Содержание

От издательства

Отзывы и пожелания

Мы всегда рады отзывам наших читателей. Расскажите нам, что вы думаете об этой книге – что понравилось или, может быть, не понравилось. Отзывы важны для нас, чтобы выпускать книги, которые будут для вас максимально полезны.

Вы можете написать отзыв на нашем сайте www.dmkpress.com, зайдя на страницу книги и оставив комментарий в разделе «Отзывы и рецензии». Также можно послать письмо главному редактору по адресу dmkpress@gmail.com; при этом укажите название книги в теме письма.

Если вы являетесь экспертом в какой-либо области и заинтересованы в написании новой книги, заполните форму на нашем сайте по адресу http://dmkpress.com/authors/publish_book/ или напишите в издательство по адресу dmkpress@gmail.com.

Список опечаток

Хотя мы приняли все возможные меры для того, чтобы обеспечить высокое качество наших текстов, ошибки все равно случаются. Если вы найдете ошибку в одной из наших книг, мы будем очень благодарны, если вы сообщите о ней главному редактору по адресу dmkpress@gmail.com. Сделав это, вы избавите других читателей от недопонимания и поможете нам улучшить последующие издания этой книги.

Нарушение авторских прав

Пиратство в интернете по-прежнему остается насущной проблемой. Издательство «ДМК Пресс» очень серьезно относится к вопросам защиты авторских прав и лицензирования. Если вы столкнетесь в интернете с незаконной публикацией какой-либо из наших книг, пожалуйста, пришлите нам ссылку на интернет-ресурс, чтобы мы могли применить санкции.

Ссылку на подозрительные материалы можно прислать по адресу электронной почты dmkpress@gmail.com.

Мы высоко ценим любую помощь по защите наших авторов, благодаря которой мы можем предоставлять вам качественные материалы.

Предисловие

«Я хотел бы использовать машинное обучение, но не могу уделять ему много времени». Мы слишком часто слышим это от самых разных людей, от инженеров до исследователей, работающих в других областях. Им нужны готовые решения для машинного обучения, не требующие трудоемкой подготовки и отладки. В ответ на возникший спрос появилась новая отрасль *автоматизированного машинного обучения* (automated machine learning, AutoML), и я рад, что у читателей будет первое исчерпывающее руководство в этой области.

Я сам очень увлечен автоматизацией машинного обучения с тех пор, как в 2014 г. стартовал наш проект Automatic Statistician. Я хочу, чтобы все исследователи AutoML стремились к амбициозной цели: мы должны попытаться автоматизировать все аспекты полного конвейера машинного обучения и анализа данных. Сюда входит автоматизация сбора данных и организации экспериментов; очистки данных и подстановки недостающих данных; выбора и преобразования признаков; исследования, критического оценивания и объяснения моделей; распределения вычислительных ресурсов; оптимизации гиперпараметров; вывода; мониторинга моделей и обнаружения аномалий. Это очень обширный список, и в идеале следует автоматизировать все, что в нем перечислено.

Конечно, здесь нужно сделать оговорку. Хотя полная автоматизация служит хорошей мотивацией для исследователей и долгосрочной целью для инженеров, на практике пользователи предпочитают *полуавтоматизацию* и постепенное выведение человека из цикла машинного обучения по мере необходимости. Но выгода от автоматизации заключается не только в исключении человеческого труда. Продвигаясь к полной автоматизации, мы попутно разрабатываем мощные инструменты, которые помогут сделать практику машинного обучения прежде всего более систематичной (поскольку в наши дни она очень ситуативна и хаотична), а также более эффективной.

Это достойные цели, даже если мы не преуспеем в полной автоматизации, но, как показывает эта книга, текущие методы AutoML уже превосходят человеческих экспертов машинного обучения в ряде задач. Эта тенденция, вероятно, будет только усиливаться по мере нашего прогресса и по мере того, как вычисления становятся все дешевле, и поэтому AutoML явно является одной из «горячих» тем. Сейчас самое время заняться AutoML, и данная книга – отличная отправная точка.

