(\theta)(j = 1, \ldots, d))$ – мутация
    $\Theta_{new} \leftarrow \Theta_{new} \cup \{\theta_{new}\}$ – мутация
  end while
  $\Theta \leftarrow \Theta \cup \Theta_{new}$
end while

Перечислим слабые стороны эволюционных алгоритмов:

1. Алгоритмы требуют многократного вычисления функции f, которая может вычисляться очень долго при обучении моделей.

2. При мутации не учитывается положительный опыт, есть риск ухудшения набора гиперпараметров при неудачном сэмплировании.

Successive Halving

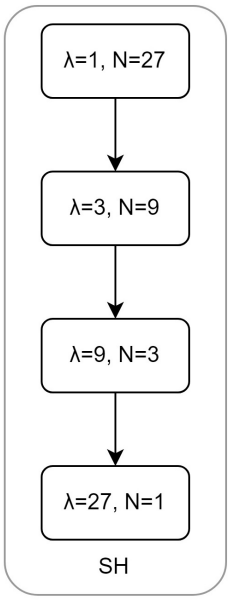


Рис. 1: Схема работы Successive Halving, $\lambda_{min} = 1, \lambda_{max} = 27, \eta = 3$

Successive Halving (рис. 1) – алгоритм подбора гиперпараметров, подразумевающий наличие бюджета вычислений, то есть целевую функцию f можно вычислить с фиксированным бюджетом [12]. Например, можно ограничить количество итераций обучения нейросети или ограничить число деревьев при обучении случайного леса. У алгоритма есть три гиперпараметра: минимальный бюджет λ_{min}, максимальный бюджет λ_{max}, и η – параметр шкалирования. На первой итерации из равномерного распределения сэмплируются N_1 кандидатов, для них вычисляется функция f с бюджетом $\lambda_1 = \lambda_{min}$. На второй итерации берутся $N_2 = N_1/\eta$ лучших кандидатов, для них вычисляется f с бюджетом $\lambda_2 = \eta\lambda_1$. На i-ой итерации функция f вычисляется для $N_i = N_1/\eta^{i-1}$ лучших кандидатов с шага $i-1$ с бюджетом $\lambda_i = \eta^{i-1}\lambda_1$. λ_1 и N_1 выбираются так, чтобы бюджет вычислений на последней итерации был равен 1 для единственного кандидата. Таким образом, алгоритм работает как случайный поиск, который выделяет больше вычислительных ресурсов перспективным кандидатам и отсекает тех, кто при низком бюджете показывает себя хуже остальных.

Algorithm 2 Successive Halving

Ввод: Бюджеты вычислений $\lambda_{min}, \lambda_{max}$, параметр шкалирования η, максимальное число наборов N_{max}, функция ошибок $f(\theta, \lambda)$, распределения гиперпараметров $p_j(\theta)(j = 1, \ldots, d)$.

Вывод: θ^* – набор гиперпараметров с наименьшим значением $f(\cdot, \lambda_{max})$ из всех обработанных наборов.

$s \leftarrow \lfloor \log_\eta \frac{\lambda_{max}}{\lambda_{min}} \rfloor + 1$

$\lambda_1 \leftarrow \lambda_{min}$

$N_1 \leftarrow N_{max}$

$\Theta \leftarrow \{\theta^k \mid k = 1, \ldots, N_{max}\},\ \theta_j^k \sim p_j(\theta)\ \forall j = 1, \ldots, d$

for $k = 1, \ldots, s$ **do**

  $F \leftarrow \{f(\theta, \lambda_k) \mid \forall \theta \in \Theta\}$

  $\Theta \leftarrow \{\lfloor \frac{N_k}{\eta} \rfloor$ лучших по значению $f(\cdot, \lambda_k)$ наборов гиперпараметров$\}$

  $N_{k+1} \leftarrow \lfloor \frac{N_k}{\eta} \rfloor$

  $\lambda_{k+1} \leftarrow \lambda_k \eta$

end for

HyperBand

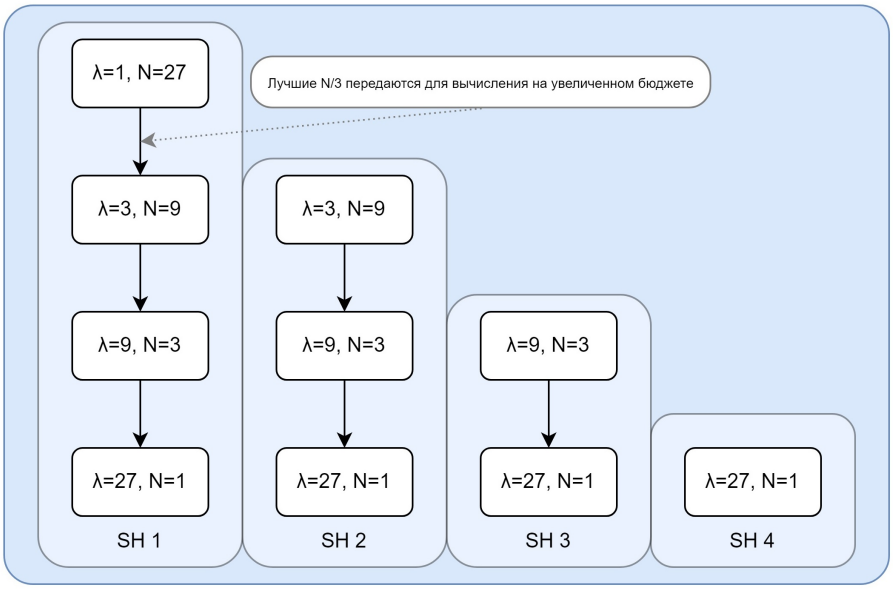


Рис. 2: Схема работы HyperBand, $\lambda_{min} = 1, \lambda_{max} = 27, \eta = 3$

HyperBand (рис. 2) расширяет идеи SH – производится несколько запусков SH с различными значениями λ_{min} [13]. Для каждого нового запуска увеличивается мини-

мальный бюджет. Таким образом, функции f, которые долго сходятся, проверяются для случайных инициализаций и с низким бюджетом, и с высоким.

Algorithm 3 HyperBand

Ввод: Бюджеты вычислений $\lambda_{min}, \lambda_{max}$, параметр шкалирования η, максимальное число наборов N_{max}, функция ошибок $f(\theta, \lambda)$, распределения гиперпараметров $p_j(\theta)(j = 1, \ldots, d)$.

