Стохастические квазиньютоновские методы оптимизации в контексте глубоких нейронных сетей.

А. Жогов[a], М. Сысак[a], Б. Усеинов[a]

[a]*Московский физико-технический институт (национальный исследовательский университет), Москва, Россия*

Ключевые слова: Квазиньютоновские методы, Машинное обучение, Глубинное обучение, Глубокие нейронные сети, SGD, SGD-Momentum, Adam, LBFGS, Barzilai-Borwein, Задача оптимизации

1. Введение

В данной статье рассматриваются стохастические квазиньютоновские методы для решения задачи классификации с помощью глубоких нейронных сетей. Квазиньютоновские методы являются широко используемыми методами в задачах выпуклой оптимизации [1, 2, 3]. При этом, для обучения нейронных сетей в качестве базового выбора алгоритма их используют редко. Это связано с тем, что в задачах глубинного обучения вычислительно затратно использовать детерминированные методы оптимизации, а стохастические версии квазиньютоновских методов обладают свойством неустойчивости [4, 5].

В задачах классификации, в частности, изображений, нейронные сети показывают результаты [6, 7], превосходящие результаты классических методов[1] машинного обучения. Время обучения и получаемая точность решения зависят по большей части от выбора метода оптимизации, который используется для минимизации функции потерь 2. В связи с этим

Email addresses: zhogov.aa@phystech.edu (А. Жогов), sysak.ma@phystech.edu (М. Сысак), useinov.br@phystech.edu (Б. Усеинов)

[1]Такие методы как деревья решений, наивная байесовская классификация, линейная регрессия, логистичекая регрессия, градиентный бустинг и др.

предлагается использовать для решения задачи оптимизации стохастические модификации квазиньютоновских методов вместо стохастических методов первого порядка, традиционно применяемых к задачам обучения нейронных сетей.

Рассматриваются два стохастических квазиньютоновских метода, один из которых является модификацией LBFGS [2] и один — модификацией BB [1] (подробнее о модификациях в разделах 1.1, 3). Проведено сравнение стохастических квазиньютоновских методов со стохастическими методами первого порядка — SGD, SGD-Momentum [8] и Adam [9]. Сравнение методов проведено на задачах классификации MNIST [10] с использованием полносвязной нейронной сети архитектуры 1 и CIFAR-10 [11] с использованием сверточной нейронной сети архитектуры 3. Для сравнения методов использованы конечное значение точности (accuracy) на отложенной выборке и время, затрачиваемое на фиксированное число итераций.

В данной работе будет проверена работоспособность стохастических квазиньютоновских методов в задачах обучения нейронных сетей. В результате методы будут отранжированы по описанным выше метрикам на каждой рассматриваемой задаче оптимизации, что позволит сделать вывод о целесообразности их использования.

1.1. Обзор литературы

Стохастический градиентный спуск (SGD) был предложен в работе [12]. В последующих работах были получены оценки его сходимости для выпуклых [13] и для невыпуклых [14] целевых функций. Основными недостатками этого метода являются сублинейная сходимость и возможная сходимость к седловой точке, а не к локальному минимуму. Для решения этих проблем были предложены методы, описываемые ниже.

В качестве решения проблемы сублинейной скорости сходимости SGD выступал метод тяжелого шарика [15], альтернативное название которого — SGD-Momentum, что связано с распространенной физической интерпретацией. Его идея состоит в добавлении дополнительного слагаемого, играющего роль инерции движения. Доказано [16], что в сильно выпуклых задачах оптимизации он имеет асимптотически ту же скорость сходимости, что и градиентный спуск, но с меньшей константой. В [8] обсуждается идея SGD-Momentum и его превосходство над SGD в задачах оптимизации, возникающих при использовании нейронных сетей в машинном обучении. Дальнейшей модификацией, которая обла-

дает асимптотически более быстрой сходимостью по итерациям в задачах выпуклой оптимизации, стал ускоренный градиентный метод [17], идея которого состоит в вычислении градиента в точке, смещенной относительно текущей. Его стохастическая модификация рассматривается в работе [18]. Его достоинства по сравнению с предыдущими модификациями и обсуждение его идеи также приводятся в [8].

