УДК 004.05
DOI 10.17513/snt.39862

ГРАДИЕНТНЫЕ АЛГОРИТМЫ ОПТИМИЗАЦИИ В ПОДБОРЕ ГИПЕРПАРАМЕТРОВ МОДЕЛЕЙ МАШИННОГО ОБУЧЕНИЯ SCIKIT-LEARN

Парфентьев К.В., Пронин Д.Д., Борисов В.К., Кабаков В.А.

ФГБОУ ВО «Московский государственный технический университет имени Н.Э. Баумана», Москва, e-mail: bauman@bmstu.ru

В статье рассмотрена задача подбора гиперпараметров в рамках классического машинного обучения, кратко описаны наиболее распространенные методы подбора гиперпараметров и механизм их работы. Предложен алгоритм HOptim градиентной оптимизации гиперпараметров моделей классического машинного обучения из python-библиотеки scikit-learn, использующий в своей основе такие оптимизаторы первого порядка, как итерационный метод наискорейшего спуска, ускоренный градиент Нестерова, adagrad, RMSprop, adadelta, adam. Проведено тестирование алгоритма HOptim на модели GradientBoosting в задаче классификации на датасете Telco Customer Churn с предварительно удаленными категориальными признаками и регрессии на датасете House prices regression с предварительно удаленными категориальными признаками. Начальная инициализация гиперпараметров модели произведена значениями, равными единице, и значениями, подобранными с использованием алгоритма GridSearch CV, с целью последующего применения HOptim. Проведено тестирование связки алгоритма подбора гиперпараметров GridSearch CV и HOptim, проведен анализ эффективности ее работы. На графиках продемонстрирована поверхность ошибки модели GradientBoosting в зависимости от значений ее гиперпараметров max_depth, n_estimators в упомянутых задачах и кривые, демонстрирующие улучшение значений метрик качества в ходе работы HOptim.

Ключевые слова: классическое машинное обучение, оптимизация гиперпараметров, scikit-learn

GRADIENT-BASED OPTIMIZATION ALGORITHMS FOR HYPERPARAMETER TUNING OF SCIKIT-LEARN MACHINE LEARNING MODELS

Parfentev K.V., Pronin D.D., Borisov V.K., Kabakov V.A.

Bauman Moscow State Technical University, Moscow, e-mail: bauman@bmstu.ru

The paper considers the problem of hyperparameter selection in the framework of classical machine learning, briefly describes the most common methods of hyperparameter selection and the mechanism of their operation. The HOptim algorithm of gradient optimization of hyperparameters of classical machine learning models from python library scikit-learn is proposed, which uses such first-order optimizers as steepest descent method, Nesterov's accelerated gradient, adagrad, RMSprop, adadelta, adam. We tested the HOptim algorithm on the GradientBoosting model in the classification task on the Telco Customer Churn dataset with pre-deleted categorical features and regression on the House prices regression dataset with pre-deleted categorical features. Initialization of hyperparameters of the model was performed with values equal to one and values selected using GridSearch CV algorithm with the purpose of subsequent application of HOptim. Testing of the GridSearch CV and HOptim hyperparameter selection algorithm was carried out and its efficiency was analyzed. The graphs show the error surface of the GradientBoosting model depending on the values of its hyperparameters max_depth, n_estimators in the mentioned tasks and the curves showing the improvement of quality metrics values during HOptim operation.

Keywords: classical machine learning, hyperparameter optimization, scikit-learn

Гиперпараметры в классическом машинном обучении – параметры алгоритма, которые определяются не автоматически, во время обучения, а задаются исследователем непосредственно перед самим обучением модели и не меняются в его процессе. Подбор оптимальных гиперпараметров является важной частью процесса настройки алгоритма, так как они могут существенно повлиять на качество результатов во время тестирования и на скорость обучения.

Известно множество способов автоматического подбора гиперпараметров. Ниже приводятся примеры некоторых из них.

Самым естественным способом подбора гиперпараметров является перебор по сетке со скользящим контролем (GridSearch CV). В его процессе исследователем фиксируются несколько значений требуемых гиперпараметров, после чего алгоритм автоматически перебирает все их комбинации, при этом на каждой из них модель обучается и тестируется. По завершении указанного процесса, выбирается модель, показавшая лучшее качество. Главным недостатком указанного метода является асимптотическое время выполнения, так как в ходе его работы осуществляется прямой полный пе-

ребор гиперпараметров, сопровождающийся вычислительно трудоемким скользящим контролем (cross validation).