Вывод: θ^* – набор гиперпараметров с наименьшим значением $f(\cdot, \lambda_{max})$ из всех обработанных наборов. Возвращаем лучшую θ^*

1: $s_{max} \leftarrow \lfloor \log_\eta \frac{\lambda_{max}}{\lambda_{min}} \rfloor + 1$
2: $\Theta^* \leftarrow \emptyset$ – множество оптимальных значений для каждого запуска SH.
3: **for** $s = 0, \ldots, s_{max}$ **do**
4: $\quad \hat{\lambda}_{min} \leftarrow \lambda_{min}\eta^s$
5: $\quad \hat{N}_{max} \leftarrow \frac{N_{max}}{\eta^s}$
6: $\quad \theta^* \leftarrow SuccessiveHalving(\hat{\lambda}_{min}, \lambda_{max}, \eta, \hat{N}_{max}, f, p_j(\theta)(j = 1, \ldots, d))$
7: $\quad \Theta^* \leftarrow \Theta^* \cup \{\theta^*\}$
8: **end for**

Параметры $\lambda_{min}, \lambda_{max}$ и N_{max} обычно выбираются из следующих соображений: λ_{min} выбирается как минимальный бюджет, при котором имеется смысл вычислять оптимизируемую функцию, λ_{max} выбирается как максимальный приемлемый бюджет вычислений, N_{max} выбирается равным λ_{max} (так на последней итерации SH останется ровно один набор гиперпараметров).

У SH и HyperBand есть следующие слабые стороны:

1. Алгоритмы полностью полагаются на случайное сэмплирование.
2. Алгоритмы не учитывают положительный и отрицательный опыт при генерации новых кандидатов.
3. Не всегда гиперпараметры, отсеянные при низком бюджете, являются плохими при высоком.

Tree-structured Parzen estimator

Данный метод в начале работает как случайный поиск, а после оценки некоторого заранее заданного числа значений функции начинает действовать по следующему алгоритму [7]:

1. Набор параметров строится следующим образом. Для каждого параметра выполняются действия:

   (a) На основе значений целевой функции строятся распределения $l_{par}(x)$ лучших значений параметра и $g_{par}(x)$ худших значений параметра.

   (b) Сэмплируются новые значения параметра из $l_{par}(x)$.

   (c) Значение параметра выбирается на основе полученных распределений лучших и худших как значение, максимизирующее Expected Improvement: $EI = \frac{l_{par}(x)}{g_{par}(x)}$

2. Для полученного набора параметров вычисляется значение целевой функции.

Этот алгоритм является одним из самых технически продвинутых на данный момент, но у него также есть слабые стороны:

1. Для корректного построения распределений параметров $l_{par}(x)$ и $g_{par}(x)$ необходим "прогрев" случайным поиском в течение большого числа итераций.

2. Алгоритм требует многократного вычисления функции f, которая может вычисляться очень долго при обучении моделей.

Стоит сказать, что есть и другие методы, например, основанные на байесовской оптимизации, но они работают только с непрерывными гиперпараметрами, а на данный момент у моделей машинного обучения достаточно большая часть параметров дискретна или имеет только строковое представление.

Предлагаемый метод: EvoHyperBand

Предлагается объединить Successive Halving и эволюционный алгоритм следующим образом: на этапе перехода между уровнями бюджета λ_i и $\lambda_{i+1} = \eta\lambda_i$ брать $\frac{N_i}{\nu\eta}$ лучших кандидатов, а оставшиеся $\frac{N_i}{\eta} - \frac{N_i}{\nu\eta}$ получать скрещиванием и мутацией лучших, как в эволюционном алгоритме. Таким образом, получим метод, который:

1. Не будет полностью полагаться на случайное сэмплирование в начале итерации Successive Halving.

2. Благодаря мутации на высоких уровнях бюджета возможно возвращение некоторых значений параметров, которые были на низких уровнях бюджета.

3. Перебор будет более широким за счет дополнительной случайности при мутации.

Algorithm 4 Модификация Successive Halving: **EvoSH**

Ввод: Бюджеты вычислений $\lambda_{min}, \lambda_{max}$, параметр шкалирования η, **параметр доли** ν**, вероятность мутации** p, максимальное число наборов N_{max}, функция ошибок $f(\theta, \lambda)$, распределения гиперпараметров $p_j(\theta)(j = 1, \ldots, d)$.

Вывод: θ^* – набор гиперпараметров с наименьшим значением $f(\cdot, \lambda_{max})$ из всех обработанных наборов.

$s \leftarrow \lfloor \log_\eta \frac{\lambda_{max}}{\lambda_{min}} \rfloor + 1$

$\lambda_1 \leftarrow \lambda_{min}$

$N_1 \leftarrow N_{max}$

$\Theta \leftarrow \{\theta^k \mid k = 1, \ldots, N_{max}\},\ \theta_j^k \sim p_j(\theta)\ \forall j = 1, \ldots, d$

for $k = 1, \ldots, s$ **do**

  $F \leftarrow \{f(\theta, \lambda_k) \mid \forall \theta \in \Theta\}$

  $\Theta \leftarrow \{\lfloor \frac{N_k}{\eta\nu} \rfloor$ лучших по значению $f(\cdot, \lambda_k)$ наборов гиперпараметров$\}$

  $\Theta_{new} \leftarrow \emptyset$

  while $|\Theta_{new}| < \lfloor \frac{N_k}{\eta} \rfloor - \lfloor \frac{N_k}{\eta\nu} \rfloor$ **do**

    $\theta_1, \theta_2 \leftarrow \Theta$ – случайный выбор без возвращения двух родителей

    $\theta_{new} \leftarrow Crossover(\theta_1, \theta_2)$ – скрещивание

    $\theta_{new} \leftarrow Mutation(\theta_{new}, p, p_j(\theta)(j = 1, \ldots, d))$ – мутация (сэмплирование параметров из заданных распределений с вероятностью p)

    $\Theta_{new} \leftarrow \Theta_{new} \cup \{\theta_{new}\}$

  end while

  $\Theta \leftarrow \Theta \cup \Theta_{new}$

  $N_{k+1} \leftarrow \lfloor \frac{N_k}{\eta} \rfloor$

  $\lambda_{k+1} \leftarrow \lambda_k \eta$

end for

Algorithm 5 EvoHyperBand

Ввод: Бюджеты вычислений $\lambda_{min}, \lambda_{max}$, параметр шкалирования η, **параметр доли** ν**, вероятность мутации** p максимальное число наборов N_{max}, функция ошибок $f(\theta, \lambda)$, распределения гиперпараметров $p_j(\theta)(j = 1, \ldots, d)$.

Вывод: θ^* – набор гиперпараметров с наименьшим значением $f(\cdot, \lambda_{max})$ из всех обработанных наборов.