Для решения проблемы сходимости к седловым точкам основным подходом является покомпонентное масштабирование градиентов. Первым известным подходом стал AdaGrad, предложенный в [19]. Его идея состоит в масштабировании каждой компоненты градиента на корень из суммы квадратов значений этой компоненты по всем итерациям. Его главная проблема следует из монотонного возрастания этой суммы — с ростом числа итераций шаг по всем компонентам стремится к нулю. Эта проблема была частично решена в алгоритме RMSProp [20], в котором сумма была заменена на скользящее экспоненциальное среднее. Идеи алгоритмов RMSProp и Momentum совместил в себе Adam, представленный в [9].

Квазиньютоновские методы чаще используются в задачах выпуклой оптимизации и на практике показывают сверхлинейную скорость сходимости [3]. В 1987 году был предложен квазиньютоновский метод BFGS [2], в котором на каждой итерации на основе информации о кривизне целевой функции строится плотная матрица, являющаяся оценкой обратного гессиана. В контексте нейронных сетей недостаток этого метода состоит в неустойчивости при применении стохастической модификации напрямую, что вызвано использованием информации о кривизне различных функций на каждой итерации. Подробно эта проблема обсуждается в [4]. Для решения этой проблемы предлагалось множество различных решений [21, 22, 23], одно из которых [23] будет рассмотрено в данной работе.

Другим квазиньютоновским методом является метод Barzilai-Borwein (BB), предложенный в [1]. В данном методе приближение обратного гессиана строится подбором подходящего размера шага на основе информации о кривизне целевой функции. Как и в случае с BFGS, при переходе к стохастической модификации получаемый размер шага становится неустойчивым, что показано в [5]. Существует несколько подходов к решению этой проблемы [5, 24], в данной статье будет рассмотрена одна из них [24].

2. Постановка задачи

В данной работе будут рассмотрены методы, описанные в 3, в контексте задачи классификации. По набору данных $\mathcal{X} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$, $\mathbf{x}_i \in \mathbb{R}^d$ — вектор признаков i-го объекта, а $y_i \in \mathbb{Y}$ — его метка, строится такое отображение $f\colon \mathbb{R}^n \to \mathbb{Y}$, чтобы наиболее точно выполнялось $\forall\, i\ f(x_i) \approx y_i$. Здесь $\mathbb{Y} = \{1, \ldots, c\}$, где c — количество классов. Классический подход состоит в введении параметрического семейства функций $f\colon \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{Y}'$, где $\mathbb{Y}' = \{\mathbf{p} \in [0,1]^c \mid \sum_{i=1}^c p_i = 1\}$ — пространство вероятностных распределений над $\mathbb{Y}$, а m — размерность вектора параметров. В рассматриваемой задаче f представляет из себя глубокую нейронную сеть, а $\theta \in \mathbb{R}^m$ — вектор ее параметров. Архитектуры сетей, рассмотренных в этой работе, описаны в разделе 4. Для подбора вектора θ вводится функция потерь $\mathcal{L}\colon\ \mathbb{Y}' \times \mathbb{Y} \to \mathbb{R}$. В данной статье используется перекрестная энтропия

$$\mathcal{L}(\mathbf{p}, y) = -\log p_y \tag{1}$$

Минимизация такой функции требует максимизации вероятности верного класса p_y, поэтому она может выступать как количественная мера качества рассматриваемой функции f в контексте задачи классификации. Оптимальное значение параметра задается как решение оптимизационной задачи

$$\min_\theta F_{\mathcal{X}}(\theta) \triangleq \frac{1}{n} \sum_{i=1}^n \mathcal{L}(f(x_i, \theta), y_i). \tag{2}$$

