МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

МАТЕРИАЛЫ

IX-й Международной научной конференции

«МАТЕМАТИЧЕСКОЕ И ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ИНФОРМАЦИОННЫХ, ТЕХНИЧЕСКИХ И ЭКОНОМИЧЕСКИХ СИСТЕМ»

Томск, 26–28 мая 2022 г.

Под общей редакцией
кандидата технических наук И.С. Шмырина

Томск
Издательство Томского государственного университета
2022

V. ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ, МАШИННОЕ ОБУЧЕНИЕ, БОЛЬШИЕ ДАННЫЕ

DOI: 10.17223/978-5-907572-27-0-2022-32

АВТОМАТИЧЕСКИЙ ПОДБОР ГИПЕРПАРАМЕТРОВ ДЛЯ ОБУЧЕНИЯ НЕЙРОННЫХ СЕТЕЙ

Гитман Д.А., Пахомова Е.Г.
Томский государственный университет
dgtm2001@gmail.com, pakhomovaeg@yandex.ru

Введение

Подбор гиперпараметров – один из наиболее важных этапов обучения нейросетей. К сожалению, подбор гиперпараметров – трудный и долгий процесс. Его оптимизация может снять с пользователя большой объем работы. Максимальным уровнем оптимизации будет автоматизация. На текущий день существует множество подходов к автоматизации подбора гиперпараметров. Популярны такие методы, как поиск по решетке [1] и случайный поиск [2]. Оба метода основаны на полном переборе заданного вручную или случайным образом подмножества пространств гиперпараметров. Менее распространенные: байесовская оптимизация [3], основанные на градиенте [4], популяционные [5,6] и bandit-based методы [7]. Эти методы могут применяться для оптимизации любых функций, значение которых мы можем посчитать в любой точке. Обычно, градиенты этих функций мы вычислить не можем. В основном, больше никаких предположений о функциях эти методы не делают, и, соответственно, не могут учитывать структуру конкретно нейросетевой задачи оптимизации. Хотя многие такие методы используют раннюю остановку, если обучение не выглядит удачным, более сложный анализ конкретных нейросетевых функций они не проводят.

Однако, несмотря на достаточное количество существующих методов, большинство из них не имеют широкого применения в профессиональном сообществе (за исключением поиска по решетке, случайного поиска и их аналогов). Одной из причин этого явления может быть то, что данные методы не учитывают структуру конкретной задачи, не имеют полного доступа ко всей доступной информации в процессе обучения, а, значит, часто будут работать не так эффективно, как если бы поиск проводился с участием человека-профессионала в этой области.

Способ автоматизации, который будет описан в данной работе, отличается от представленных выше. Он основан на том, что нам многое понятно об устройстве конкретно нейросетевой функции. Мы понимаем, например, что сокращение веса (weight decay), исключение (dropout), увеличение данных (data augmentation) можно использовать для профилактики переобучения. И хотя иногда гиперпараметры могут влиять не только на те показатели, для изменения которых они были созданы, в большинстве случаев они ведут себя предсказуемо. Так, например в [8] показано, что сокращение веса (weight decay) не только помогает при переобучении, но и может улучшать значения функции потерь на тренировочной выборке.

Таким образом, ISearch отличается от других методов по двум параметрам. Во-первых, он использует знания об устройстве конкретно нейросетевой структуры и поэтому не подходит для других функций. Во-вторых, ISearch – интерактивный метод, и носит скорее рекомендательный характер. Пользователь может соглашаться или не соглашаться с предложенным данным методом направлением для исследования.

Помимо метода автоматического подбора гиперпараметров в работе рассматриваются такие модификации оптимизаторов, как SGDW [9] и нормализованный импульс (momentum) [10,11]. Исследуемые гипотезы были следующими: SGDW позволяет сде-

лать менее зависимыми такие гиперпараметры, как импульс (momentum) и сокращение веса (weight decay), нормализованный импульс (momentum) позволяет уменьшить зависимость коэффициента скорости обучения (learning rate) и импульса (momentum). Зависимые гиперпараметры существенно сложнее подбирать. Их обязательно надо искать парами. Сокращение веса (weight decay), при котором значение функции потерь на проверочной выборке (test loss) наименьшее при фиксированном импульсе (momentum), может не совпадать с сокращением веса (weight decay), при котором значение функции потерь на проверочной выборке (test loss) наименьшее при другом фиксированном импульсе (momentum).

1. Обзор методов

1.1. Поиск по решетке

Пользователь указывает конечный набор значений для каждого гиперпараметра. Поиск по решетке сравнивает результаты обучения всех множеств из декартова произведения этих наборов гиперпараметров и запоминает наилучший результат. С увеличением размера набора значений каждого параметра требуемое количество сеток, которые надо обучить, возрастает геометрически. Это большой минус поиска по решетке.

1.2. Случайный поиск

Случайный поиск [2] использует случайный выбор наборов гиперпараметров для исследования из заданного пользователем диапазона значений, пока не исчерпается определенный бюджет для поиска. Этот метод работает лучше, чем поиск по решетке, когда некоторые гиперпарамеры гораздо важнее других, т.к. поиск по решетке много внимания уделяет так называемым «не важным» параметрам. Это проиллюстрировано на рис. 1 (рисунок взят из [1]).