Если по какой-то причине исследователю требуется перебрать большое число гиперпараметров, то целесообразнее использовать случайный поиск (Random search) [1], в ходе которого для каждого из гиперпараметров выбирается значение из заданного наперед распределения, что помогает в первую очередь перебирать значения наиболее значимых гиперпараметров, которые сильно влияют на качество модели, таким образом с бóльшей вероятностью находить их удачную комбинацию.

Итеративные методы подбора гиперпараметров, основанные на байесовской оптимизации и их модернизации [2], в отличие от алгоритмов, которые были перечислены выше, тем или иным образом обращаются к результатам предыдущих итераций, но так же, как и методы, основанные на переборе, имеют высокую вычислительную сложность.

Альтернативным подходом к подбору гиперпараметров модели является оптимизация их значений посредством градиентного спуска. Существуют подходы к подбору гиперпараметров конкретных моделей, к примеру определенных архитектур нейросетей [3], или их оптимизаторов, но стоит отметить, что данный подход не получил распространения в рамках классического машинного обучения, несмотря на его большой потенциал.

Целью исследования алгоритмов подбора оптимальных гиперпараметров является повышение эффективности и точности моделей машинного обучения, улучшение их обобщающей способности, сокращение времени и ресурсов, затрачиваемых на обучение, а также упрощение и автоматизация процесса выбора оптимальных параметров.

В этой статье будет рассмотрен универсальный алгоритм подбора гиперпараметров для моделей классического машинного обучения, реализованных в библиотеке scikit-learn, основанный на градиентной оптимизации.

Материалы и методы исследования

Пусть определена модель M, процесс обучения и получаемые в его результате параметры которой детерминированы относительно начального набора ее гиперпараметров $h_1,h_2,\ldots,h_n$. Пусть $h_1,h_2,\ldots,h_n$ выражаются вещественными числами, а также задана некоторая метрика Q, по которой осуществляется оценивание качества работы модели M на этапе тестирования.

Необходимо разбить набор данных на обучающую и тестовую подвыборки и зафиксировать их, начать обучение модели M на обучающей подвыборке и оценить качество ее работы по метрике Q на тестовой подвыборке. Так как результаты обучения детерминированы относительно $h_1,h_2,\ldots,h_n$, то каждому набору $h_{1i},h_{2i},\ldots,h_{ni}$ будет соответствовать единственное значение метрики качества Q_i. Значит, можно рассмотреть Q как функцию от $h_1,h_2,\ldots,h_n$, такую, что $Q: h_1 \times h_2 \times \ldots \times h_n \to \mathbb{R}$.

Далее необходимо определить функцию $L(h_1,h_2,\ldots,h_n)$ таким образом, чтобы ее минимизация была эквивалентна задаче улучшения качества по метрике Q (максимизации или минимизации функции $Q(h_1,h_2,\ldots,h_n)$ в зависимости от решаемой задачи и выбранной метрики). Тогда задачу подбора оптимальных гиперпараметров $h_1^*, h_2^*, ..., h_n^*$ модели M можно представить следующим образом:

$$h_1^*, h_2^*, ..., h_n^* = \arg\min_{h_1,h_2,...,h_n}(L).$$

Затем предлагается воспользоваться градиентными алгоритмами оптимизации. Градиент в рассматриваемой задаче определяется следующим образом: $\nabla L = \sum_{i=1}^{n} \frac{dL}{dh_i}$.

Так как на практике большинство численных гиперпараметров классических моделей машинного обучения задается натуральными числами, можно рассматривать следующее численное приближение компонент градиента:

$$\frac{dL}{dh_i} \approx \frac{\Delta L}{\Delta h_i} = \frac{L(h_1,\ldots,h_i+1,...,h_n) - L(h_1,\ldots,h_i,...,h_n)}{1} = L(h_1,\ldots,h_i+1,...,h_n) - L(h_1,\ldots,h_i,...,h_n).$$

Для каждой из компонент градиента предлагается использовать коэффициент масштаба $s = b_h - a_h$, где b_h, a_h – границы заданного пользователем промежутка, на котором осуществляется поиск гиперпараметра h. Данный множитель добавляется с целью масштабирования значений компонент градиента, так как в практических задачах размеры промежутков, в которых с большой долей вероятности будут находиться оптимальные значения различных гиперпараметров, могут сильно отличаться (например, для модели GradientBoosting из scikit-learn гиперпараметр n_estimators на практике, с большой долей вероятности, будет лежать в отрезке [50; 300], а max_depth – в отрезке [2; 7]).

Тогда $\nabla_i L = (L(h_1, \ldots, h_i + 1, ..., h_n) - L(h_1, \ldots, h_i, ..., h_n)) \cdot s_i$.

Пусть $h = h_1,h_2,\ldots,h_n$, а $s = h_{1i},h_{2i},\ldots,h_{ni}$.