1: $s_{max} \leftarrow \lfloor \log_\eta \frac{\lambda_{max}}{\lambda_{min}} \rfloor + 1$
2: $\Theta^* \leftarrow \emptyset$ – множество оптимальных значений для каждого запуска SH.
3: **for** $s = 0, \ldots, s_{max}$ **do**
4: $\quad \hat{\lambda}_{min} \leftarrow \lambda_{min}\eta^s$
5: $\quad \hat{N}_{max} \leftarrow \frac{N_{max}}{\eta^s}$
6: $\quad \theta^* \leftarrow EvoSH(\hat{\lambda}_{min}, \lambda_{max}, \eta, \nu, p, \hat{N}_{max}, f, p_j(\theta)(j = 1, \ldots, d))$
7: $\quad \Theta^* \leftarrow \Theta^* \cup \{\theta^*\}$
8: **end for**

Данную модификацию Successive Halving будем запускать в режиме HyperBand и назовем данный алгоритм EvoHyperBand. Эксперименты показали, что для большинства задач оптимальное значение $\nu = 2$.

Предлагаемый метод: EvoHyperBandMut

У предложенного метода остается проблема эволюционных алгоритмов: при мутации никак не учитывается предыдущий положительный опыт, то есть сэмплирование при мутации полностью случайное.

Предлагается модифицировать мутацию следующим образом:

- При переходе с первого уровня первой итерации Successive Halving в мутации происходит сэмплирование из заданных пользователем распределений (как в эволюционном алгоритме выше).
- Все наборы гиперпараметров и значения целевой функции на них сохраняются.
- После каждой итерации Successive Halving для каждого оптимизируемого гиперпараметра *par* строится оценка плотности $\hat{p}_{par}$.
  - Для непрерывных – ядровая оценка плотности суммой гауссиан с параметром $bandwidth = n^{-\frac{1}{5}}$ (n – количество точек), для категориальных – частотная

оценка. Параметр *bandwidth* взят из сборника [15].

– Плотность оценивается по наборам гиперпараметров, для которых значение целевой функции больше некоторого квантиля χ (χ брался равным 0.5).

- При мутации каждый мутируемый параметр сэмплируется из плотности $\hat{p}_{par}$ с вероятностью p (параметр метода).

Предполагается, что такая модификация позволит получать лучшие наборы гиперпараметров в результате мутации, назовем ее EvoHyperBandMut.

Algorithm 6 EvoHyperBandMut

Ввод: Бюджеты вычислений $\lambda_{min}, \lambda_{max}$, параметр шкалирования η, **параметр доли ν, параметр квантиля χ, вероятность мутации** p, максимальное число наборов N_{max}, функция ошибок $f(\theta, \lambda)$, распределения гиперпараметров $p_j(\theta)(j = 1, \ldots, d)$.

Вывод: θ^* – набор гиперпараметров с наименьшим значением $f(\cdot, \lambda_{max})$ из всех обработанных наборов.

$s_{max} \leftarrow \lfloor \log_\eta \frac{\lambda_{max}}{\lambda_{min}} \rfloor$

$\hat{p}_j(\theta) \leftarrow p_j(\theta)\ (j = 1, \ldots, d)$

$\hat{\Theta} \leftarrow \emptyset$

$\hat{F} \leftarrow \emptyset$

$\Theta^* \leftarrow \emptyset$ – множество оптимальных значений для каждого запуска SH.

for $s = 0, \ldots, s_{max}$ **do**

    $\hat{\lambda}_{min} \leftarrow \lambda_{min}\eta^s$

    $\hat{N}_{max} \leftarrow \frac{N_{max}}{\eta^s}$

    $\theta^*, \hat{\Theta}, \hat{F} \leftarrow EvoSHMut(\hat{\lambda}_{min}, \lambda_{max}, \eta, \nu, \chi, p, \hat{N}_{max}, f, p_j(\theta), \hat{p}_j(\theta)), \hat{\Theta}, \hat{F})$

    $\Theta^* \leftarrow \Theta^* \cup \{\theta^*\}$

end for

Algorithm 7 EvoSHMut

Ввод: Бюджеты вычислений $\lambda_{min}, \lambda_{max}$, параметр шкалирования η, **параметр доли** ν**, параметр квантиля** χ**, вероятность мутации** p, максимальное число наборов N_{max}, функция ошибок $f(\theta, \lambda)$, распределения гиперпараметров $p_j(\theta)(j = 1, \ldots, d)$, **оценки распределений гиперпараметров** $\hat{p}_j(\theta)(j = 1, \ldots, d)$**, множество наборов гиперпараметров для построения оценок** $\hat{\Theta}$**, множество значений функции** f **для построения оценок** $\hat{F}$.

Вывод: θ^* – набор гиперпараметров с наименьшим значением $f(\cdot, \lambda_{max})$ из всех обработанных наборов, $\hat{\Theta}, \hat{F}$.

1: $s \leftarrow \lfloor \log_\eta \frac{\lambda_{max}}{\lambda_{min}} \rfloor$
2: $\lambda_1 \leftarrow \lambda_{min}$
3: $N_1 \leftarrow N_{max}$
4: $\Theta \leftarrow \{\theta^k \mid k = 1, \ldots, N_{max}\},\ \theta_j^k \sim p_j(\theta)\ \forall j = 1, \ldots, d$
5: **for** $k = 1, \ldots, s$ **do**
6: $\quad F \leftarrow \{f(\theta, \lambda_k) \mid \forall \theta \in \Theta\}$
7: $\quad \hat{F} = \hat{F} \cup F$
8: $\quad \hat{F}_\chi = Quantile(\hat{F}, \chi)$
9: $\quad \hat{\Theta} \leftarrow \{\theta \mid \theta \in \Theta \cup \hat{\Theta}, f(\theta, \lambda_k) < \hat{F}_\chi\}$
10: $\quad \hat{p}_j(\theta) \leftarrow KDE(\{\theta_j \mid \theta \in \hat{\Theta}\})\ (j = 1, \ldots, d)$
11: $\quad \Theta \leftarrow \{\lfloor \frac{N_k}{\eta\nu} \rfloor$ лучших по значению $f(\cdot, \lambda_k)$ наборов гиперпараметров из$\Theta\}$
12: $\quad \Theta \leftarrow EvoSampling(\Theta, p, \hat{p}_j(\theta)(j = 1, \ldots, d))$ (строки 8–15 в алгоритме 4)
13: $\quad N_{k+1} \leftarrow \lfloor \frac{N_k}{\eta} \rfloor$
14: $\quad \lambda_{k+1} \leftarrow \lambda_k \eta$
15: **end for**

Таким образом, у метода три гиперпараметра: ν – доля кандидатов, χ – квантиль для выбора значений параметров в мутации и p – вероятность мутации. χ рекомендуется брать равным 0.5.

Вычислительные эксперименты

Исходные данные и условия эксперимента

Параметры ν, χ фиксированы и равны соответственно 2 и $\frac{1}{2}$ (медиане), чтобы сравнение с эволюционным алгоритмом было корректным (если не оговорено иное). Вероятность

мутации p выбрана равной 0.3. Параметр $\lambda_{min} = 1$.