Книга содержит актуальные обзоры основных методов, необходимых в AutoML (оптимизация гиперпараметров, метаобучение и поиск нейронной архитектуры), подробное обсуждение существующих систем AutoML и де-

тальную оценку состояния дел в AutoML на основе серии конкурсов, которые проводятся с 2015 г. Поэтому я настоятельно рекомендую эту книгу каждому исследователю машинного обучения, желающему начать работу в этой области, и каждому практикующему специалисту, желающему понять методы, лежащие в основе всех существующих инструментов AutoML.

Зубин Гахрамани
Профессор Кембриджского университета
и главный научный сотрудник Uber
Сан-Франциско, США
Октябрь 2018 г.

Введение

В последнее десятилетие наблюдается бурный рост количества исследований и применений машинного обучения; в частности, методы глубокого обучения позволили добиться значительных успехов во многих прикладных областях, таких как компьютерное зрение, обработка речи и игры. Однако прикладные решения в области машинного обучения чрезвычайно чувствительны к многочисленным нюансам реализации проектов, что служит серьезным препятствием для новых пользователей. Это особенно актуально для бурно развивающейся области глубокого обучения, где разработчикам приложений приходится выбирать правильные нейронные архитектуры, процедуры обучения, методы регуляризации и гиперпараметры всех этих компонентов, чтобы заставить свои сети делать то, чего от них ждут, с достаточной производительностью[1]. Процесс подбора подходящей модели машинного обучения приходится повторять для каждого приложения. Даже опытные специалисты часто бывают вынуждены проходить через утомительную серию проб и ошибок, пока не найдут удачный вариант для конкретного набора данных.

Область автоматизированного машинного обучения нацелена на принятие конструкторских решений на основе данных объективным и автоматизированным способом: пользователь просто предоставляет данные, а система AutoML автоматически находит оптимальное решение для этого конкретного случая. Таким образом, AutoML делает машинное обучение доступным для тех, кто не имеет возможности детально изучать технологии, лежащие в его основе. Это можно рассматривать как *демократизацию* машинного обучения: с AutoML современное машинное обучение доступно каждому.

Как мы покажем в этой книге, подходы AutoML уже достаточно развиты, чтобы соперничать, а иногда и превосходить экспертов в области машинного обучения. Проще говоря, AutoML может привести к повышению производительности, экономя при этом значительное количество времени и денег, поскольку хороших экспертов в области машинного обучения трудно найти, и их услуги дорого стоят. В результате в последние годы интерес

---

[1] Термин *performance*, который в IT-литературе обычно переводят как «производительность», на самом деле очень многозначен. В машинном обучении в зависимости от контекста он может означать точность модели, ее быстродействие, стабильность работы, а иногда все одновременно, т. е. совокупный уровень качества. В этой книге мы тоже используем перевод «производительность», подразумевая под ним свойства модели, зависящие от контекста. – *Прим. перев.*

инвесторов к AutoML резко возрос, и несколько крупных технологических компаний сейчас разрабатывает собственные системы AutoML. Однако стоит отметить, что цели демократизации машинного обучения гораздо лучше служат системы AutoML с открытым исходным кодом, чем платные черные ящики.

Эта книга представляет собой обзор быстро развивающейся области AutoML. В связи с тем, что в настоящее время научное сообщество сосредоточено на глубоком обучении, некоторые исследователи ошибочно приравнивают AutoML к *поиску нейронной архитектуры* (neural architecture search, NAS); но вы наверняка знаете, что, хотя NAS является отличным примером AutoML, понятие AutoML намного шире, чем NAS. Цель этой книги – предоставить исходные данные и отправные точки для исследователей, заинтересованных в разработке собственных подходов к AutoML, рассказать о доступных системах практикам, которые хотят применить AutoML для решения своих задач, и предоставить обзор состояния дел исследователям, уже работающим в AutoML. Книга разделена на три части, посвященные этим различным аспектам AutoML.