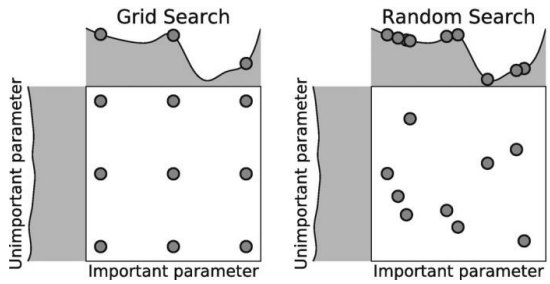


Рис. 1. Сравнение работы поиска по решетке и случайного поиска для функции с одним «важным» и одним «не важным» параметром

1.3. Байесовская оптимизация

Байесовская оптимизация [3] –семейство итеративных алгоритмов с двумя ключевыми составляющими: вероятностная суррогатная модель (probabilistic surrogate model) – наше предположение о виде истинной функции (objective function), и функция сбора данных (acquisition function), благодаря которой выбирается следующая точка для обучения. Функция сбора данных определяет «полезность» различных точек-кандидатов, выбирая компромисс между использованием уже имеющейся информации (exploitation), т.е. приближением к предполагаемому минимуму и «разведыванием» (exploration) новой области пространства. На рис. 2, (рисунок взят из [1]), проиллюстрирована работа байесовской оптимизации для одномерной функции.

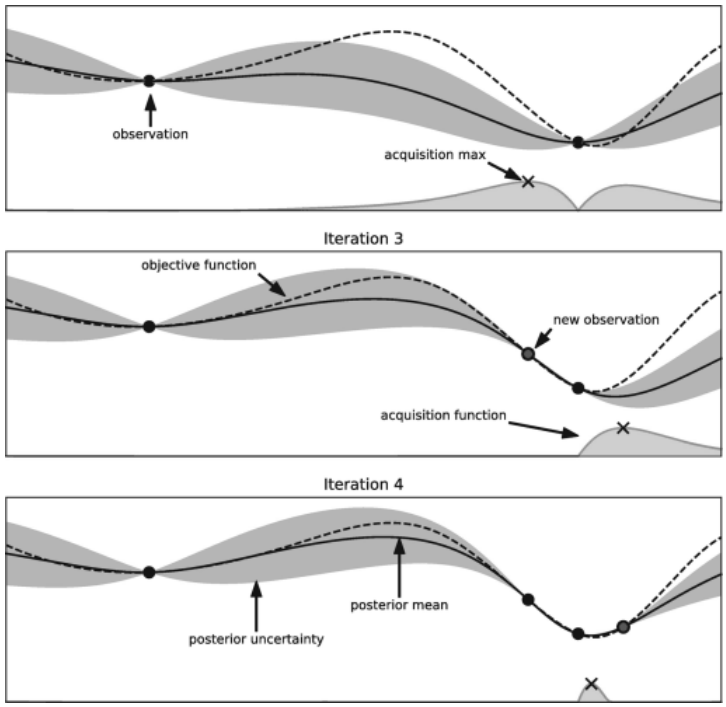


Рис. 2. Демонстрация работы байесовской оптимизации для одномерной функции: сплошная черная линия – предполагаемый вид функции; пунктирная линия – истинный вид функции; серая функция находит компромисс между тем, чтобы получить больше знаний о виде функции, и приближением к локальному минимуму; точка, где значение серой функции максимально, считается наиболее удачным выбором для следующего обучения

1.4. Методы, основанные на градиенте

Суть этих методов в том, чтобы использовать градиентный спуск относительно некоторых гиперпараметров. В Hyperpgadient [4], например, настраивается лишь один гиперпараметр – коэффициент скорости обучения (learning rate). На каждом шаге этот гиперпараметр приближается к оптимальному значению благодаря градиентному спуску относительно коэффициента скорости обучения (learning rate). При этом вычисление градиента не требует дополнительных затрат. Для его получения используется градиент относительно весов, который вычисляется оптимизатором.

1.5. Популяционные методы

К популяционным методам относятся эволюционные [5] и генетические [6] алгоритмы. Это методы оптимизации, которые поддерживают набор конфигураций и улучшают их путем применения местных возмущений (мутаций) и комбинации различных членов (кроссоверов) для получения нового поколения лучших конфигураций. На первом этапе создается начальная популяция. К объектам этой популяции применяется кроссовер и мутация. Из получившихся новых конфигураций и тех, что были изначально, выбираются лучшие (способов выбора тех конфигураций, которые перейдут на следующих этап, много) и для них эта процедура повторяется.

2. ISearch

2.1. Структура программы

В этом разделе описана структура программы и объяснены причины выбора именно такой структуры.

class ISearch():
def __init__(self)
def add_result (self,data,point)
def next_point (self)

def why(self)
def stop_signal(self,data, flag,patience,parameter,epochs)
class Engine():
def__init__(self,teach, upload,backup)
def start(self,cur_point=None,stop_signal=None,method=None)
def upload (self)

Программа состоит из двух блоков, а именно – из двух классов. Первый отвечает за связь пользователя с алгоритмом автоматического подбора (будем называть его Engine). Второй отвечает за подбор параметров (назовем его ISearch по названию метода). Основными методами класса ISearch являются функции next_point, add_result и why. next_point возвращает предлагаемую алгоритмом следующую точку для обучения. Чтобы определить следующую точку, ISearch сохраняет у себя наилучшую обученную модель. Сделать это позволяет функция add_result. Класс ISearch не влияет на сам процесс обучения. Он предлагает гиперпараметры для обучения и потом получает результаты. Функция why позволяет пользователю запросить причины выбора следующей точки для обучения. Engine имеет два метода, доступных пользователю: start и upload. upload позволяет запустить алгоритм с последнего сохранения, start запускает алгоритм сначала – Engine создает объект класса ISearch, у которого может запрашивать следующие точки, передавать статистики обучения и получать информацию о причинах выбора тех или иных гиперпараметров. Стоит отметить, что в функцию add_result могут прийти статистики обучения не той точки, которую предлагал метод, а той, которую выбрал пользователь. Пользователь, в свою очередь, должен передать в конструктор Engine три функции. Функция teach по заданным гиперпараметрам обучит сеть и вернет статистики обучения и саму модель, backup позволит алгоритму делать сохранения в удобном для пользователя формате, upload загружает последнее сохранение алгоритма. Не предполагается, что пользователь будет передавать в функцию start какие-либо данные, в случае необходимости это сделает функция upload. Функция пользователя teach, помимо гиперпараметров, должна принимать на вход функцию из класса ISearch – stop_signal, которая определит, когда обучение нужно прекратить. Обучение прекращается либо когда истекло заданное количество эпох, либо когда сеть начала переобучаться. Так реализуется ранняя остановка (early stopping) в пользовательской функции. Таким образом, всего есть три блока – функции, предоставляемые пользователем для сохранения и загрузки текущего состояния алгоритма и для обучения сети, класс ISearch, выбирающий гиперпараметры для обучения, и Engine, который осуществляет связь между пользователем и методом.