Градиенты будут рассчитываться следующим образом, с учетом применения коэффициента масштаба s:

1) Классический градиент:

$$g_t = \nabla L_t$$
$$h_{t+1} = h_t - lr \cdot g_t .$$

Здесь и далее: lr– гиперпараметр оптимизатора, выбираемый пользователем;

t – номер шага спуска.

2) Ускоренный градиент Нестерова [4]:

$$v_t = \gamma v_{t-1} + lr \cdot \nabla L(h_{t-1} - \gamma v_{t-1})$$
$$h_{t+1} = h_t - lr \cdot v_t .$$

3) Adagrad [5]:

$$g_t = \nabla L_t$$
$$G_t = G_{t-1} + g_{t-1}^2$$
$$h_{t+1} = h_t - \frac{lr}{\sqrt{G_t + \varepsilon}} \cdot g_t .$$

Здесь и далее: γ – гиперпараметр оптимизатора, выбираемый пользователем,

$\varepsilon \ll 1$ – наперед заданное малое число, вводимое во избежание возможного деления на 0.

4) RMSprop [6]:

$$g_t = \nabla L_t$$
$$E[g^2]_t = \gamma E[g^2]_{t-1} + (1-\gamma) g_t^2$$
$$h_{t+1} = h_t - \frac{lr}{\sqrt{E[g^2]_t + \varepsilon}} \cdot g_t .$$

5) Adadelta [7]:

$$g_t = \nabla L_t$$
$$E[g^2]_t = \gamma E[g^2]_{t-1} + (1-\gamma) g_t^2$$
$$\Delta h_t = -\frac{\sqrt{E[\Delta h^2]_{t-1} + \varepsilon}}{\sqrt{E[g^2]_t + \varepsilon}} g_t$$
$$E[\Delta h^2]_t = \gamma E[\Delta h^2]_{t-1} + (1-\gamma) \Delta h_t^2$$
$$h_{t+1} = h_t - \frac{\sqrt{E[\Delta h^2]_t + \varepsilon}}{\sqrt{E[g^2]_t + \varepsilon}} \cdot g .$$

6) Adam [8]:

$$g_t = \nabla L_t$$
$$m_t = \frac{\beta_1 m_{t-1} + (1-\beta_1) g_t}{1-\beta_1^t}$$
$$v_t = \frac{\beta_2 v_{t-1} + (1-\beta_2) g_t^2}{1-\beta_2^t}$$
$$h_{t+1} = h_t - \frac{lr}{\sqrt{v_t + \varepsilon}} m_t ,$$

где β_1, β_2 – гиперпараметры оптимизатора, выбираемые пользователем.

Перед началом работы алгоритма HOptim набор данных разделяется на три части: тренировочная, валидационная и тестовая. На каждой итерации алгоритм оптимизации гиперпараметров обучает полученную модель M с гиперпараметрами $h_{1t},h_{2t},\ldots,h_{nt}$, где t – номер шага, на тренировочной подвыборке, вычисляет компоненты градиента на валидационной подборке, делая предсказание с помощью обученной ранее модели и изменяет гиперпараметры, совершая шаг градиентного спуска. Параллельно с этим вычисляется качество модели на тестовой подвыборке, набор гиперпараметров, обеспечивший лучшие результаты на этапе тестирования, сохраняется.

Подобный подход к вычислению компонент градиента позволяет алгоритму не переобучить модель M путем увеличения ее сложности за счет гиперпараметров, а подобрать такой их набор, который позволит максимизировать качество работы на данных, недоступных в процессе обучения самой модели M, а следовательно, улучшить ее качество на этапе тестирования.

В некоторых случаях предлагается запоминать значения компонент градиентов с целью ускорения работы алгоритма в условии возникновения циклов (для некоторых оптимизаторов это может быть полезно, так как, например, Adam способен через некоторое число итераций выходить из подобных циклов, накапливая «энергию»).

Результаты исследования и их обсуждение

Предложенный метод проходит испытание на классических задачах классификации и регрессии.

Эксперимент: классификация.

В качестве набора данных для классификации используется Telco Customer Churn с предварительно удаленными столбцами, содержащими категориальные признаки. В качестве модели для классификации применена модель scikit-learn GradientBoostingClassifier. Функция, по которой осуществляется спуск, задается следующим образом:

$$L = 1 - Q \text{ (accuracy loss)},$$

где Q – оценка качества работы модели по метрике accuracy score. Здесь и далее в столбец skip записывается количество компонент, вычисления которых удалось сэкономить благодаря их записи в память, а в столбец not skip – количество вычисленных компонент. Этот и следующие эксперименты проводятся на процессоре Intel Core i5.