Часть I представляет собой обзор методов AutoML. Она содержит масштабный обзор для новичков и может служить справочником для опытных исследователей AutoML.

В главе 1 рассмотрена задача оптимизации гиперпараметров – простейшая и наиболее распространенная задача, которую решает AutoML, – и описан широкий спектр различных подходов с особым акцентом на методы, которые в настоящее время являются наиболее эффективными.

В главе 2 показано, как *научить модель учиться*, т. е. как использовать полученный ранее опыт оценки моделей машинного обучения при решении новых задач обучения с новыми данными. Такие методы имитируют процесс развития навыков человека от новичка к эксперту и могут значительно сократить время, необходимое для получения хороших результатов при решении совершенно новых задач машинного обучения.

В главе 3 представлен полный обзор методов для NAS. Это одна из самых сложных задач в AutoML, поскольку пространство проектирования чрезвычайно велико, а одна оценка нейронной сети может занять очень много времени. Тем не менее в этой области ведутся активные исследования, и регулярно появляются новые интересные подходы к решению задачи NAS.

Часть II посвящена реальным системам AutoML, которые могут применять даже начинающие пользователи. Если вы заинтересованы в применении AutoML для решения задач машинного обучения, вам следует начать именно с этой части. Все главы этой части подробно описывают представленные в них системы, чтобы дать представление об их эффективности на практике.

В главе 4 описана Auto-WEKA, одна из первых систем AutoML. Она основана на хорошо известном наборе инструментов машинного обучения WEKA и перебирает различные методы классификации и регрессии, настройки их гиперпараметров и методы предварительной обработки данных. Все это

доступно через графический пользовательский интерфейс WEKA одним нажатием кнопки, без необходимости написания кода.

В главе 5 представлен обзор Hyperopt-Sklearn – фреймворка AutoML, основанного на популярном фреймворке scikit-learn. Глава также содержит несколько практических примеров использования системы.

В главе 6 описан фреймворк Auto-sklearn, который также основан на scikit-learn. В нем применяются те же методы оптимизации, что и в Auto-WEKA, и добавлено несколько улучшений по сравнению с другими системами того времени, например метаобучение для теплого старта оптимизации и автоматическое ансамблирование. В главе сравнивается производительность Auto-sklearn с производительностью двух систем из предыдущих глав, Auto-WEKA и Hyperopt-Sklearn. К слову, Auto-sklearn – это система, которая победила в испытаниях, описанных в части III данной книги.

В главе 7 представлен обзор Auto-Net – системы автоматического глубокого обучения, которая выбирает архитектуру и гиперпараметры глубоких нейронных сетей. Даже начальная версия Auto-Net позволила создать первую автоматически настраиваемую нейронную сеть, которая победила в соревновании с экспертами-людьми.

В главе 8 описана система TPOT, которая автоматически строит и оптимизирует древовидные конвейеры машинного обучения. Эти конвейеры более гибкие, чем подходы, которые рассматривают только набор фиксированных компонентов машинного обучения, соединенных заранее определенными способами.

В главе 9 описана система Automatic Statistician, позволяющая автоматизировать науку о данных путем создания полностью автоматизированных отчетов, включающих анализ данных, а также прогностические модели и сравнение их эффективности. Уникальной особенностью Automatic Statistician является то, что она предоставляет описания результатов на естественном языке, подходящие для неспециалистов в области машинного обучения.

Наконец, в части III и главе 10 представлен обзор задач в области AutoML, над которыми исследователи работают с 2015 г. Назначение этого обзора – стимулировать разработку методов, которые хорошо справляются с практическими задачами, и определить лучший подход. В главе подробно описаны идеи и концепции, лежащие в основе текущих исследовательских задач, а также результаты предыдущих исследований.