Такая структура выбрана по ряду причин. Во-первых, класс Engine не обязан работать именно с ISearch. Можно подключить любой другой метод автоматического подбора, если предварительно определить в нем три функции, описанные выше. Во-вторых, в будущем при подобной организации появляется возможность подбирать разные гиперпараметры разными методами. Для этого нужно будет просто оформить их в один класс, который будет создаваться в Engine вместо ISearch

2.2. Структура алгоритма

На текущем этапе программа подбирает следующие гиперпараметры: скорость обучения (learning rate), сокращение веса (weight decay), импульс (momentum), расписание коэффициента скорости обучения (scheduler) и разогрев (warm up). Эффективность алгоритма проверялась на двух сетях – ResNet-18 [12] и MobileNet [13]. Оптимизатором выбран SGD с нормализованным импульсом (momentum) и сокращением веса (weight decay). Для расписания коэффициента скорости обучения используется ступенчатая функция, которая уменьшается, если функция потерь на проверочном наборе данных (test loss) перестает падать. Одной из причин того, что сеть перестает обучаться, может быть большой коэффициент скорости обучения (learning rate). Поэтому при возникно-

вении такой необходимости функция уменьшит этот параметр. В [14] показано, что при правильном подборе гиперпараметров не имеет значения, какой оптимизатор использовать, поэтому оптимизатор исключен из списка гиперпараметров, которые подбирает ISearch. В разделе «Эксперименты» будет показана причина выбора SGD именно с нормализованным импульсом (momentum). В [9,10] описаны метод нормализации импульса (momentum) и SGDW. Предполагается, что эти оптимизации должны позволить ускорить и упростить процесс подбора гиперпараметров за счет уменьшения зависимости таких гиперпараметров, как скорость обучения (learning rate) и импульс (momentum), импульс (momentum) и сокращение веса (weight decay). Далее будет показано, что при использовании нормализованного импульса (momentum) можно не сосредотачиваться на подборе импульса (momentum). Поэтому этот гиперпараметр взят равным 0.9. Разогрев (warm up) и гиперпараметры, отправляемые в функцию для задания расписания (scheduler) коэффициента скорости обучения (learning rate), подбираются случайно. ISearch довольно сложно расширять на другие гиперпараметры. Нужно понимать, когда, зачем и как стоит менять параметры, которые мы хотим добавить в метод. В качестве альтернативы гиперпараметры, не включенные еще в ISearch, могут подбираться любым другим методом автоматического подбора. В нашем случае это случайный поиск. Про подбор коэффициента сокращения веса (weight decay) и коэффициента скорости обучения (learning rate) будет рассказано ниже.

2.2.1. Коэффициент сокращения веса (weight decay)

сокращения веса (weight decay) – это гиперпараметр, позволяющий предотвратить переобучение. Переобучение определяется следующими способами:

1. График точности обучения на проверочной выборке (test accuracy) начинает уменьшаться. Для профилактики используется метод ранней остановки (early stopping).
2. В сравнении с другим запуском можно обнаружить переобучение. Пусть есть два запуска: запуск 1 и запуск 2. Запуск 2 переобучился относительно запуска 1, если значение точности обучения на проверочной выборке (train accuracy) у запуска 2 выше, а значение точности обучения на тренировачной выборке (test accuracy) ниже. Аналогично переобучение можно определять и по значениям функций потерь на тренировочной и проверочной выборке (train loss, test loss).

При обнаружении переобучения начинается поиск коэффициента сокращения веса (weight decay) до тех пор, пока не найдется оптимальное значение. Предполагается, что есть локальный минимум при фиксированных остальных гиперпараметрах, поэтому используется тернарный поиск для нахождения коэффициента сокращения веса (weight decay)

2.2.2. Коэффициент скорости обучения (learning rate)

Начальный коэффициент скорости обучения (learning rate) берется максимально большой – такой, чтобы за первую эпоху значение точности обучения на проверочной выборке (test accuracy) увеличилось. Это делается для избежания переобучения [15].

2.2.3. Импульс (momentum)

В разделе «Эксперименты» будет показано, что настройкой такого гиперпараметра, как импульс (momentum), можно пренебречь, поэтому импульс (momentum) всегда равен константному значению 0.9

2.2.4. Расписание (scheduler)

В качестве расписания скорости обучения (learning rate scheduler) была выбрана ступенчатая функция, которая уменьшается, если функция потерь или точность обучения на проверочном наборе данных (test loss, test accuracy) перестает падать или возрастать соответственно. На данный момент гиперпараметры, передаваемые в функцию, подбираются случайным поиском.