Для экспериментов была написана собственная реализация эволюционного алгоритма и HyperBand для точности сравнения с предложенным методом. Реализация полностью совместима с интерфейсом моделей библиотеки scikit-learn [16]. Параметр шкалирования η в HyperBand взят равным 3 во всех экспериментах, $\lambda_{min} = 1$

Размер популяции N в эволюционном алгоритме – 10, $M = \frac{N}{2}$. Вероятность мутации $p = 0.3$. Распределения гиперпараметров при мутации выбраны равномерными. Скрещивание – биномиальное, шанс взятия параметра у родителя 1 – 0.5, у родителя 2 – 0.5.

Предложенные методы сравнивались с HyperBand, эволюционным алгоритмом и Tree-structured Parzen Estimator (TPE) из библиотеки optuna [17]. TPE запускался со стандартными параметрами optuna. Для корректного сравнения все алгоритмы запускались с одинаковым ограничением по времени работы, равным времени работы HyperBand.

В качестве моделей для оптимизации гиперпараметров использовались реализации случайного леса и метода опорных векторов из библиотеки scikit-learn, реализация градиентного бустинга из библиотеки LightGBM [16, 18]. Для случайного леса рассматривались следующие гиперпараметры:

- Доля признаков $\in [0.1, 0.9]$.
- Минимальное количество объектов для разбиения $\in [2, 200]$
- Минимальное количество объектов в листе дерева $\in [1, 100]$
- Критерий разбиения $\in \{gini, entropy\}$

В качестве бюджета вычислений использовалось число деревьев.

Для метода опорных векторов рассматривались следующие гиперпараметры:

- Коэффициент регуляризации $C \in [0.01, 10]$.
- Тип ядра $\in \{linear, poly, rbf, sigmoid\}$
- Коэффициент γ для ядра $\in [0.0001, 1]$

В качестве бюджета вычислений использовалось максимальное число итераций оптимизатора. Параметр $coef0 = 0$.

Для градиентного бустинга рассматривались следующие гиперпараметры:

- Максимальная глубина $\in [1, 20] \cup \{-1\}$.
- Темп обучения $\in [0.001, 0.5]$
- Количество листьев $\in [20, 127]$
- Коэффициент L1-регуляризации $\in [0, 2]$
- Коэффициент L2-регуляризации $\in [0, 2]$
- Минимальное количество объектов в листе $\in [10, 400]$

В качестве бюджета вычислений использовалось число деревьев.

Метод тестируется на наборах данных с сайта open-ml (таб. 1) [19]:

Название	Тип	Описание
credit-g	Классификация	Риск дефолта по кредиту
climate-model-simulation-crashes	Классификация	Результат симуляции климата
qsar-biodeg	Классификация	Биодеградация химикатов
ilpd	Классификация	Детекция пациентов
blood-transfusion-service-center	Классификация	Классификация доноров крови
kr-kp	Классификация	Исход шахматной игры
space-ga	Регрессия	Явка на выборах
topo_2_1	Регрессия	Разработка лекарств
us_crime	Регрессия	Количество преступлений в округе

Таблица 1: Наборы данных

Наборы данных (таб. 1) разбивались на обучающую и тестовую выборки в соотношении 7:3. Алгоритмы оптимизировали среднее значение функции потерь (или качества), вычисленное по кросс-валидации с 3 разбиениями по обучающей выборке. В задачах регрессии использовалась функция ошибки MSE, в задачах классификации оптимизировалась точность (accuracy).

Для каждого датасета выполнено 10 запусков каждого алгоритма. После подбора гиперпараметров по кросс-валидации модель обучалась на лучшем наборе на всех данных обучающей выборки для получения предсказания на тесте. В таблицах отображены средние значения качества и среднеквадратичные отклонения, оцененные по запускам.

Оптимизация случайного леса

Рассмотрим значения точности в задачах классификации при использовании случайного леса с бюджетом $\lambda_{max} = 243 = \eta^5$.

Алгоритм	credit-g	climate	qsar
HyperBand	0.7439 ± 0.0055	0.9171 ± 0.0009	0.8543 ± 0.0036
Эволюция	0.7394 ± 0.0078	0.9167 ± 0.0000	0.8294 ± 0.0122
EvoHyperBand	$\mathbf{0.7453 \pm 0.0051}$	0.9174 ± 0.0015	0.8538 ± 0.0057
EvoHyperBandMut	0.7440 ± 0.0068	$\mathbf{0.9174 \pm 0.0011}$	$\mathbf{0.8562 \pm 0.0041}$
TPE	0.7425 ± 0.0029	0.9167 ± 0.0000	0.8446 ± 0.0047

Таблица 2: Точность случайного леса на кросс-валидации (задачи классификации)

Рассмотрим значения среднеквадратичной ошибки в задачах регрессии для случайного леса с бюджетом $\lambda_{max} = 243 = \eta^5$.

Алгоритм	space_ga	topo_2_1	us_crime
HyperBand	0.0154 ± 0.0005	0.0008 ± 0.0000	0.0191 ± 0.0003
Эволюция	0.0161 ± 0.0005	0.0008 ± 0.0000	0.0194 ± 0.0001
EvoHyperBand	$\mathbf{0.0153 \pm 0.0004}$	0.0008 ± 0.0000	$\mathbf{0.0191 \pm 0.0002}$
EvoHyperBandMut	$\mathbf{0.0153 \pm 0.0002}$	0.0008 ± 0.0000	0.0192 ± 0.0002
TPE	0.0154 ± 0.0003	0.0008 ± 0.0000	0.0194 ± 0.0000

Таблица 3: MSE случайного леса на кросс-валидации (задача регрессии)

Видим, что предложенные методы во всех случаях дают лучшее значение качества на кросс-валидации, обгоняя не только базовые алгоритмы эволюции и HyperBand, но и алгоритм TPE, работающий по другому принципу (таб. 2, таб. 3).

Рассмотрим точность на отложенной выборке для этих же задач.

Алгоритм	credit-g	climate	qsar
HyperBand	0.7690 ± 0.0151	0.9065 ± 0.0028	0.8768 ± 0.0079
Эволюция	$\mathbf{0.7705 \pm 0.0096}$	$\mathbf{0.9074 \pm 0.0000}$	0.8540 ± 0.0101
EvoHyperBand	0.7590 ± 0.0188	$\mathbf{0.9074 \pm 0.0000}$	0.8730 ± 0.0082
EvoHyperBandMut	0.7565 ± 0.0147	0.9046 ± 0.0059	$\mathbf{0.8768 \pm 0.0067}$
TPE	$\mathbf{0.7705 \pm 0.0098}$	$\mathbf{0.9074 \pm 0.0000}$	$\mathbf{0.8777 \pm 0.0076}$

Таблица 4: Точность случайного леса на отложенной выборке (задачи классификации)

В таблице 4 видно, что не всегда лучшие значения на кросс-валидации приводят к лучшим значениям на тесте. Это может быть связано со смещенностью оценки на параметры при малом количестве разбиений в кросс-валидации или из-за того, что отложенная выборка может иметь другую структуру, например, другие распределения базовых признаков. Далее будем рассматривать только кросс-валидацию, значения на тесте можно найти в приложении.