Насколько нам известно, это первый разносторонний анализ всех аспектов AutoML: методов, лежащих в его основе, доступных систем, реализующих AutoML на практике, и задач для их оценки. Эта книга содержит справочную информацию, достаточную для начала разработки собственных систем AutoML, а также подробно описывает существующие системы, которые можно непосредственно применить для решения широкого круга задач машинного обучения. Область AutoML стремительно развивается, поэтому мы постарались объяснить и систематизировать последние достижения. Мы надеемся, что вам понравится эта книга и вы присоединитесь к растущему сообществу энтузиастов AutoML.

Благодарности

Мы хотим поблагодарить всех авторов глав, благодаря которым удалось издать эту книгу. Мы также благодарны программе исследований и инноваций Horizon 2020 Европейского союза за покрытие расходов на издание этой книги.

Франк Хуттер
Фрайбург, Германия

Ларс Коттхофф
Ларами, штат Вайоминг, США

Хоакин Ваншорен
Эйндховен, Нидерланды

Октябрь 2018 г.

Часть I

МЕТОДЫ AutoML

Глава 1

Оптимизация гиперпараметров

Маттиас Фойрер (✉), факультет компьютерных наук, Университет Фрайбурга, Фрайбург, Баден-Вюртемберг, Германия, e-mail: feurerm@informatik.uni-freiburg.de

Франк Хуттер, факультет компьютерных наук, Университет Фрайбурга, Фрайбург, Германия

Растущий интерес к сложным и вычислительно дорогим моделям машинного обучения с большим количеством гиперпараметров, таким как системы автоматического машинного обучения (AutoML) и глубокие нейронные сети, привел к возрождению исследований по *оптимизации гиперпараметров* (hyperparameter optimization, HPO). В этой главе мы представим обзор наиболее известных подходов к реализации HPO. Сначала мы обсудим способы оптимизации *функций черного ящика* (blackbox function), основанные на безмодельных методах и байесовской оптимизации. Поскольку высокие вычислительные требования многих современных приложений машинного обучения делают прямую оптимизацию методом черного ящика чрезвычайно дорогостоящей, далее мы сосредоточимся на современных методах *переменной точности* (multi-fidelity method), которые используют гораздо более дешевые в вычислительном плане аппроксимированные варианты функции черного ящика для приблизительной оценки качества настройки гиперпараметров. В завершение главы мы обсудим нерешенные проблемы и направления будущих исследований.

1.1. Введение

Каждая система машинного обучения имеет гиперпараметры, и самой основной задачей в автоматизированном машинном обучении является автоматическая настройка этих гиперпараметров для оптимизации производительности. Современные глубокие нейронные сети особенно сильно зависят

от широкого спектра гиперпараметров, определяющих архитектуру нейронной сети, регуляризацию и оптимизацию. Автоматизированная оптимизация гиперпараметров имеет несколько важных применений и позволяет:

- сократить человеческие трудозатраты, необходимые для применения машинного обучения. Это особенно важно в контексте AutoML;
- улучшить производительность алгоритмов машинного обучения (адаптируя их к конкретной задаче). Это привело к появлению новых важных эталонов машинного обучения, предложенных в нескольких исследованиях (например, [105, 140]);
- улучшить воспроизводимость и достоверность научных исследований. Автоматизированный процесс HPO явно более воспроизводим, чем ручной поиск. Он облегчает справедливое сравнение научных результатов, поскольку различные методы можно справедливо сравнивать только в том случае, если все они обладают одинаковым уровнем настройки для решения поставленной задачи [14, 133].