2.2.5. Разогрев (warm up)

Разогрев нужен, если на первых эпохах мы используем большие значения коэффициента скорости обучения (learning rate). Сначала коэффициент скорости обучения (learning rate) линейно возрастает до заданного большого значения. Количество эпох, выделяемое для увеличения коэффициента скорости обучения (learning rate), является гиперпараметром, подбираемым случайно.

3. Модификации SGD с импульсом (momentum) и коэффициентом сокращения веса (weight decay)

3.1. Нормализованный импульс

В данном разделе рассматривается модификация оптимизатора SGD с импульсом (momentum) и коэффициентом сокращения веса (weight decay) путем нормализации импульса (momentum). Предполагается, что нормализация импульса (momentum) позволит сделать коэффициент скорости обучения (learning rate) и импульс (momentum) менее зависимыми. Чтобы увидеть зависимость этих гиперпараметров, распишем рекурсию.

Формула SGD с импульсом (momentum) и коэффициентом сокращения веса (weight decay) выглядит так:

$$\begin{aligned} d_{k+1} &= \beta d_k + g_k + \lambda L_k, \\ w_{k+1} &= w_k - \alpha d_{k-1}, \end{aligned} \tag{1}$$

где w – веса (weight), α – коэффициент скорости обучения (learning rate), β – импульс (momentum), g – стохастический градиент функции потерь (loss), λ – коэффициент сокращения веса (weight decay), $L_k = \| w^2 \|_k$.

Распишем рекурсию:

$$w_{k+1} = w_{k-1} - \alpha d_k - \alpha d_{k-1} = w_0 - \alpha\left(d_1 + ... + d_{k+1}\right),$$

$$d_{k+1} = \beta\left(\beta d_{k-1} + g_{k-1} + \lambda L_{k-1}\right) + g_k + \lambda L_k = \beta^2 d_{k-1} + \beta g_{k-1} + \beta\lambda L_{k-1} + g_k + \lambda L_k =$$

$$= g_k + \lambda L_k + \left(g_{k-1} + \lambda L_{k-1}\right)\beta + ... + \left(g_0 + \lambda L_0\right)\beta^k + \beta^{k+1} d_0,$$

$$w_{k+1} = w_0 - \alpha\left(\beta + ... + \beta^{k+1}\right) d_0 - \alpha\left(1 + ... + \beta^k\right)\left(g_0 + \lambda L_0\right) - ... - \alpha\left(g_k + \lambda L_k\right).$$

Предположим, что k – достаточно большое, и соберем суммы в скобках по формуле геометрической прогрессии:

$$\begin{aligned} w_{k+1} &= w_0 - \alpha\frac{1}{1-\beta}\left(\beta d_0 + g_0 + \lambda L_0 + g_1 + \lambda L_1 + ... + g_k + \lambda L_k\right),\ d_0 = 0, \\ w_{k+1} &= w_0 - \alpha\frac{1}{1-\beta}\left(g_0 + g_1 + ... + g_k\right) - \alpha\frac{\lambda}{1-\beta}\left(L_0 + L_1 + ... + L_k\right). \end{aligned} \tag{2}$$

Произведение α и $\frac{1}{1-\beta}$ указывает на зависимость гиперпараметров. Домножим в (1) градиент на $\left(1-\beta\right)$, чтобы избежать зависимости:

$$\begin{aligned} d_{k+1} &= \beta d_k + (1-\beta) g_k + \lambda L_k, \\ w_{k+1} &= w_k - \alpha d_{k-1} \end{aligned} \tag{3}$$

Формула изменения весов примет вид

$$w_{k+1} = w_0 - \alpha\left(g_0 + g_1 + ... + g_k\right) - \alpha\frac{\lambda}{1-\beta}\left(L_0 + L_1 + ... + L_k\right). \tag{4}$$

Если сравнить полученные формулы (2) и (4) для обновления весов, видно, что если использовать формулу нормализованного импульса (momentum) (3), зависимость коэффициента скорости обучения (learning rate) и импульса (momentum) уменьшается.

3.2. Оптимизатор SGDW

В данном разделе рассматривается модификация оптимизатора SGD с импульсом (momentum) и коэффициентом сокращения веса (weight decay) путем добавления коэффициента сокращения веса (weight decay) непосредственно к весам. Предполагается, что такая модификация позволит сделать коэффициент сокращения веса (weight decay) и импульс (momentum) менее зависимыми.

Зависимость этих гиперпараметров в формуле SGD с импульсом (momentum) (1) можно увидеть, если расписать рекурсию (2) и обратить внимание на произведение λ и $\frac{1}{1-\beta}$.

Если в (1) добавлять коэффициент сокращения веса (weight decay) непосредственно к весам, получим формулу обновления весов для SGDW оптимизатора:

$$\begin{aligned} d_{k+1} &= \beta d_k + g_k, \\ w_{k+1} &= w_k - \alpha d_{k-1} + \lambda L_k. \end{aligned} \tag{5}$$

Распишем рекурсию, чтобы убедиться, что зависимость исчезла:

$$w_{k+1} = w_{k-1} - \alpha d_k + \lambda L_{k-1} - \alpha d_{k+1} + \lambda L_k = w_0 - \alpha\left(d_1 + ... + d_{k+1}\right) + \lambda\left(L_0 + ... + L_k\right),$$

$$d_{k+1} = \beta\left(\beta d_{k-1} + g_{k-1}\right) + g_k = \beta^2 d_{k-1} + \beta g_{k-1} + g_k =$$

$$= g_k + g_{k-1}\beta + ... + g_0\beta^k + \beta^{k+1} d_0,$$

$$w_{k+1} = w_0 - \alpha\left(\beta + ... + \beta^{k+1}\right) d_0 - \alpha\left(1 + ... + \beta^k\right) g_0 - ... - \alpha g_k + \lambda\left(L_0 + ... + L_k\right).$$