Классификация: плохие начальные гиперпараметры.

В этом эксперименте начальные значения гиперпараметров приняты равными единице.

Варьируемые гиперпараметры: max_depth, n_estimators.

Максимальное число шагов: 50.

Начальный accuracy score модели: 0,725

Результаты применения HOptim показаны в табл. 1 и на рис. 1, 2.

Таблица 1

Результаты применения HOptim

Оптимизатор	Лучший acc. score	Время, с	*lr*	γ	β_1	β_2	skip	not skip
Класс. град.	0,797	**3,60**	5	нет	нет	нет	**135**	16
Г. Нестерова	0,799	96,2	10	0,7	нет	нет	114	37
Adagrad	0,803	58,60	0,4	нет	нет	нет	113	38
RMSprop	**0,804**	145,50	0,2	0,7	нет	нет	74	77
Adadelta	0,796	25,20	нет	0,4	нет	нет	125	26
Adam	**0,804**	152,20	0,3	нет	0,4	0,6	68	83

Источник: составлено авторами.

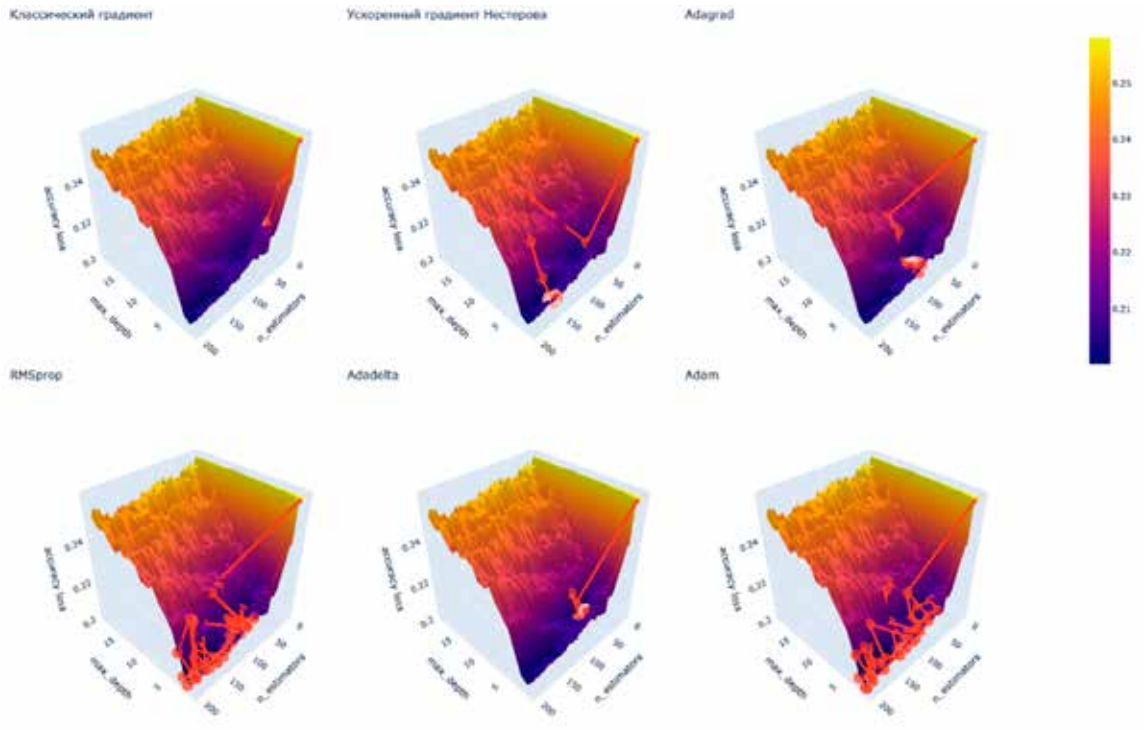

Рис. 1. Поверхность функции потерь для набора данных Telco Customer Churn и модели GradientBoostingClassifier – работа оптимизаторов
Источник: составлено авторами