Задача HPO имеет долгую историю, восходящую к 1990-м гг. (например, [77, 82, 107, 126]), и уже тогда было установлено, что различные конфигурации гиперпараметров работают лучше для разных наборов данных [82]. В отличие от этого знания довольно новым является понимание, что методику HPO можно использовать для адаптации конвейеров общего назначения к конкретным областям применения [30]. В настоящее время также считается общепризнанным фактом, что настраиваемые гиперпараметры лучше, чем настройки по умолчанию, предоставляемые распространенными библиотеками машинного обучения [100, 116, 130, 149].

В связи с растущим использованием машинного обучения в компаниях HPO также представляет значительный коммерческий интерес и играет все большую роль в инструментах внутри компании [45] как часть облачных сервисов машинного обучения [6, 89] или как самостоятельная услуга [137].

Реализация HPO сталкивается с несколькими проблемами, которые затрудняют применение технологии на практике:

- оценка функций для больших моделей (например, в глубоком обучении), сложных конвейеров машинного обучения или больших наборов данных обходится чрезвычайно дорого;
- пространство конфигураций часто бывает сложным (состоящим из смеси непрерывных, категориальных и условных гиперпараметров) и многомерным. Кроме того, не всегда ясно, какие из гиперпараметров алгоритма необходимо оптимизировать и в каких диапазонах;
- обычно у нас нет доступа к градиенту функции потерь относительно гиперпараметров. Более того, другие свойства целевой функции, часто используемые в классической оптимизации, например выпуклость и гладкость, обычно не применимы к гиперпараметрам;
- невозможно напрямую оптимизировать эффективность обобщения, поскольку наборы обучающих данных имеют ограниченный размер.

Заинтересованных читателей, желающих глубже изучить данную проблематику, мы отсылаем к другим обзорам HPO [64, 94].

Данная глава построена следующим образом. Сначала мы формально определим задачу HPO и обсудим ее варианты (раздел 1.2). Затем мы об-

судим алгоритмы оптимизации черного ящика для решения задачи HPO (раздел 1.3). Далее мы сосредоточимся на современных методах переменной точности, которые позволяют использовать HPO для оптимизации даже очень дорогостоящих в вычислительном плане моделей благодаря приблизительным мерам производительности, которые дешевле, чем полные оценки модели (раздел 1.4). Затем мы представим обзор наиболее важных систем оптимизации гиперпараметров и приложений к AutoML (раздел 1.5) и завершим главу обсуждением нерешенных проблем (раздел 1.6).

1.2. Постановка задачи

Обозначим за $\mathcal{A}$ алгоритм машинного обучения с N гиперпараметрами. Обозначим область n-го гиперпараметра через Λ_n, а общее *пространство конфигураций гиперпараметров* как $\boldsymbol{\Lambda} = \Lambda_1 \times \Lambda_2 \times \ldots \Lambda_N$. Вектор гиперпараметров обозначим как $\boldsymbol{\lambda} \in \boldsymbol{\Lambda}$, а алгоритм $\mathcal{A}$ с его гиперпараметрами, представленными в $\boldsymbol{\lambda}$, обозначим как $\mathcal{A}_{\boldsymbol{\lambda}}$.

Область гиперпараметра может быть вещественной (например, скорость обучения), целочисленной (например, количество слоев), бинарной (например, использовать или нет раннюю остановку) или категориальной (например, выбор оптимизатора). Для целочисленных и вещественных гиперпараметров области в основном ограничены практическими соображениями, за некоторыми исключениями [12, 113, 136].

Кроме того, пространство конфигураций может обладать *условностью*, т. е. один гиперпараметр может иметь смысл только в том случае, если другой гиперпараметр (или некоторая комбинация гиперпараметров) принимает определенное значение. Условные пространства имеют форму направленных ациклических графов. Такие условные пространства возникают, например, при автоматической настройке конвейеров машинного обучения, когда выбор между различными алгоритмами предварительной обработки и машинного обучения моделируется как категориальный гиперпараметр – задача, известная как *полный выбор модели* (full model selection, FMS), или комбинированная задача *выбора алгоритма и оптимизации гиперпараметров* (combined algorithm selection and hyperparameter optimization, CASH) [30, 34, 83, 149]. Они также возникают при оптимизации архитектуры нейронной сети: например, число слоев может быть целочисленным гиперпараметром, а гиперпараметры каждого слоя i активны только в том случае, если глубина сети не меньше i [12, 14, 33].