Теперь рассмотрим значения точности в задачах классификации при использовании случайного леса с бюджетом 20.

Алгоритм	credit-g	climate	qsar
HyperBand	0.7331 ± 0.0093	0.9167 ± 0.0000	0.8377 ± 0.0141
Эволюция	0.7346 ± 0.0092	0.9167 ± 0.0000	0.8263 ± 0.0150
EvoHyperBand	0.7313 ± 0.0083	0.9167 ± 0.0000	0.8232 ± 0.0129
EvoHyperBandMut	0.7335 ± 0.0058	0.9167 ± 0.0000	0.8374 ± 0.0157
TPE	$\mathbf{0.7429 \pm 0.0061}$	0.9167 ± 0.0000	$\mathbf{0.8401 \pm 0.0056}$

Таблица 5: Точность случайного леса на кросс-валидации, бюджет 20 (задачи классификации)

Методы, в основе которых лежит HyperBand, показывают (таб. 5) себя хуже в условиях низкого бюджета вычислений. При этом нет необходимости использовать их в таких задачах, так как целевая функция быстро вычисляется и каждая итерация вносит сильный вклад в итоговое качество.

Оптимизация метода опорных векторов

Рассмотрим значения точности для метода опорных векторов при бюджете вычислений 1000.

Алгоритм	ilpd	blood	kr-kp
HyperBand	0.7279 ± 0.0032	0.7702 ± 0.0048	0.9499 ± 0.0044
Эволюция	0.7380 ± 0.0041	0.7840 ± 0.0012	0.9596 ± 0.0025
EvoHyperBand	0.7270 ± 0.0030	0.7757 ± 0.0049	0.9552 ± 0.0027
EvoHyperBandMut	0.7294 ± 0.0040	0.7716 ± 0.0056	0.9550 ± 0.0032
TPE	$\mathbf{0.7425 \pm 0.0000}$	$\mathbf{0.7877 \pm 0.0000}$	$\mathbf{0.9644 \pm 0.0000}$

Таблица 6: Точность метода опорных векторов на кросс-валидации (задачи классификации)

Предложенные методы дают качество в среднем лучшее, чем HyperBand, и при этом худшее, чем эволюционный алгоритм (таб. 6). Лучшее качество в этих задачах достигается с помощью эволюционного алгоритма или TPE. Это может быть связано с тем, что в качестве бюджета вычислений используется количество итераций решения оптимизационной задачи метода опорных векторов, и оценивать качество по малому числу итераций до сходимости некорректно.

Оптимизация градиентного бустинга

Рассмотрим значения точности для градиентного бустинга при бюджете вычислений 243.

Алгоритм	credit-g	ilpd	kr-kp
HyperBand	0.7529 ± 0.0024	0.7117 ± 0.0042	0.9848 ± 0.0025
Эволюция	0.7530 ± 0.0049	0.7132 ± 0.0046	0.9836 ± 0.0028
EvoHyperBand	$\mathbf{0.7550 \pm 0.0037}$	0.7097 ± 0.0046	0.9850 ± 0.0018
EvoHyperBandMut	0.7540 ± 0.0048	$\mathbf{0.7134 \pm 0.0055}$	$\mathbf{0.9854 \pm 0.0012}$
TPE	0.7485 ± 0.0020	0.7060 ± 0.0000	0.9847 ± 0.0010

Таблица 7: Точность градиентного бустинга на кросс-валидации (задачи классификации)

Рассмотрим значения среднеквадратичной ошибки в задачах регрессии для градиентного бустинга при бюджете вычислений 500.

Алгоритм	space_ga	topo_2_1	us_crime
HyperBand	0.0129 ± 0.0001	0.0008 ± 0.0001	0.0189 ± 0.0001
Эволюция	0.0128 ± 0.0002	$\mathbf{0.0008 \pm 0.0000}$	0.0187 ± 0.0001
EvoHyperBand	0.0129 ± 0.0002	0.0008 ± 0.0001	$\mathbf{0.0187 \pm 0.0001}$
EvoHyperBandMut	$\mathbf{0.0128 \pm 0.0001}$	$\mathbf{0.0008 \pm 0.0000}$	0.0188 ± 0.0001
TPE	$\mathbf{0.0128 \pm 0.0001}$	$\mathbf{0.0008 \pm 0.0000}$	$\mathbf{0.0187 \pm 0.0001}$

Таблица 8: MSE градиентного бустинга на кросс-валидации (задачи регрессии)

При оптимизации градиентного бустинга предложенные методы во всех случаях дают лучшее значение качества на кросс-валидации (таб. 7, таб. 8).

Таким образом, при сравнении на реальных задачах в большинстве случаев предложенный алгоритм дает лучшее качество, а модификация мутации позволяет достичь качества еще выше.

Анализ влияния параметров метода

Используем датасет с данными о кредитах для анализа зависимости точности метода от его параметров ν и χ. На графиках линиями изображены средние значения функции качества на кросс-валидации. Прозрачные области поверх линий – среднеквадратичное отклонение качества.

Рассмотрим параметр доли кандидатов ν.

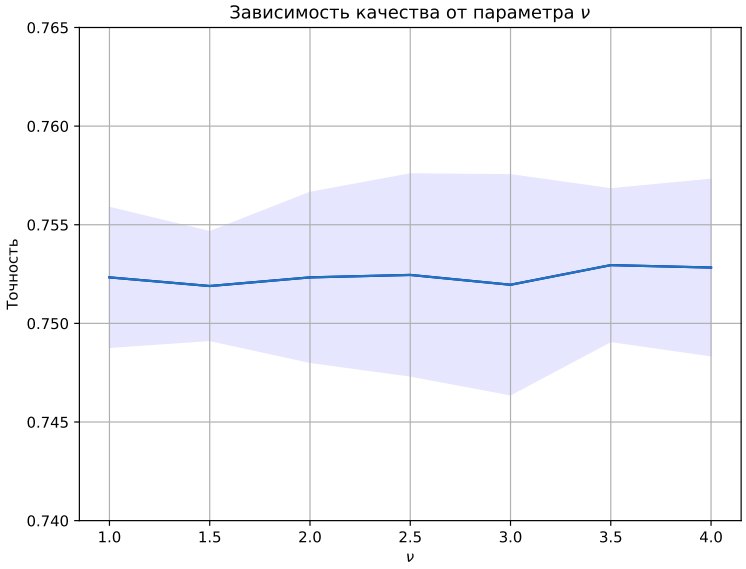


Рис. 3: Зависимость итогового качества от параметра метода ν

Видим, что с ростом параметра доли ν увеличивается среднее качество, но при этом растет и среднеквадратичное отклонение (рис. 3). Это закономерно, так как большее число кандидатов получается с помощью скрещивания и мутации, а значит в метод вносится дополнительная случайность.