Предположим, что k достаточно большое, и соберем суммы в скобках по формуле геометрической прогрессии:

$$w_{k+1} = w_0 - \alpha\frac{1}{1-\beta}\left(\beta d_0 + g_0 + g_1 + ... + g_k\right) + \lambda\left(L_0 + ... + L_k\right)\frac{n!}{r!(n-r)!},\ d_0 = 0,$$

$$w_{k+1} = w_0 - \alpha\frac{1}{1-\beta}\left(g_0 + g_1 + ... + g_k\right) + \lambda\left(L_0 + ... + L_k\right).$$

Мы избавились от произведения, значит, зависимость должна уменьшиться, но необходимы экспериментальные подтверждения, чтобы убедиться, что сделанные нами допущения не влияют на результат.

Формулы нормализованного импульса (momentum) (3) и SGDW (5) можно объединить: $d_{k+1} = \beta d_k + (1-\beta) g_k$, $w_{k+1} = w_k - \alpha d_{k-1} + \lambda L_k$. Расписанная рекурсия тогда примет вид $w_{k+1} = w_0 - \alpha\left(\beta d_0 + g_0 + g_1 + ... + g_k\right) + \lambda\left(L_0 + ... + L_k\right)$.

Теоретически, так мы уменьшили зависимость гиперпараметров от импульса (momentum), что позволит нам его зафиксировать и не подбирать.

4. Эксперименты

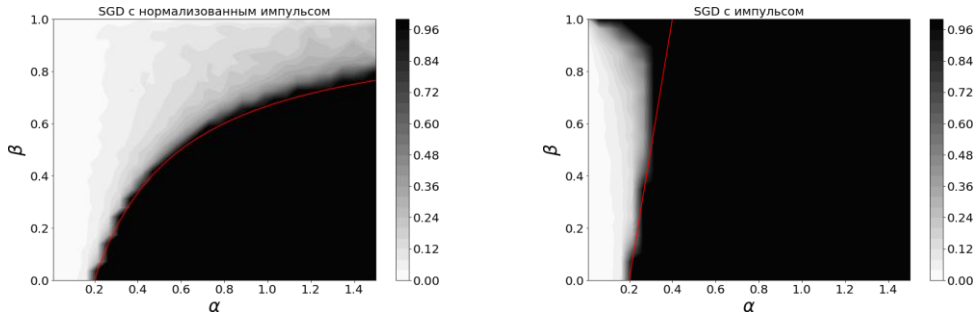
4.1. Сравнение оптимизаторов SGDW и нормализованного импульса

4.1.1. Зависимость коэффициента скорости обучения (learning rate) и импульса (momentum)

Чтобы оценить пользу формулы нормализованного импульса (momentum), рассмотрим зависимость таких гиперпараметров, как скорость обучения (learning rate) и импульс (momentum) вблизи локального минимума. В этой области функцию потерь (loss) можно приблизить к квадратичной. Поэтому рассмотрим зависимость гиперпараметров на квадратичной функции. Эта идея и способ построения графиков взяты из [16].

Обратим внимания на рис. 3а. При достаточно маленьком коэффициенте скорости обучения (learning rate) импульс (momentum) никак не влияет на значение функции потерь (loss) для формулы нормализованного импульса (momentum). С другой стороны,

если посмотреть на подобный рисунок для стохастического градиентного спуска с импульсом (momentum) (рис. 3б), видно, что для него эти рассуждения не верны. Не важно, чему равен коэффициент скорости обучения (learning rate), зависимость от импульса (momentum) есть всегда.

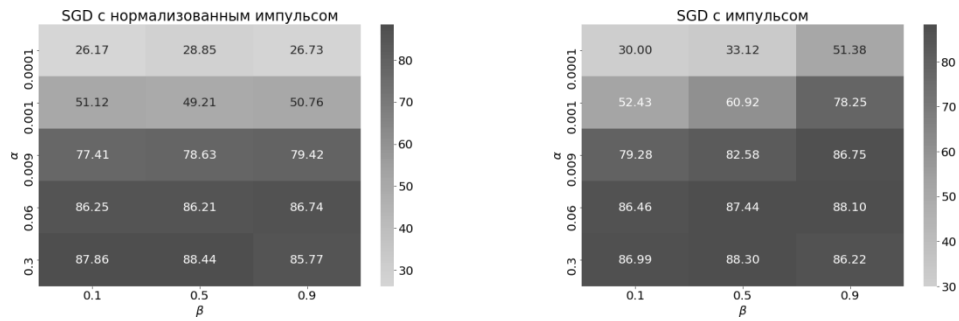


а б

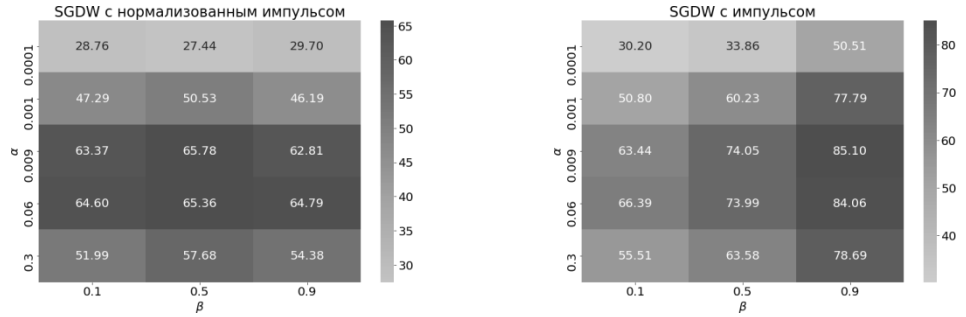
Рис. 3. Зависимость значения функции потерь (loss) от коэффициента скорости обучения (learning rate) и импульса (momentum) для квадратичной функции: а – оптимизатор SGD с нормализованным импульсом (momentum, б – оптимизатор SGD с импульсом (momentum)

Данные рисунки позволяют утверждать, что в окрестности локального минимума оптимизатор с нормализованным импульсом (momentum) сделает гиперпараметры коэффициента скорости обучения (learning rate) и импульса менее зависимыми.