При заданном наборе данных $\mathcal{D}$ наша цель состоит в том, чтобы найти

$$\boldsymbol{\lambda}^* = \underset{\boldsymbol{\lambda} \in \boldsymbol{\Lambda}}{\operatorname{argmin}} \, \mathbb{E}_{(D_{train}, D_{valid}) \sim \mathcal{D}} \mathbf{V}(\mathcal{L}, \mathcal{A}_{\boldsymbol{\lambda}}, D_{train}, D_{valid}), \tag{1.1}$$

где $\mathbf{V}(\mathcal{L}, \mathcal{A}_{\boldsymbol{\lambda}}, D_{train}, D_{valid})$ измеряет потери модели, созданной алгоритмом $\mathcal{A}$ с гиперпараметрами $\boldsymbol{\lambda}$ на обучающих данных D_{train} и оцененной на проверочных данных D_{valid}. На практике мы имеем доступ только к конечным данным $D \sim \mathcal{D}$ и поэтому должны аппроксимировать ожидание в уравнении (1.1).

Популярными вариантами протокола проверки $\mathbf{V}(\cdot, \cdot, \cdot, \cdot)$ являются выделение контрольных данных (holdout) и ошибка перекрестной проверки для заданной пользователем функции потерь (например, коэффициент неправильной классификации); обзор протоколов проверки представлен в статье Бишля и др. [16]. Существуют несколько стратегий для сокращения времени оценки: можно тестировать алгоритмы машинного обучения только на подмножестве сверток [149], только на подмножестве данных [78, 102, 147] или на небольшом количестве итераций; мы обсудим некоторые из этих стратегий более подробно в разделе 1.4. В публикациях по многозадачной [147] и многоисточниковой [121] оптимизации были предложены дополнительные вычислительно дешевые вспомогательные задачи, которые можно решать вместо уравнения (1.1). Они способны без особых вычислительных затрат предоставить информацию для поддержки процесса HPO, но не обязательно обучают модель на нужном наборе данных и поэтому не дают пригодную для использования модель в качестве побочного продукта.

1.2.1. Альтернативы оптимизации: ансамблирование и маргинализация

Решение уравнения (1.1) с помощью одного из методов, описанных в остальной части этой главы, обычно требует подбора алгоритма машинного обучения $\mathcal{A}$ с несколькими векторами гиперпараметров $\boldsymbol{\lambda}_t$. Вместо использования оператора argmin можно либо построить ансамбль (который стремится минимизировать потери для заданного протокола валидации), либо интегрировать все гиперпараметры (если рассматриваемая модель является вероятностной). Сравнение выбора моделей по частотному и байесовскому методу представлено в работе Гайона и др. [50] и ссылках на нее.

Использование только одной конфигурации гиперпараметров может быть расточительным, если процесс HPO нашел несколько потенциально хороших конфигураций, и их объединение в ансамбль может существенно улучшить производительность [109]. Это особенно полезно в системах AutoML с большим пространством конфигураций (например, в FMS или CASH), где хорошие конфигурации могут быть очень разнообразными, что увеличивает потенциальный выигрыш от их объединения [4, 19, 31, 34]. В целях дальнейшего повышения производительности метод Automatic Frankensteining [155] использует HPO для обучения стековой модели [156] на выходах моделей, найденных с помощью HPO; модели второго уровня затем объединяются с помощью традиционной стратегии ансамблирования.

В упомянутых методах ансамблирование применяют после процедуры HPO. Хотя это улучшает производительность на практике, базовые модели не оптимизированы для ансамблирования. Можно выполнить прямую оптимизацию этих моделей, что позволит максимально улучшить существующий ансамбль [97].