Рассмотрим параметр квантиля значений гиперпараметров χ.

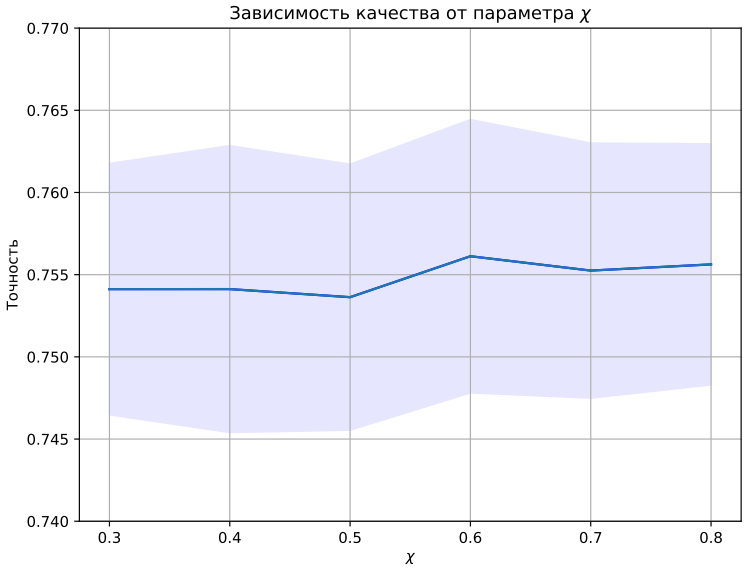


Рис. 4: Зависимость итогового качества от параметра метода χ

Видим, что с ростом квантиля χ среднее качество достигает максимума при значении 0.6, а среднеквадратичное отклонение монотонно уменьшается (рис. 4). Это происходит из-за того, что для настройки оценок плотностей используются только лучшие

значения параметров, а значит уменьшается случайность при мутации. При значении 0.6 достигается компромисс между переиспользованием результатов и случайностью сэмплирования.

Выводы

Предложенный метод показывает лучшее качество при оптимизации случайного леса и градиентного бустинга с высоким бюджетом вычислений. При малом бюджете и для метода опорных векторов лучше себя показали эволюционный алгоритм и TPE. Объединение идей HyperBand и эволюционных алгоритмов позволило использовать преимущества обоих алгоритмов и преодолеть их слабые стороны.

Модификация мутации позволяет предложенному методу достичь лучшего качества благодаря переходу от полной случайности в изменении параметров к контролируемой случайности в сторону более перспективных значений гиперпараметров. На отложенной тестовой выборке улучшение не так однозначно, но при использовании наборов данных большего объема разница между кросс-валидацией и отложенной выборкой была бы меньше.

Предложенный метод работает хуже, чем эволюционные алгоритмы или TPE в задачах, в которых вычислительный бюджет мал. Это связано с тем, что размеры популяций на итерациях HyperBand становятся слишком малы и метод начинает сильнее полагаться на меньшее число наборов с лучшим качеством при скрещивании.

Дальнейшими направлениями исследования могут быть изучение возможности изменения принципов скрещивания для учета значений оптимизируемой функции, более сложная модификация мутации, учитывающая связи между параметрами и сэмплирование из неисследованных участков пространства гиперпараметров.

Часть 2

В данной части рассматривается метод DEHB, работающий с непрерывными гиперпараметрами [20]. Предлагается модификация метода, проводятся эксперименты для сравнения результатов работы модифицированного метода и базового.

Алгоритм DEHB

В алгоритме DEHB сэмплирование из равномерного распределения производится один раз – сэмплируются выборки кандидатов для каждого уровня бюджета вычислений, далее лучшие кандидаты на каждом уровне бюджета сохраняются и передаются в следующие запуски SH. На каждом шаге SH производится модифицированная дифференциальная эволюция – текущие кандидаты, полученные из предыдущего запуска SH, скрещиваются с мутантами, полученными с помощью сэмплирования лучших кандидатов с предыдущей итерации текущего запуска SH [21].

Предлагаемая модификация метода DEHB

У алгоритма DEHB есть недостаток – он использует случайное сэмплирование всего один раз, и далее кандидаты меняются только за счет дифференциальной эволюции. Алгоритм часто застревает в локальном минимуме и выходит на плато (область примерного постоянства) по значению функции ошибок, что может быть связано с жадным переиспользованием кандидатов, лежащих на маленьком расстоянии друг от друга и имеющим близкие значения функции потерь. Чтобы этого избежать, предлагается добавлять шумовые векторы к векторам-«мутантам» при эволюции. Благодаря шумовым векторам увеличится широта перебора зависимости функции потерь от гиперпараметров. Это приводит к тому, что некоторые наборы гиперпараметров из популяций будут находиться вне локального минимума. При скрещивании с наборами вне локального минимума DEHB сможет найти новые оптимальные решения. DEHB создает вектор-“мутант” по следующей формуле: $v_{i,g} = x_{r_1,g} + F(x_{r_2,g} - x_{r_3,g})$, где r_1, r_2, r_3 – случайные кандидаты из текущей популяции, F – множитель, равный 0.5. Предлагается находить вектор-“мутант” по модифицированной формуле $v_{i,g} = x_{r_1,g} + F(x_{r_2,g} - x_{r_3,g}) + m(t)\varepsilon$, где $m(t) : \mathbb{R} \to \mathbb{R}$ – множитель при шумовом векторе. $m(t)$ может выбираться константой либо зависеть от номера итерации t и уменьшаться со временем, смещая поведение оптимизатора от исследования (exploration) к использованию (exploitation). Также он может зависеть от динамических условий оптимизации – например, увеличиваться после выхода на плато, что препятствует застреванию процесса оптимизации в локальном минимуме. Тщательный выбор функции $m(t)$ обеспечивает наилучшую производительность модифицированного метода оптимизации.

Зададим $m(t)$ следующим образом:

$$m(t) = \begin{cases} b + s, & \text{если модель находится на плато по ошибке} \\ b, & \text{иначе} \end{cases},$$

где b – базовое значение шума, s – шаг шума (параметры метода). Выход на плато определяется отсутствием улучшения качества в течение 10 итераций.