Рассматриваемый класс функций накладывает ограничения на применимые для решения задачи (2) методы. Целевая функция является дифференцируемой почти всюду, но не является выпуклой, поскольку функция f, задаваемая глубокой нейронной сетью, в общем случае невыпукла. Кроме того, часто размер выборки n достаточно велик, что делает вычисление градиента $\nabla F_{\mathcal{X}}$ на каждой итерации численного метода вычислительно затратной задачей. Для ускорения градиентных методов вместо точного значения градиента используют его несмещенную оценку. На каждой итерации случайно формируется подмножество исходной выборки $\mathcal{X}' = \{(x_{i_j}, y_{i_j})\}_{j=1}^s \subset \mathcal{X}$, $s \ll n$ (далее такие подмножества будем называть минибатчами), и градиент считается по вспомогательной функции

$$F_{\mathcal{X}'}(\theta) \triangleq \frac{1}{s} \sum_{j=1}^s \mathcal{L}(f(x_{i_j}, \theta), y_{i_j}), \tag{3}$$

то есть вместо $\nabla F_{\mathcal{X}}(\theta)$ используется $\nabla F_{\mathcal{X}'}(\theta)$. Как легко видеть, среднее значение функций $F_{\mathcal{X}'}$ по всем минибатчам совпадает с исходной функцией $F_{\mathcal{X}}$. Таким образом, вспомогательная функция является случайной величиной, матожидание которой равно исходной целевой функции. Из этого следует, что, при вычислении градиента по вспомогательной функции, градиентные методы будут в среднем вести себя так, как при вычислениях над $F_{\mathcal{X}}$. Строгое доказательство этого факта для алгоритма SGD, рассмотренного в 3.1, приведено в [13]. В силу вероятностной природы методов, производящих вычисления над $F_{\mathcal{X}'}$ вместо $F_{\mathcal{X}}$, к их названиям принято добавлять слово «стохастический». Далее в этой работе этот термин означает, что метод вычисляет градиент по $F_{\mathcal{X}'}$. Еще одно ограничение на применимые к задаче (2) методы вытекает из большой размерности пространства параметров $\mathbb{R}^m$, что делает невозможным применение методов второго порядка, оперирующих полным гессианом целевой функции $\nabla^2 F_{\mathcal{X}}(\theta)$ размерности $m \times m$. Эта проблема частично может быть решена применением квазиньютоновских методов, позволяющих сократить расходы по памяти и время работы по сравнению с методами второго порядка.

3. Методы

В этой статье будет рассмотрено два варианта стохастических модификаций квазиньютоновских методов, применяемых в глубинном обучении. Для сравнения будет использовано два стохастических градиентных метода.

Рассматриваемые ниже методы будут сравниваться двумя способами: на фиксированной задаче будет рассматриваться значение некоторой метрики качества (подробнее в разделе 4), достигнутой за фиксированное число эпох, а также будет измеряться время, затрачиваемое на фиксированное число эпох оптимизации. Эпохой называют группу из числа итераций, равного отношению размера всей выборки $|\mathcal{X}|$ к размеру минибатча $|\mathcal{X}'|$. Такая группировка итераций используется для более устойчивой оценки целевой функции, в качестве которой на каждой эпохе вычисляется среднее значений $F_{\mathcal{X}'}(\theta)$ по всем итерациям этой эпохи. Такая оценка обладает меньшей дисперсией, чем каждое конкретное значение $F_{\mathcal{X}'}(\theta)$.

3.1. Стохастический градиентный спуск (SGD)

На каждой итерации генерируется минибатч $\mathcal{X}'$, после чего параметры обновляются по формуле

$$\theta_{k+1} = \theta_k - \eta_k \nabla F_{\mathcal{X}'}(\theta_k) \tag{4}$$

Здесь последовательность длин шага $\{\eta_k\}$ — параметр алгоритма. В данной статье используется постоянный шаг $\eta_k \equiv \eta$.

Каждая итерация алгоритма требует $O(m)$ времени (без учета времени вычисления градиента), $O(m)$ памяти для хранения градиента $\nabla F_{\mathcal{X}'}(\theta)$ и 1 вычисление градиента.