Наконец, при работе с байесовскими моделями часто удается интегрировать гиперпараметры алгоритма машинного обучения, например, используя

максимизацию доказательств [98], *усреднение байесовской модели* [56], *выборку по уровням* [111] или *эмпирический байесовский подход* [103].

1.2.2. Оптимизация по нескольким целям

На практике часто бывает необходимо найти компромисс между двумя или более целями, такими как производительность модели и потребление ресурсов [65] (см. также главу 3) или несколько функций потерь [57]. Возможные решения могут быть получены двумя способами.

Во-первых, если известно ограничение на вторичный показатель производительности (например, максимальное потребление памяти), то задачу можно сформулировать как задачу оптимизации с ограничениями. Мы обсудим обработку ограничений в байесовской оптимизации в разделе 1.3.2.4.

Во-вторых (и это более общий случай), можно применить многокритериальную оптимизацию для поиска *фронта Парето* – набора конфигураций, которые являются оптимальным компромиссом между целями в том смысле, что для каждой конфигурации на фронте Парето не существует другой конфигурации, которая работает лучше хотя бы для одной цели и хотя бы столь же хорошо для всех других целей. Затем пользователь может выбрать конфигурацию с фронта Парето. Мы отсылаем заинтересованного читателя к дополнительной литературе по этой теме [53, 57, 65, 134].

1.3. Оптимизация гиперпараметров методом черного ящика

В общем случае любой метод оптимизации черного ящика применим к HPO. Вследствие невыпуклой природы задачи алгоритмы глобальной оптимизации обычно предпочтительнее, но определенная локальность в процессе оптимизации полезна для того, чтобы добиться заметного прогресса за несколько оценок функции. Сначала мы обсудим методы HPO без моделей, а затем опишем методы байесовской оптимизации с черным ящиком.

1.3.1. Оптимизация методом черного ящика без моделей

Поиск по сетке (grid search) – это самый базовый метод HPO, также известный как *полное факторное планирование* (full factorial design) [110]. Пользователь задает конечное множество значений для каждого гиперпараметра, а поиск по сетке оценивает декартово произведение этих множеств. Такой подход страдает от проклятия размерности, поскольку требуемое количество оценок функций растет экспоненциально по мере увеличения размерности

пространства конфигураций. Дополнительный недостаток поиска по сетке заключается в том, что увеличение разрешения дискретизации (уменьшение шага гиперпараметров) существенно увеличивает требуемое число оценок функций.

Простой альтернативой поиску по сетке является *случайный поиск* (random search)[1] [13]. Как следует из названия, случайный поиск выбирает конфигурации случайным образом, пока не будет исчерпан определенный бюджет на поиск. Этот подход работает лучше, чем поиск по сетке, когда некоторые гиперпараметры намного важнее других (это свойство имеет место во многих случаях [13, 61]). Интуитивно понятно, что при фиксированном бюджете в B оценок функций количество различных значений, которые может позволить себе поиск по сетке для каждого из N гиперпараметров, составляет только $B^{1/N}$, в то время как случайный поиск будет исследовать B различных значений для каждого из них (рис. 1.1).

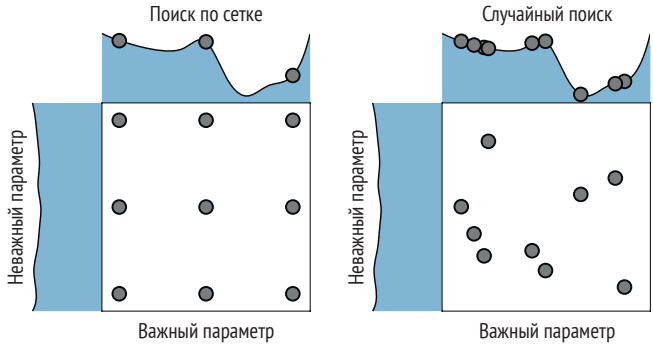

Рис. 1.1 ❖ Сравнение поиска по сетке и случайного поиска для минимизации функции с одним важным и одним неважным параметром. Этот рисунок основан на рис. 1 из работы Бергстры и Бенджио [13]