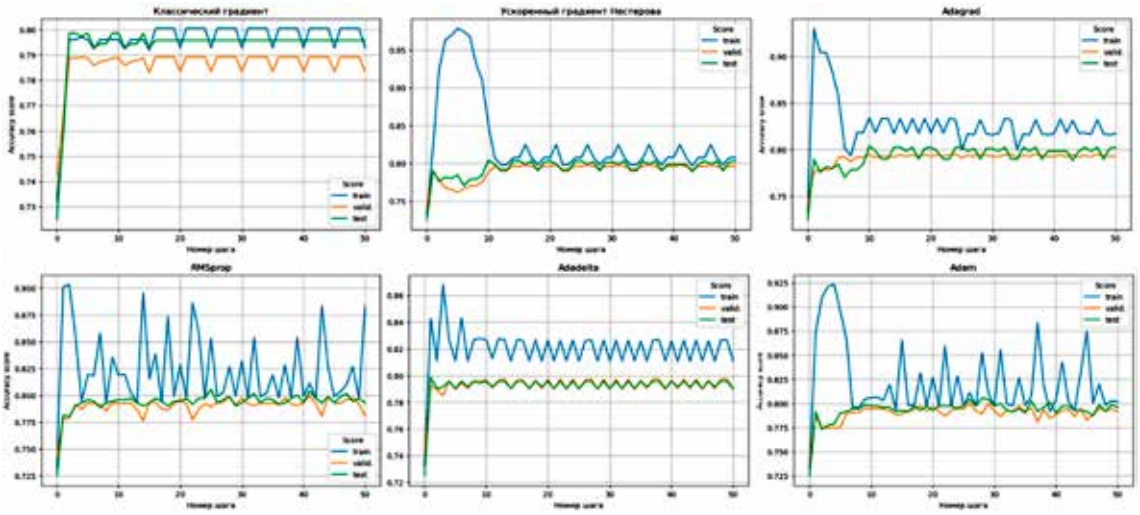

Рис. 2. Accuracy score модели GradientBoostingClassifier в зависимости от шага и выбранного оптимизатора
Источник: составлено авторами

Результаты, отраженные в табл. 1, показывают, что оптимизаторы RMSprop и Adam наиболее качественно справляются с задачей. В дополнение можно отметить, что эти оптимизаторы не попадают в цикл, что позволяет им перебрать большее количество комбинаций гиперпараметров (рис. 1, 2).

Классификация: GridSearch CV + HOptim

Перед использованием HOptim гиперпараметры были предварительно подобраны с помощью GridSearch CV.

Варьируемые гиперпараметры: max_ depth, n_estimators, min_samples_split, min_ samples_ leaf.

Максимальное число шагов: 50.

Начальный accuracy score модели: 0,795

Результаты применения HOptim показаны в табл. 2 и на рис. 3.

Видно, что в этой задаче результаты работы алгоритма HOptim с неудачной стартовой позиции и с более удачной позиции, найденной с помощью GridSearch CV, различаются слабо, но стоит отметить, что лучший результат в ходе эксперимента достигнут ускоренным градиентом Нестерова в связке с GridSearch CV (табл. 2 и рис. 3).

Эксперимент: регрессия.

Рассматривается применение предложенного алгоритма на примере задачи регрессии.

В качестве набора данных для регрессии используется House prices regression с предварительно удаленными столбцами, содержащими категориальные признаки. В качестве модели для регрессии применяется модель skelearn GradientBoostingRegressor. Функция, по которой осуществляется спуск, задается следующим образом: $L = \log_{10}(1+Q)$ (log MSE loss), где Q – среднеквадратичная ошибка модели.

Регрессия: плохие начальные гиперпараметры.

В этом эксперименте начальные значения гиперпараметров приняты равными единице.

Таблица 2

Результаты применения HOptim

Оптимизатор	Лучший acc. score	Время, с	*lr*	γ	β_1	β_2	skip	not skip
Класс. град.	0,797	**71,6**	10	нет	нет	нет	**177**	74
Г. Нестерова	**0,809**	196,8	10	0,95	нет	нет	76	175
Adagrad	0,8043	263,3	0,5	нет	нет	нет	63	188
RMSprop	0,803	324,7	0,1	0,8	нет	нет	51	200
Adadelta	0,804	309,2	нет	0,6	нет	нет	60	191
Adam	0,8014	241,2	0,1	нет	0,6	0,7	50	201

Источник: составлено авторами.