Проверим это утверждение для сети ResNet-18 датасета CIFAR-10. Коэффициент сокращения веса (weight decay) зафиксирован и равен 0.0008. Обратим внимание на рисунки 4а и 4в. Если зафиксировать значение коэффициента скорости обучения (learning rate) и посмотреть на то, как меняется точность обучения (test accuracy) с изменением импульса (momentum), мы обнаружим, что изменения не превышают 3% для оптимизатора SGD с нормализованным импульсом (momentum) и не превышают 6% для оптимизатора SGDW с нормализованным импульсом (momentum). Если же посмотреть теперь на рисунки 4б и 4г, мы увидим, что при фиксированном значении коэффициента скорости обучения (learning rate) значение точности обучения (test accuracy) с изменением импульса (momentum) может меняться до 26% как в SGD с импульсом (momentum), так и в SGDW с импульсом (momentum). Это подтверждает нашу теорию о том, что с добавлением нормализации импульса (momentum) к оптимизаторам зависимость коэффициета скорости обучения (learning rate) и импульса (momentum) снижается.



а б

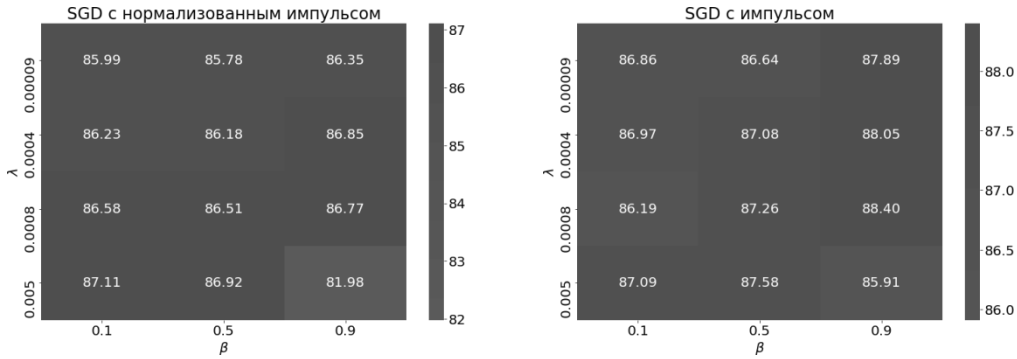


в г

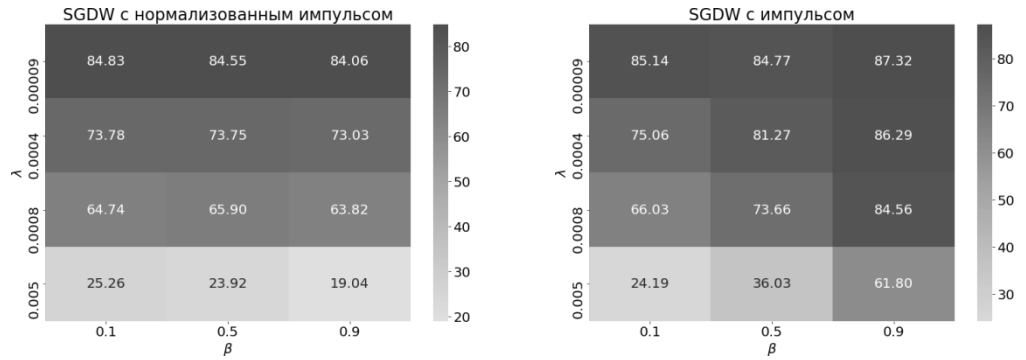
Рис. 4. Зависимость значения точности обучения (test accuracy) от коэффициента скорости обучения (learning rate) и импульса (momentum) для нейросетевой функции: а – оптимизатор SGD с нормализованным импульсом (momentum), б – оптимизатор SGD с импульсом (momentum), в – оптимизатор SGDW с нормализованным импульсом (momentum), г – оптимизатор SGDW с импульсом (momentum)

4.1.2. Зависимость импульса (momentum) и коэффициента сокращение веса (weight decay)

В данном разделе мы экспериментально проверим теорию о том, что оптимизатор SGDW снижает зависимость коэффициента сокращения веса (weight decay) и импульса (momentum). На рис. 5 представлены результаты обучения сети с использованием различных оптимизаторов. Обратим внимание, что результаты для SGDW с нормализованным импульсом (momentum) (рис. 5в) указывают на снижение зависимости, а для SGDW с импульсом (momentum) (рис. 5г) без нормализации коэффициент сокращения веса (weight decay) и импульс (momentum) зависимы, что противоречит нашей гипотезе. Также стоит обратить внимание на то, что в среднем на этих данных SGD с импульсом (momentum) (рис. 5б) и SGD с нормализованным импульсом (momentum) (рис. 5а) работают лучше. Поэтому оптимизатором был выбран именно SGD с нормализованным импульсом.