Результаты экспериментов

В качестве модели для оптимизации гиперпараметров взяты RandomForestClassifier и RandomForestRegressor из библиотеки scikit-learn. Параметры $b = 0, s = 0.1$. При увеличении параметра b шум добавляется слишком рано, и это уменьшает стабильность оптимизации. Большие значения параметра s также уменьшают стабильность, но иногда позволяют достичь лучшего качества моделей. Рассматривались следующие гиперпараметры для обоих алгоритмов:

- Доля признаков $\in [0.1, 0.9]$
- Минимальное количество объектов для разбиения $\in [2, 128]$
- Минимальное количество объектов в листе дерева $\in [1, 64]$

На графиках линиями изображены средние значения функции ошибки на валидационной выборке для итерации обучения на наборе гиперпараметров. Прозрачные области поверх линий – среднеквадратичное отклонение ошибки. По осям x – количество запусков дифференциальной эволюции для каждой популяции. Функции ошибки: 1-accuracy для классификации и MSE для регрессии. Метод тестируется на данных из библиотеки scikit-learn и наборах данных с сайта open-ml (таб. 9) [16, 19]:

Название	Тип	Описание
steel-plates-fault	Классификация	Брак стальных пластин
ilpd	Классификация	Детекция пациентов
scene	Классификация	Предсказание городской среды
spambase	Классификация	Детекция спама
California Housing	Регрессия	Стоимость недвижимости

Таблица 9: Наборы данных

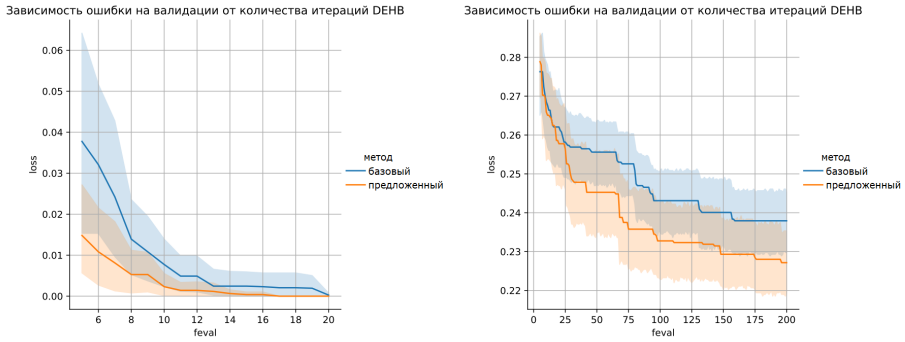


Рис. 5: Зависимость минимизируемой функции от номера итерации для задачи распознавания брака стальных пластин и задачи распознавания пациентов

Видим, что в задаче распознавания брака стальных пластин и в задаче распознавания пациентов с заболеваниями печени предложенный метод стабильно предсказывает лучшие наборы гиперпараметров, чем базовая стратегия мутации (рис. 5). Кроме того, даже несмотря на усреднение ошибок по различным запускам, невооруженным взглядом видно, что базовый метод без зашумления дольше выходит с каждого плато (“ступеньки” на синем графике длиннее).

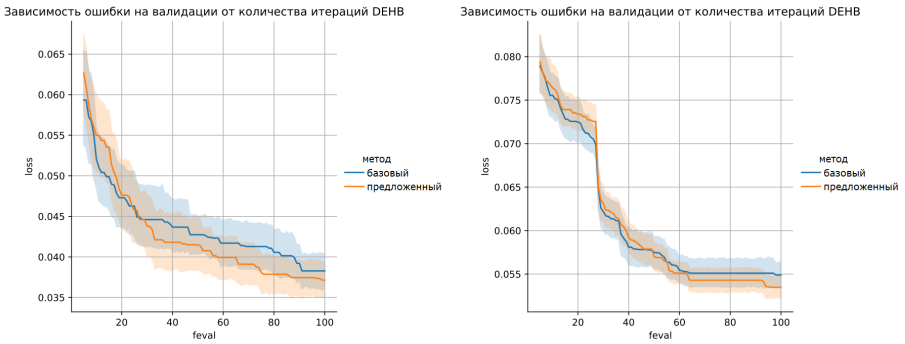


Рис. 6: Задача предсказания городской среды и задача распознавания спама

При тестировании метода на задаче предсказания городской среды по векторным представлениям картинки и задаче распознавания спама метод показывает менее стабильный прирост качества, однако можно заметить, что предложенный метод выходит с плато быстрее классического (рис. 6).

Сводные таблицы итоговых ошибок и их среднеквадратичных отклонений в методах приведены ниже:

Модель/Данные	1	2	3
DEHB	0.001 ± 0.005	$\mathbf{0.242 \pm 0.012}$	0.038 ± 0.006
Предложенный	$\mathbf{0.001 \pm 0.001}$	0.234 ± 0.014	$\mathbf{0.035 \pm 0.006}$

Таблица 10: Результаты экспериментов

Модель/Данные	4	5
DEHB	0.054 ± 0.005	0.553 ± 0.092
Предложенный	$\mathbf{0.051 \pm 0.005}$	$\mathbf{0.542 \pm 0.081}$

Таблица 11: Результаты экспериментов

Видим, что предложенный метод показывает улучшение на большинстве задач (таб. 10, таб. 11). Также стоит заметить, что параметры метода стоит подбирать в зависимости от размерности пространства гиперпараметров, так как шум одной и той же величины с увеличением размерности будет вносить больший вклад в мутацию.

Выводы

В работе рассматривалась задача автоматической настройки конфигурации моделей машинного обучения, задаваемой вектором гиперпараметров. Была предложена модификация операции мутации при генерации новых конфигураций гиперпараметров в базовом алгоритме DEHB за счет добавления дополнительного шума с изменяемой дисперсией в зависимости от выхода оптимизируемой функции на стабильный уровень значений (плато). Результаты экспериментов показали, что метод показывает улучшение сходимости и нахождение оптимума лучшего качества на большинстве задач. Добавление шума при выходе на плато позволяет получить новых кандидатов и продолжить подбор гиперпараметров с помощью дифференциальной эволюции.

Результаты работы

Метод, предложенный в первой части работы, позволяет находить оптимальные значения гиперпараметров в задачах с большим бюджетом вычислений. В среднем он ра-

ботает лучше HyperBand и базового эволюционного алгоритма. Исследована зависимость результатов работы предложенного метода от новых параметров.

Во второй части работы предложена модификация DEHB, исправляющая застревание метода в локальных минимумах функции ошибок. Также исследована зависимость результатов работы от введенных параметров. Вторая часть работы представлена на конференции ИТиММ-2024 в РЭУ имени Г.В. Плеханова в рамках доклада на секции 2: “Рандомизация поиска гиперпараметров в методе DEHB”, Абрамов В.А, Китов В.В.

Список литературы

1. Bergstra J., Bengio Y. Random search for hyper-parameter optimization //Journal of machine learning research. – 2012. – Т. 13. – №. 2.

2. Bischl B. et al. Hyperparameter optimization: Foundations, algorithms, best practices, and open challenges //Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery. – 2023. – Т. 13. – №. 2. – С. e1484.

3. Awad N., Mallik N., Hutter F. Dehb: Evolutionary hyperband for scalable, robust and efficient hyperparameter optimization //arXiv preprint arXiv:2105.09821. – 2021.