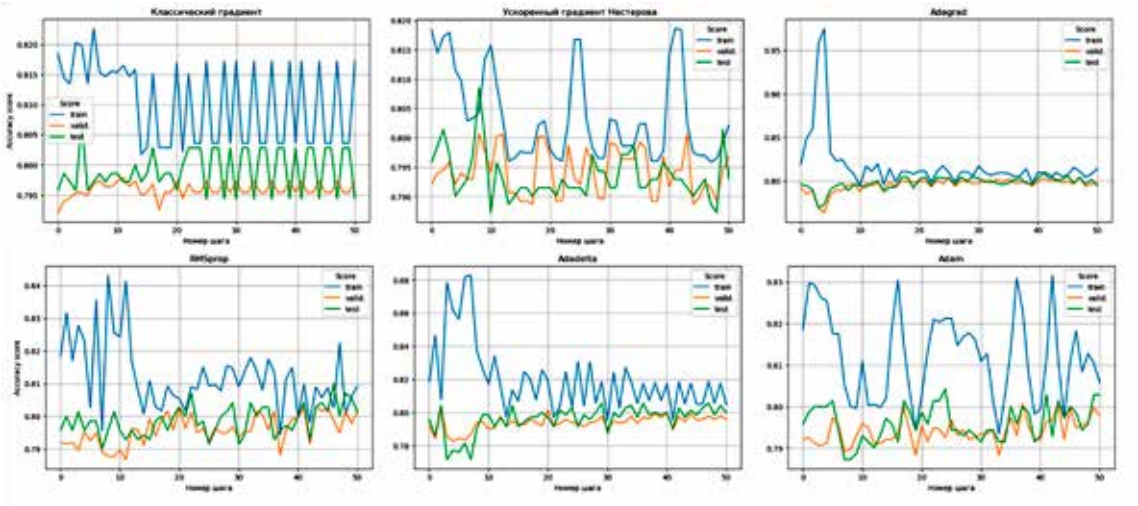

Рис. 3. Accuracy score модели GradientBoostingClassifier в зависимости от шага и выбранного оптимизатора. Источник: составлено авторами

Таблица 3

Результаты применения HOptim

Оптимизатор	Лучший log MSE loss	Время, с	*lr*	γ	β_1	β_2	skip	not skip
Класс. град.	9,075	17,10	2	нет	нет	нет	78	73
Г. Нестерова	**9,000**	65,4	2	0,85	нет	нет	66	85
Adagrad	9,001	21,00	0,4	нет	нет	нет	97	54
RMSprop	**9,000**	39,90	0,2	0,5	нет	нет	82	69
Adadelta	9,045	**9,70**	нет	0,35	нет	нет	114	37
Adam	8,996	46,40	0,3	нет	0,4	0,6	66	85

Источник: составлено авторами.

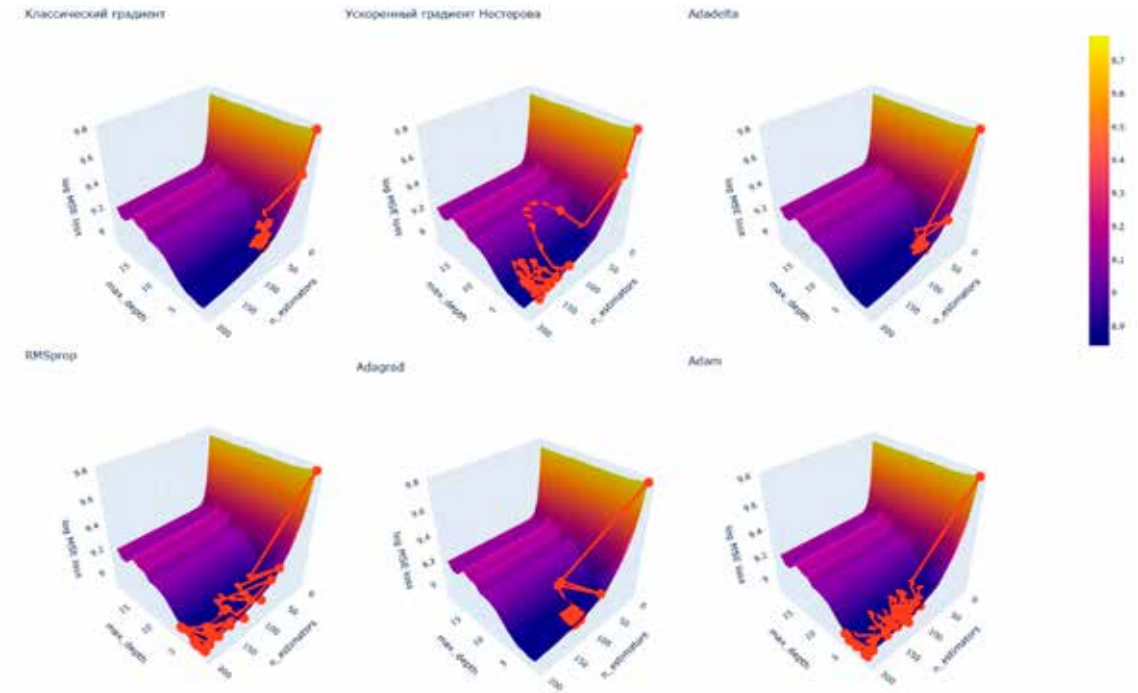

Рис. 4. Поверхности функции потерь для набора данных House prices regression и модели GradientBoostingRegressor Источник: составлено авторами