а б

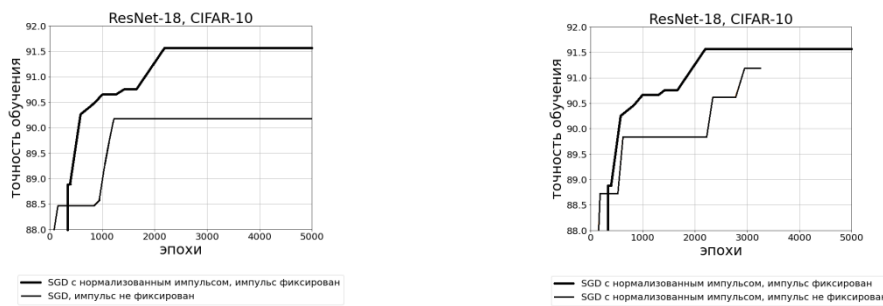


в г

Рис. 5. Зависимость значения точности обучения (test accuracy) от коэффициента сокращения веса (weight decay) и импульса (momentum) для нейросетевой функции: а – оптимизатор SGD с нормализованным импульсом (momentum), б – оптимизатор SGD с импульсом (momentum), в – оптимизатор SGDW с нормализованным им пульсом (momentum), г – оптимизатор SGDW с импульсом (momentum)

4.1.3. Сравнение оптимизаторов с использованием метода ISearch

Для этого раздела метод ISearch был немного изменен. После того, как этап нахождения коэффициента сокращения веса (weight decay) пройден, начинается случайный поиск, где меняются все параметры, кроме коэффициента сокращения веса (weight decay) и, в некоторых запусках, импульса (momentum). В том числе меняется и коэффициент скорости обучения (learning rate). Сравнивать будем результаты с фиксированным импульсом (momentum) и с импульсом (momentum), который в обновленном случайном поиске тоже подбирается. Предполагается, что из-за отсутствия зависимости результаты не должны отличаться. Здесь на рис. 6а видно, насколько сложнее методу подобрать гиперпараметры, когда они зависимы. Коэффициент сокращения веса (weight decay) был подобран для значения импульса, равного 0.9. Если потом это значение начать менять, оптимизатор, для которого параметры независимы, справится с задачей лучше, что и видно на рис. 6а. На рис. 6б методы достигают примерно одинаковой точности, что говорит о независимости гиперпараметров от импульса (momentum).



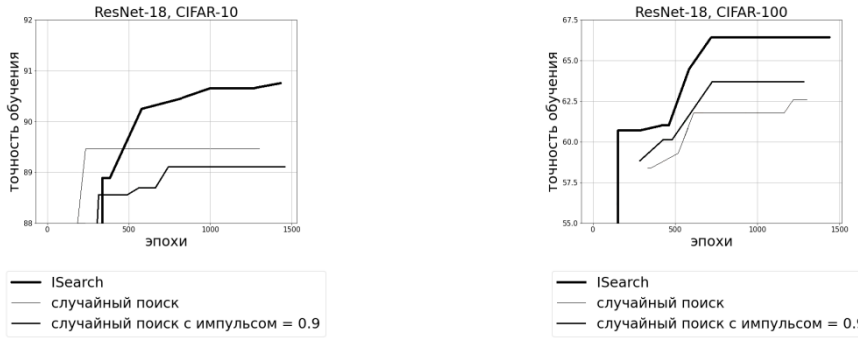
а б

Рис. 6. Сравнение оптимизаторов: а – сравнение для оптимизатора SGD с импульсом (momentum), б – сравнение для оптимизатора SGD с нормализованным импульсом (momentum)

4.2. Сравнение ISearch и случайного поиска для сети Resnet-18

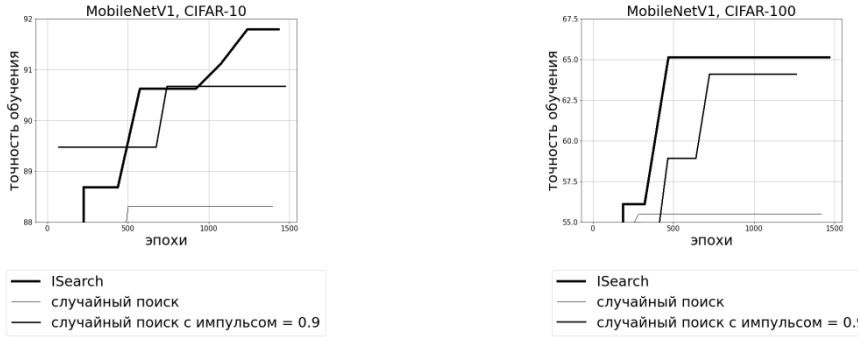
Результаты представлены для методов ISearch, полностью случайного поиска и случайного поиска с фиксированным значением импульса. Диапазоны параметров, подбираемых в ISearch случайно, совпадают с диапазонами этих же параметров в случайных поисках. Во всех методах обучение прекращалось по принципу ранней остановки (early stopping).

На рис. 7, 8 видно, что ISearch работает лучше случайного поиска на всех представленных сетях и датасетах, хотя случайный поиск при фиксированном импульсе работает не намного хуже, в отличие от обычного случайного поиска, которому, чтобы прийти к тем же результатам, потребуется сильно больше времени.