Помимо SGD в виде 4, в данной работе рассматривается его модификация, называемая методом тяжелого шарика (Heavy Ball Method), или SGD-Momentum. На каждой итерации генерируется минибатч $\mathcal{X}'$, после чего целевая переменная обновляется по правилу

$$\theta_{k+1} = \theta_k - \eta \nabla F_{\mathcal{X}'}(\theta_k) + \beta(\theta_k - \theta_{k+1}) \tag{5}$$

Анализ сходимости этого метода приведен в [16], где показано, что в некотором классе целевых функций он имеет меньшую константу линейной сходимости, нежели 4.

Время работы и расходы по памяти этой модификации алгоритма совпадают с SGD асимптотически, но на самом деле памяти и времени требуется в два раза больше (без учета вычисления градиента), поскольку надо хранить два значения градиента вместо одного. Аналогично SGD, на каждой итерации градиент вычисляется 1 раз.

3.2. Adam

В алгоритме 1 описан псевдокод метода Adam, представленного в [9]. В этом методе на каждой итерации обновляются скользящие экспоненциальные средние, то есть величины вида

$$\overline{x}_{k+1} = \gamma \overline{x}_k + (1 - \gamma) x_{k+1}$$

для коэффициента $\gamma \in (0, 1)$. Такие значения поддерживаются алгоритмом для градиента с коэффициентом β_1 и для покомпонентного квадрата градиента с коэффициентом β_2. В статье [9] используется $\beta_1 = 0.9$ и $\beta_2 = 0.999$. В оригинальной статье было показано, что полученные величины являются оценками градиента и его квадрата по всей выборке,

смещенными на коэффициенты $(1 - \beta_1^{t+1})$ и $(1 - \beta_2^{t+1})$ соответственно. Поэтому для получения несмещенных оценок следующим шагом обе величины делятся на соответствующие им коэффициенты. Наконец, вектор параметров обновляется на величину $-\eta \cdot m_{t+1}/(\sqrt{v_{t+1}} + \varepsilon)$. Поправка ε здесь нужна для численной стабильности на случай близких к нулю градиентов. В статье [9] используется $\varepsilon = 10^{-8}$. От SGD такое правило отличается тем, что вместо градиента теперь используется скользящее среднее градиентов по всем итерациям, покомпонентно деленное на корень из скользящего среднего их квадратов. Основная мотивация состоит в том, что подобное масштабирование позволяет увеличить шаг вдоль направлений с маленьким градиентом, предотвращая сходимость к седловым точкам. Данный вопрос подробно рассматривается в [25] на примере алгоритма RMSprop [20], который отличается от Adam тем, что вместо экспоненциального среднего градиентов в нем используется значение градиента на текущей итерации.

Algorithm 1: Adam
Здесь все операции с векторами покомпонентные;
β_i^{t+1} обозначает возведение числа β_i в степень $t+1$, $i \in \{1, 2\}$.

Input : Число итераций N
Размер шага η
Экспоненциальные скорости затухания $\beta_1, \beta_2 \in [0, 1)$
Начальное приближение θ_0
Поправка на численную неустойчивость ε
Output: Итоговое значение параметра θ_N
$m_0 \leftarrow 0$
$v_0 \leftarrow 0$
for $t = 0, 1, \ldots, N-1$ **do**
  Сгенерировать минибатч $\mathcal{X}'_t \subset \mathcal{X}$
  $g_{t+1} \leftarrow \nabla F_{\mathcal{X}'_t}(\theta_t)$
  $m_{t+1} \leftarrow \beta_1 \cdot m_t + (1 - \beta_1) \cdot g_{t+1}$
  $v_{t+1} \leftarrow \beta_2 \cdot v_t + (1 - \beta_2) \cdot g_{t+1}^2$
  $\hat{m}_{t+1} \leftarrow m_{t+1}/(1 - \beta_1^{t+1})$
  $\hat{v}_{t+1} \leftarrow v_{t+1}/(1 - \beta_2^{t+1})$
  $\theta_{t+1} \leftarrow \theta_t - \eta \cdot \hat{m}_{t+1}/(\sqrt{\hat{v}_{t+1}} + \varepsilon)$
end
return θ_N

Данный метод имеет те же асимптотики по времени и памяти и тре-

бует такого же количества вычислений градиента, что и SGD.