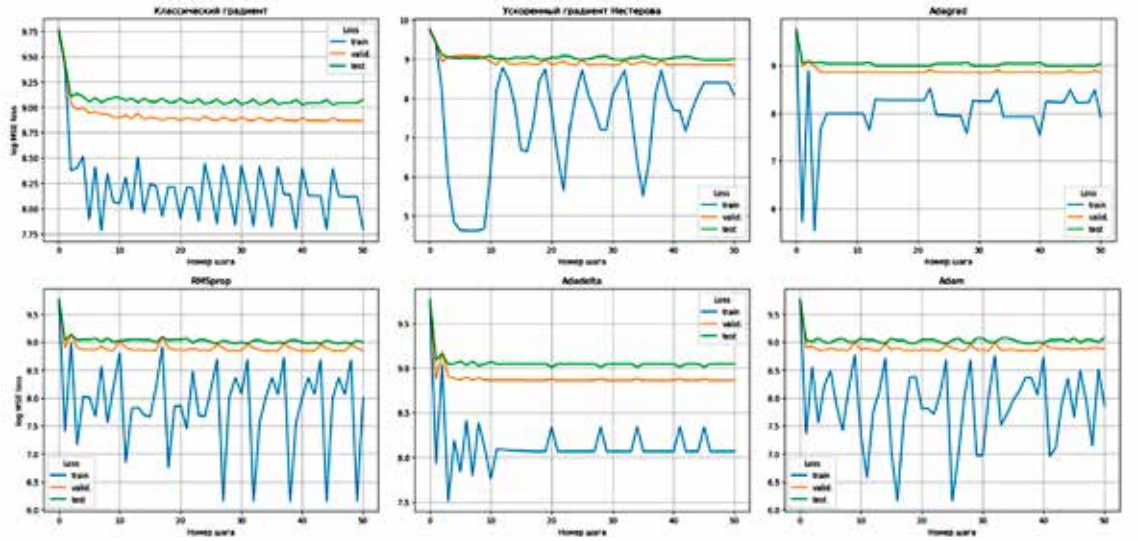

Рис. 5. Log MSE loss модели GradientBoostingClassifier в зависимости от шага и выбранного оптимизатора
Источник: составлено авторами

Варьируемые гиперпараметры: max_depth, n_estimators.

Максимальное число шагов: 50.

Начальный log MSE loss модели: 9,771

Результаты применения HOptim показаны в табл. 3 и на рис. 4, 5.

Результаты, отраженные в табл. 3, показывают, что градиент Нестерова и RMSprop наиболее качественно справляются с задачей, минимизируя функцию потерь лучше других оптимизаторов.

Важно отметить, что градиент Нестерова более подвижен, чем другие оптимизаторы в этой задаче, что позволяет ему осуществить оценку большего количества комбинаций гиперпараметров, что наглядно представлено на рис. 4, 5.

Регрессия: GridSearch CV + HOptim

Перед использованием HOptim гиперпараметры были предварительно подобраны с помощью GridSearch CV.

Варьируемые гиперпараметры: max_depth, n_estimators, min_samples_split, min_samples_leaf.

Максимальное число шагов: 50.

Начальный log MSE loss модели: 9,278

Результаты применения HOptim показаны в табл. 4 и на рис. 6.

В описанном эксперименте наилучшие гиперпараметры найдены алгоритмом HOptim без использования GridSearch CV (табл. 4 и рис. 6).

В большинстве случаев целесообразно сначала использовать GridSearch CV или любой другой алгоритм подбора гиперпараметров с большим шагом, после чего использовать HOptim с малым коэффициентом *lr*. В случае, если решение, найденное GridSearch CV, будет находиться рядом с минимумом функции потерь, то HOptim позволит улучшить результат, осуществляя поиск в окрестности найденного решения.

Таблица 4

Результаты применения HOptim

Оптимизатор	Лучший log MSE loss	Время, с	*lr*	γ	β_1	β_2	skip	not skip
Класс. град.	9,198	119,8	2	нет	нет	нет	109	142
Г. Нестерова	9,095	145,2	0,7	0,8	нет	нет	**144**	107
Adagrad	9,108	142,4	0.1	нет	нет	нет	54	197
RMSprop	9,1922	219,2	0.1	0.8	нет	нет	50	201
Adadelta	9,1778	234,7	нет	0.6	нет	нет	51	200
Adam	**9,06137**	**92,2**	0.1	нет	0.6	0.7	99	152

Источник: составлено авторами.