а б

Рис. 7. Сравнение значений наивысшей точности обучения (test accuracy) полученной методами ISearch, случайного поиска и случайного поиска с фиксированным импульсом (momentum), равным 0.9: а – результаты точности обучения (accuracy) для сети ResNet-18 на датасете CIFAR-10, б – результаты точности обучения (accuracy) для сети ResNet-18 на датасете CIFAR-100



а б

Рис. 8. Сравнение значений наивысшей точности обучения (test accuracy) полученной методами ISearch, случайного поиска и случайного поиска с фиксированным импульсом (momentum), равным 0.9: а – результаты точности обучения (accuracy) для сети MobileNetV1 на датасете CIFAR-10, б – результаты точности обучения (accuracy) для сети MobileNetV1 на датасете CIFAR-100

Заключение

ISearch работает лучше случайного поиска, как минимум для таких сетей, как ResNet-18 [12] и MobileNetV1 [13], и на двух датасетах CIFAR-10 [17] и CIFAR-100 [18]. Не смотря на то, что ISearch требует доработки и пока рано говорить об эффективности данного метода для произвольных наборов данных, результаты, представленные в данной статье, позволяют утверждать, что можно использовать наши знания о структуре нейронных сетей, чтобы создать новый метод автоматического подбора гиперпараметров, который, возможно, обретет бо́льшую популярность по сравнению с известными методами за счет интерактивности и за счет того, что выбор точек для обучения часто может совпадать с тем, как бы для данной модели подбирали гиперпараметры пользователи, если бы делали это самостоятельно.

ЛИТЕРАТУРА

1. *Feurer M, Hutter F.* AutoML Book. – Hyperparameter Optimization, 2019.

2. *Bergstra J., Bengio Y.* Random search for hyper-parameter optimization // Journal of machine learning research. – 2012. – 13 (2).
3. *Brochu E., Cora V.M., De Freitas N.* A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning // CoRR, abs/1012.2599. – 2010.
4. *Baydin A.G., Cornish R., Martınez-Rubio D., Schmidt M., Wood F.D.* Online learning rate adaptation with hypergradient descent // CoRR, abs/1703.04782. – 2017.
5. *Simon D.* Evolutionary optimization algorithms. – John Wiley & Sons, 2013.
6. *Whitley D.* A genetic algorithm tutorial // Statistics and computing. – 1994. – 4 (2). – P. 65–85.
7. *Li L., Jamieson K., DeSalvo G., Rostamizadeh A., Talwalkar. A.* Hyperband: A novel bandit-based approach to hyperparameter optimization // The Journal of Machine Learning Research. – 2017. – 18(1). – P. 6765–6816.
8. *Krizhevsky A., Sutskever I., Hinton G.E.* Imagenet classification with deep convolutional neural networks // Advances in neural information processing systems. – 2012. – 25. – P. 1097–1105.
9. *Loshchilov I., Hutter F.* Fixing weight decay regularization in adam. – CoRR, abs/1711.05101. – 2017.
10. *Polyak B.T.* Comparison of convergence rate of one-step and multistep optimization algorithms in the presence of noise. – 1977.
11. *Ma J., Yarats D.* Quasi-hyperbolic momentum and adam for deep learning. – CoRR, abs/1810.06801. – 2018.
12. *He K., Zhang X., Ren Sh., Sun J.* Deep residual learning for image recognition. – CoRR, abs/1512.03385. – 2015.
13. *Howard A.G., Zhu M., Chen B., Kalenichenko D., Wang W., Weyand T., Andreetto M., Adam H.* Mobilenets: Efficient convolutional neural networks for mobile vision applications. – CoRR, abs/1704.04861. – 2017.
14. *Schmidt R.M., Schneider F., Hennig P.* Descending through a crowded valley – benchmarking deep learning optimizers/ – 2021.
15. *Li Y., Wei. C., Ma T.* Towards explaining the regularization effect of initial large learning rate in training neural networks // arXiv preprint arXiv:1907.04595. – 2019.
16. *Gitman I., Lang H., Zhang P., Xiao L.* Understanding the role of momentum in stochastic gradient methods. arXiv preprint arXiv:1910.13962. – 2019.
17. *Krizhevsky A., Nair V., Hinton G.* Cifar-10 (Canadian Institute for Advanced Research)
18. *Krizhevsky A., Nair V., Hinton G.* Cifar-100 (Canadian Institute for Advanced Research).

DOI: 10.17223/978-5-907572-27-0-2022-33

АЛГОРИТМ ИЗМЕНЕНИЯ СООТНОШЕНИЯ СТОРОН ВИДЕО

Гитман Д.А., Пахомова Е.Г.
Томский государственный университет
dgtm2001@gmail.com, pakhomovaeg@yandex.ru

Введение

Сегодня для каждой социальной сети требуется, чтобы соотношения сторон загруженных видео соответствовали установленным для данной платформы правилам: для Instagram это – 4:5, для TikTok – 9:16, а для YouTube – 16:9, и, более того, на разных смартфонах эффективное соотношение сторон показываемого видео может отличаться. Чтобы профессионально кадрировать видео под конкретное соотношение, нужен специалист, который будет этим заниматься. Это долгий и энергозатратный процесс, а если учитывать растущее количество разных устройств, для каждого из которых надо адаптировать видеоконтент, процесс становится непосильным.

Есть два наиболее популярных способа решения проблемы изменения соотношения сторон видео, предоставляемые многими бесплатными онлайн ресурсами. Первый способ – добавление черных полей по краям до нужного соотношения. Однако при таком подходе эффективное разрешение контента уменьшается и черные поля даже могут оказаться больше самого видео. Второй способ заключается в обрезке видео до нужного соотношения – для этого пользователю предлагается выбрать положение окна обрезки, которое будет зафиксировано для всех кадров. Если привлекающий внимание контент находится в одной области кадра на протяжении всего видео, такой метод работает хорошо. В противном случае при данном подходе будет потеряна важная информация, находящаяся за пределами выбранного окна обрезки. Чтобы избежать этого, можно разрешить окну обрезки двигаться вместе с движением объекта за которым мы наблюдаем, имитируя таким образом движение камеры.