К дополнительным преимуществам случайного поиска в сравнении с поиском по сетке относятся более легкое распараллеливание (поскольку воркерам не нужно общаться друг с другом и отказавшие воркеры не оставляют дыр в структуре) и гибкое распределение ресурсов (поскольку можно добавить произвольное количество случайных точек к структуре случайного поиска, и все равно получится структура случайного поиска; для поиска по сетке это не так).

Случайный поиск является полезной базовой схемой, поскольку он не делает никаких предположений относительно оптимизируемого алгоритма машинного обучения и – при достаточном количестве ресурсов – стремится достичь производительности, произвольно близкой к оптимальной. Поэтому чередование случайного поиска с более сложными стратегиями оптимизации позволяет гарантировать минимальную скорость сходимости,

[1] В некоторых дисциплинах он также известен как *чисто случайный поиск* (pure random search) [158].

а также добавляет исследование, которое может улучшить поиск на основе модели [3, 59]. Случайный поиск также является полезным методом для инициализации процесса поиска, поскольку он исследует все пространство конфигурации и, таким образом, часто находит параметры с приемлемой производительностью. Однако он не является универсальным решением и часто требует гораздо больше времени, чем методы направленного поиска, чтобы определить одну из наиболее эффективных конфигураций гиперпараметров: например, при выборке без замены из пространства конфигураций с N булевыми гиперпараметрами с хорошими и плохими настройками и без эффектов взаимодействия для нахождения оптимума потребуется 2^{N-1} оценок функций, в то время как направленный поиск может найти оптимум за $N + 1$ оценок следующим образом: начиная с произвольной конфигурации, перебираем гиперпараметры и изменяем их по одному за раз, сохраняя полученную конфигурацию, если производительность улучшается, и отменяя изменения, если это не так. Соответственно, методы направленного поиска, которые мы рассматриваем в следующих разделах, обычно превосходят случайный поиск [12, 14, 33, 90, 153].

Популяционные методы, такие как *генетические алгоритмы*, *эволюционные алгоритмы*, *эволюционные стратегии* и *оптимизация роя частиц*, – это алгоритмы оптимизации, которые используют *популяцию*, т. е. набор конфигураций, и улучшают эту популяцию, применяя локальные возмущения (так называемые *мутации*) и комбинации различных членов (так называемый *кроссовер*) для получения нового поколения лучших конфигураций. Эти методы просты, могут работать с различными типами данных и, что удивительно, легко распараллеливаются [91], поскольку популяция из N членов может быть оценена параллельно на N машинах.

Одним из наиболее известных популяционных методов является эволюционная версия ковариационной матрицы (CMA-ES [51]). Эта простая эволюционная стратегия выбирает конфигурации из многомерного гауссова распределения, среднее и ковариация которого обновляются в каждом поколении на основе успеха особей, составляющих популяцию. CMA-ES является одним из наиболее конкурентоспособных алгоритмов оптимизации методом черного ящика и регулярно побеждает в соревновании Black-Box Optimization Benchmarking (BBOB) [11].

За более подробной информацией о популяционных методах мы отсылаем читателя к [28, 138]. Применение популяционных методов для оптимизации гиперпараметров будет рассмотрено в разделе 1.5, применение для поиска нейронных архитектур – в главе 3, и генетическое программирование для конвейеров AutoML – в главе 8.

1.3.2. Байесовская оптимизация

Байесовская оптимизация – это современный метод глобальной оптимизации вычислительно дорогостоящих функций черного ящика, который получил широкое распространение в области HPO благодаря достижению новых пере-