3.3. SGD-BB

Главной идеей стандартного алгоритма BB является расчет размера шага с использованием информации о кривизне графика целевой функции. В данном методе гессиан целевой функции аппроксимируется матрицей $B_k = \eta_k^{-1} I$, где η_k^{-1} определяется как решение квазиньютоновского уравнения

$$B_k s_k = y_k,$$

которое сводится к одномерной задаче минимизации

$$\min_{\eta_t} \|\eta_t^{-1} s_t - y_t\|_2^2,$$

решение которой в явной форме имеет вид

$$\eta_t = \frac{\|s_t\|_2^2}{s_t^T y_t}.$$

Здесь $s_t = \theta_{t+1} - \theta_t$ и $y_t = \nabla F_{\mathcal{X}}(\theta_{t+1}) - \nabla F_{\mathcal{X}}(\theta_t)$.

Модификация SGD-BB, представленная в [24] и псевдокод которой показан в процедуре 2, использует идею выделения минибатчей. Внутри эпохи k фиксируется размер шага η_k, после чего совершается T итераций, в течение которых вектор параметров (k — индекс эпохи, t — индекс итерации внутри эпохи) обновляется по правилу:

$$\theta_{k,0} = \theta_{k-1,T}, \;\; \theta_{k,t+1} = \theta_{k,t} - \eta_k \nabla F_{\mathcal{X}'_{k,t}}(\theta_{k,t})$$

Градиент $g_{k,t}$, использующийся для нахождения η_k, оценивается как и в методе Adam 1 при помощи экспоненциального скользящего среднего

$$g_{k,t+1} = (1-\beta) g_{k,t} + \beta \nabla F_{\mathcal{X}'_{k,t}}(\theta_{k,t})$$

В отличие от оригинального метода, разность градиентов между эпохами считается как

$$y_k = g_{k,T} - g_{k-1,T},$$

а разность векторов параметров между двумя эпохами определяется как

$$s_k = T^{-1}(\theta_{k,T} - \theta_{k-1,T}).$$

Определенные так s_k, y_k дают размер шага следующей эпохи

$$\eta_{k+1} = \frac{\|s_k\|_2^2}{|s_k^T y_k|} \tag{6}$$

В отличие от оригинального BB, в знаменателе стоит модуль скалярного произведения. Это объясняется тем, что для детерминированной (то есть, не стохастической) версии алгоритма строго доказано [26], что $s_k^\top y_k > 0$ на каждой итерации. В стохастическом случае это не гарантируется, в том числе не гарантируется численная устойчивость (6), для чего требуется коррекция полученного размера шага. Значения η_k корректируются при помощи заданных параметров $\tau_0, \tau_{\min}, \tau_{\max}$

$$\hat{\eta}_k = \begin{cases} \eta_k & \text{если } \eta_k \in \left[\frac{\tau_{\min}}{k+1}, \frac{\tau_{\max}}{k+1}\right], \\ \frac{\tau_0}{k+1} & \text{иначе.} \end{cases}$$

Algorithm 2: SGD-BB

Input : Число эпох K
Число шагов в эпохе T
Размер шага $\eta_0 = \eta_1$
Экспоненциальная скорость затухания $\beta \in (0, 1]$
Начальное приближение $\theta_{0,0}$
Корректирующие $\hat{\eta}_k$ параметры τ_0, $\tau_{\min}$, $\tau_{\max}$