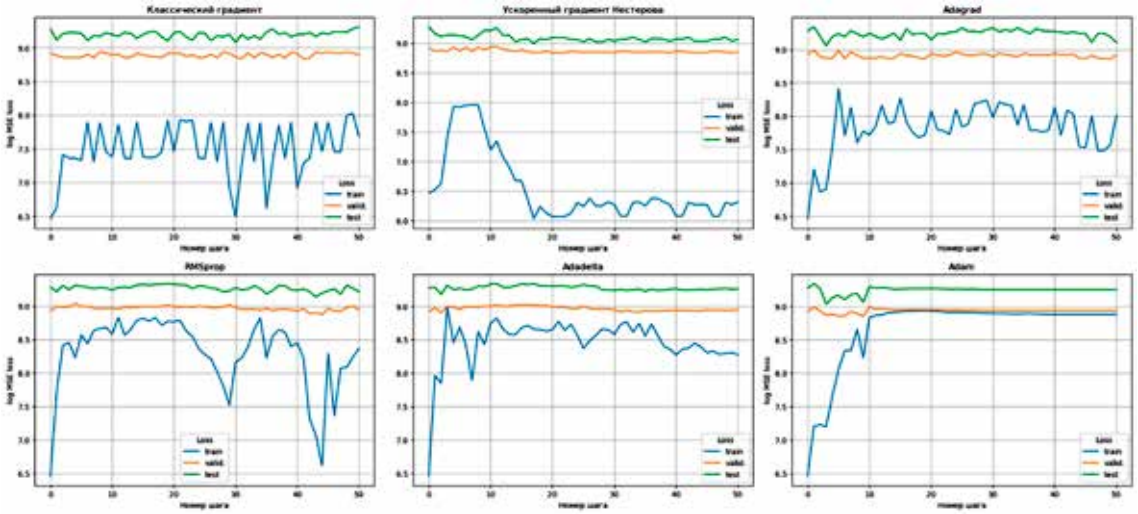

Рис. 6. Log MSE loss модели GradientBoostingClassifier в зависимости от шага и выбранного оптимизатора
Источник: составлено авторами

Заключение

В статье рассмотрена задача оптимизации гиперпараметров в рамках классического машинного обучения, кратко описаны наиболее распространенные методы подбора гиперпараметров и механизм их работы. Предложен алгоритм HOptim градиентной оптимизации гиперпараметров моделей классического машинного обучения из python-библиотеки scikit-learn, использующий в своей основе оптимизаторы первого порядка. В экспериментах показана эффективность метода HOptim и результативность его совместной работы с алгоритмом подбора гиперпараметров GridSearch CV с большим шагом сетки.

Описанный подход универсален и может быть использован на практике для улучшения качества работы моделей машинного обучения. Подбор оптимальных гиперпараметров позволяет увеличить точность предсказаний моделей. Это особенно важно в задачах классификации и регрессии, где точность работы моделей играет решающую роль.

Список литературы

1. Bergstra J., Bengio Y. Random Search for Hyper-Parameter Optimization // Journal of machine learning research. 2012. Vol. 13, Is. 2. P. 281−305. DOI: 10.5555/2503308.2188395.

2. Смирнова В.С., Шаламов В.В., Ефимова В.А., Фильченков А.А. Оптимизация гиперпараметров на основе объединения априорных и апостериорных знаний о задаче классификации // Научно-технический вестник информационных технологий, механики и оптики. 2020. № 6. С. 828–834.

3. Luo R., Tian F., Qin T. Neural architecture optimization // Advances in neural information processing systems. 2018. Vol. 31, Is. 1. P. 7827–7838.

4. Dozat T. Incorporating Nesterov momentum into Adam // ICLR. 2016. [Электронный ресурс]. URL: https://openreview.net/pdf?id=OM0jvwB8jIp57ZJjtNEZ (дата обращения: 27.10.2023).

5. Duchi J., Hazan E., Singer Y. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization // Journal of machine learning research. 2011. Vol. 12, Is. 7. P. 2121–2159. DOI: 10.5555/1953048.2021068.

6. Dauphin Y., De Vries H., Bengio Y. Equilibrated adaptive learning rates for non-convex optimization // Advances in neural information processing systems. 2015. Vol. 28, Is. 1. P. 1504–1512.

7. Matthew D. Zeiler. Adadelta: an adaptive learning rate method // arXiv preprint. [Электронный ресурс]. URL: https://arxiv.org/pdf/1212.5701.pdf (дата обращения: 27.10.2023).

8. Diederik P. Kingma Jimmy Lei Ba. Adam: a method for stochastic optimization // arXiv preprint. [Электронный ресурс]. URL: https://arxiv.org/pdf/1412.6980.pdf (дата обращения: 27.10.2023).