4. Talbi E. G. Optimization of deep neural networks: a survey and unified taxonomy. – 2020.

5. Hutter F., Hoos H. H., Leyton-Brown K. Sequential model-based optimization for general algorithm configuration //Learning and Intelligent Optimization: 5th International Conference, LION 5, Rome, Italy, January 17-21, 2011. Selected Papers 5. – Springer Berlin Heidelberg, 2011. – С. 507-523.

6. Wistuba M., Schilling N., Schmidt-Thieme L. Scalable gaussian process-based transfer surrogates for hyperparameter optimization //Machine Learning. – 2018. – Т. 107. – №. 1. – С. 43-78.

7. Bergstra J. et al. Algorithms for hyper-parameter optimization //Advances in neural information processing systems. – 2011. – Т. 24.

8. Lindauer M. et al. SMAC3: A versatile Bayesian optimization package for hyperparameter optimization //Journal of Machine Learning Research. – 2022. – Т. 23. – №. 54. – С. 1-9.

9. Schilling N., Wistuba M., Schmidt-Thieme L. Scalable hyperparameter optimization with products of gaussian process experts //Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2016, Riva del Garda, Italy, September 19-23, 2016, Proceedings, Part I 16. – Springer International Publishing, 2016. – С. 33-48.

10. Young S. R. et al. Optimizing deep learning hyper-parameters through an evolutionary algorithm //Proceedings of the workshop on machine learning in high-performance computing environments. – 2015. – С. 1-5.

11. Alibrahim H., Ludwig S. A. Hyperparameter optimization: Comparing genetic algorithm against grid search and bayesian optimization //2021 IEEE Congress on Evolutionary Computation (CEC). – IEEE, 2021. – С. 1551-1559.

12. Jamieson K., Talwalkar A. Non-stochastic best arm identification and hyperparameter optimization //Artificial intelligence and statistics. – PMLR, 2016. – С. 240-248.

13. Li L. et al. Hyperband: A novel bandit-based approach to hyperparameter optimization //Journal of Machine Learning Research. – 2018. – Т. 18. – №. 185. – С. 1-52.

14. Falkner S., Klein A., Hutter F. BOHB: Robust and efficient hyperparameter optimization at scale //International conference on machine learning. – PMLR, 2018. – С. 1437-1446.

15. Scott D. W. Multivariate density estimation: theory, practice, and visualization. – John Wiley & Sons, 2015.

16. Pedregosa F. et al. Scikit-learn: Machine learning in Python //the Journal of machine Learning research. – 2011. – Т. 12. – С. 2825-2830.

17. Akiba T. et al. Optuna: A next-generation hyperparameter optimization framework //Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining. – 2019. – С. 2623-2631.

18. Ke G. et al. Lightgbm: A highly efficient gradient boosting decision tree //Advances in neural information processing systems. – 2017. – Т. 30.

19. Vanschoren J. et al. OpenML: networked science in machine learning //ACM SIGKDD Explorations Newsletter. – 2014. – Т. 15. – №. 2. – С. 49-60.

20. Awad N., Mallik N., Hutter F. Dehb: Evolutionary hyperband for scalable, robust and efficient hyperparameter optimization //arXiv preprint arXiv:2105.09821. – 2021.

21. Storn R., Price K. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces //Journal of global optimization. – 1997. – Т. 11. – С. 341-359.

Приложение

Значения качества на отложенных тестовых выборках для всех задач.

Случайный лес в задачах регрессии:

Алгоритм	space_ga	topo_2_1	us_crime
HyperBand	0.0144 ± 0.0003	0.0008 ± 0.0000	0.0177 ± 0.0001
Эволюция	0.0149 ± 0.0006	0.0008 ± 0.0000	$\mathbf{0.0176 \pm 0.0001}$
EvoHyperBand	0.0148 ± 0.0006	0.0008 ± 0.0000	$\mathbf{0.0176 \pm 0.0001}$
EvoHyperBandMut	$\mathbf{0.0143 \pm 0.0000}$	0.0008 ± 0.0000	$\mathbf{0.0176 \pm 0.0001}$
TPE	0.0146 ± 0.0001	0.0008 ± 0.0000	0.0177 ± 0.0001

Таблица 12: MSE случайного леса на отложенной выборке (задача регрессии)

Метод опорных векторов в задачах классификации:

Алгоритм	ilpd	blood	kr-kp
HyperBand	0.6735 ± 0.1331	0.7487 ± 0.0291	0.8589 ± 0.0800
Эволюция	0.7299 ± 0.0134	0.7793 ± 0.0020	0.9556 ± 0.0069
EvoHyperBand	0.7034 ± 0.1243	0.7567 ± 0.0316	0.9352 ± 0.0232
EvoHyperBandMut	0.6427 ± 0.1708	0.7667 ± 0.0119	0.9334 ± 0.0333
TPE	$\mathbf{0.7350 \pm 0.0000}$	$\mathbf{0.7800 \pm 0.0000}$	$\mathbf{0.9609 \pm 0.0000}$

Таблица 13: Точность метода опорных векторов на отложенной выборке (задача классификации)

Градиентный бустинг с бюджетом 243 в задачах классификации:

Алгоритм	credit-g	ilpd	kr-kp
HyperBand	0.7740 ± 0.0097	0.7487 ± 0.0214	$\mathbf{0.9747 \pm 0.0072}$
Эволюция	0.7700 ± 0.0143	$\mathbf{0.7530 \pm 0.0150}$	0.9716 ± 0.0069
EvoHyperBand	0.7605 ± 0.0157	0.7462 ± 0.0094	0.9745 ± 0.0053
EvoHyperBandMut	0.7600 ± 0.0116	0.7496 ± 0.0195	0.9745 ± 0.0063
TPE	$\mathbf{0.7880 \pm 0.0040}$	0.7436 ± 0.0000	$\mathbf{0.9747 \pm 0.0015}$

Таблица 14: Точность градиентного бустинга на отложенной выборке (задача классификации)

Градиентный бустинг с бюджетом 500 в задачах регрессии:

Алгоритм	space_ga	topo_2_1	us_crime
HyperBand	0.0113 ± 0.0003	0.0008 ± 0.0000	0.0175 ± 0.0003
Эволюция	$\mathbf{0.0111 \pm 0.0004}$	0.0008 ± 0.0000	0.0174 ± 0.0003
EvoHyperBand	0.0112 ± 0.0003	0.0008 ± 0.0000	0.0175 ± 0.0004
EvoHyperBandMut	0.0115 ± 0.0004	0.0008 ± 0.0000	$\mathbf{0.0172 \pm 0.0007}$
TPE	0.0115 ± 0.0000	0.0008 ± 0.0000	0.0176 ± 0.0000

Таблица 15: MSE градиентного бустинга на отложенной выборке (задача регрессии)