Output: Итоговое значение $\theta_{K,0}$

for $k = 0, \ldots, K-1$ **do**
  if $k > 1$ **then**
    $y_{k-1} \leftarrow g_{k-1,T} - g_{k-2,T}$
    $s_{k-1} \leftarrow \frac{1}{T}(\theta_{k-1,T} - \theta_{k-2,T})$
    $\eta_k \leftarrow \frac{\|s_{k-1}\|^2}{|s_{k-1}^T y_{k-1}|}$
    $\hat{\eta}_k = \begin{cases} \eta_k & \text{если } \eta_k \in \left[\frac{\tau_{\min}}{k+1}, \frac{\tau_{\max}}{k+1}\right], \\ \frac{\tau_0}{k+1} & \text{иначе.} \end{cases}$
  end
  $g_{k,0} \leftarrow 0$
  for $t = 0, \ldots, T-1$ **do**
    Сгенерировать минибатч $\mathcal{X}'_{k,t} \subset \mathcal{X}$
    $\theta_{k,t+1} \leftarrow \theta_{k,t} - \hat{\eta}_k \nabla F_{\mathcal{X}'_{k,t}}(\theta_{k,t})$
    $g_{k,t+1} = (1-\beta) g_{k,t} + \beta \nabla F_{\mathcal{X}'_{k,t}}(\theta_{k,t})$
  end
  $\theta_{k+1,0} \leftarrow \theta_{k,T}$
end
return $\theta_{K,0}$

В своих экспериментах с набором данных CIFAR-10 [11] и архитектурой VGG [27], авторы [24] использовали следующие значения:

$$T = 400, \quad \beta = 0.01, \quad \tau_0 = 0.1, \quad \tau_{\min} = 0.1, \quad \tau_{\max} = 10.$$

Каждая итерация внутреннего цикла алгоритма требует $O(m)$ памяти и времени (без учета вычисления градиента) и 1 вычисления градиента. Вне этого цикла на каждой эпохе происходят вычисления с теми же затратами по памяти и времени, но без вычислений градиента.

Algorithm 3: two_loop_recursion
(Вычисление произведения $H_k\mathbf{x}$ по последним p парам (y, s) [3])

Input : Вектор $\mathbf{x}$
Набор пар $\{(y_i, s_i)\}_{i=k-p}^{k-1}$
Output: Произведение $H_k\mathbf{x}$
for $i = k-1, k-2, \ldots, k-p$ **do**
  $\alpha_i \leftarrow s_i^\top \mathbf{x} / s_i^\top y_i$
  $\mathbf{x} \leftarrow \mathbf{x} - \alpha_i y_i$
end
$\mathbf{r} \leftarrow \dfrac{s_{k-1}^\top y_{k-1}}{y_{k-1}^\top y_{k-1}}\mathbf{x}$
for $i = k-p, k-p+1, \ldots, k-1$ **do**
  $\beta \leftarrow \dfrac{y_i^\top \mathbf{r}}{s_i^\top y_i}$
  $\mathbf{r} \leftarrow \mathbf{r} + s_i(\alpha_i - \beta)$
end
return r

Основная идея алгоритма Multi-Batch LBFGS, предложенного в [23], состоит в построении на каждой итерации оценки обратного гессиана H_k и обновлении вектора параметров по правилу

$$\theta_{k+1} = \theta_k - \eta H_k \nabla F_{\mathcal{X}'_k}(\theta_k),$$

где $\mathcal{X}'_k$ — минибатч. В отличие от BB, здесь приближение обратного гессиана H_k строится таким образом, чтобы оно строго удовлетворяло квазиньютоновскому уравнению

$$s_{k+1} = H_{k+1} y_{k+1},$$

где

$$y_{k+1} = \nabla F_{\mathcal{X}'_{k+1} \cap \mathcal{X}'_k}(\theta_{k+1}) - \nabla F_{\mathcal{X}'_{k+1} \cap \mathcal{X}'_k}(\theta_k) \quad s_{k+1} = \theta_{k+1} - \theta_k$$

В дальнейшем будем обозначать $O_k = \mathcal{X}'_{k+1} \cap \mathcal{X}'_k$. Такие условия накладывают ограничение на процедуру генерации батчей: требуется, чтобы два батча на соседних итерациях имели пересечение фиксированного размера $|O_k| = l < s = |\mathcal{X}'_k|$. Это требование обусловлено тем, что на практике при вычислении градиентов для y_{k+1} по разным подмножествам выборки $\mathcal{X}$ оценка обратного гессиана H_k становится неустойчивой [4].

Multi-Batch LBFGS хранит последние p пар (y_i, s_i), и по ним вычисляет произведение $H_k \nabla F_{\mathcal{X}'_k}(\theta_k)$ с помощью процедуры two_loop_recursion, вынесенной в алгоритм 3 и описанной в [3]. После каждой итерации пара (y_{k-p}, s_{k-p}) заменяется на новую пару (y_k, s_k). Псевдокод всего метода приведен в алгоритме 4.

Algorithm 4: Multi-Batch LBFGS

Input : Число итераций N
Начальное приближение θ_0
Размер памяти p
Размер пересечения минибатчей l
Размер шага η

Output: Итоговое значение параметра θ_N

Сгенерировать минибатч $\mathcal{X}'_0$
$C \leftarrow \varnothing$
for $k = 0, 1, \ldots, N-1$ **do**
  $g_k \leftarrow \nabla F_{\mathcal{X}'_k}(\theta_k)$
  $p_k \leftarrow \text{two\_loop\_recursion}(g_k, C)$
  $\theta_{k+1} \leftarrow \theta_k - \eta \cdot p_k$
  Создать минибатч $\mathcal{X}'_{k+1}$
  $O_k \leftarrow \mathcal{X}'_{k+1} \cap \mathcal{X}'_k$
  $s_{k+1} \leftarrow \theta_{k+1} - \theta_k$
  $y_{k+1} \leftarrow \nabla F_{O_k}(\theta_{k+1}) - \nabla F_{O_k}(\theta_k)$
  if $k \geq p$ **then**
    $C \leftarrow C \cup \{(y_{k+1}, s_{k+1})\} \setminus \{(y_{k+1-p}, s_{k+1-p})\}$
  else
    $C \leftarrow C \cup \{(y_{k+1}, s_{k+1})\}$
  end
end

В статье [23] утверждается, что p целесообразно выбирать из отрезка $[2, 20]$, отношение размера пересечения минибатчей к размеру одного минибатча l/s может быть любым числом из интервала $(0, 0.5)$, выбор авторов оригинальной статьи — 0.25.

Каждая итерация алгоритма требует $O(mp)$ времени (без учета вычисления градиента) и памяти, а также трех вычислений градиента.

4. Эксперименты

В данном разделе будут рассмотрены два примера применения методов оптимизации, описанных в разделе 3, к задаче классификации, описанной в разделе 2. В каждом эксперименте для конкретной задачи строится и обучается нейронная сеть подходящей архитектуры. В качестве метрики качества обученной модели используется доля правильных ответов на отложенной выборке $\mathcal{X}_{\text{test}}$, с данными из которой модель не взаимодействовала во время обучения. Эту величину будем обозначать $\mathcal{A}_s$ (от слова accuracy), где s — число эпох обучения.

4.1. MNIST

В этом эксперименте метод SGD-BB сравнивается с SGD-momentum и Adam на примере оптимизации полносвязной нейронной сети 1 на наборе данных MNIST [10]. Задача состоит в определении цифры на изображении (Рис.1) размера 28×28.

Таблица 1: Архитектура нейронной сети в эксперименте с MNIST

№	Слой
0: вход	28×28 изображение, преобразованное в вектор размерности 784
1:	Полносвязный, 128 нейронов, входных связей 784, с активацией ReLU
2:	Полносвязный, 64 нейрона, входных связей 128, с активацией ReLU
3: выход	Полносвязный, 10 нейронов, входных связей 64, с активацией LogSoftMax

Таблица 2: Результаты 15 эпох обучения нейронной сети в эксперименте с MNIST

Метод	Время, 15 эпох	$\mathcal{A}_5$	$\mathcal{A}_{10}$	$\mathcal{A}_{15}$
SGD	**2.81 мин**	89.72	91.46	92.69
SGD-momentum	3.10 мин	95.81	**97.19**	97.30
Adam	3.80 мин	**96.89**	97.17	**97.57**
SGD-BB	3.33 мин	91.57	95.11	97.01

Рис. 1: Примеры изображений в эксперименте с MNIST

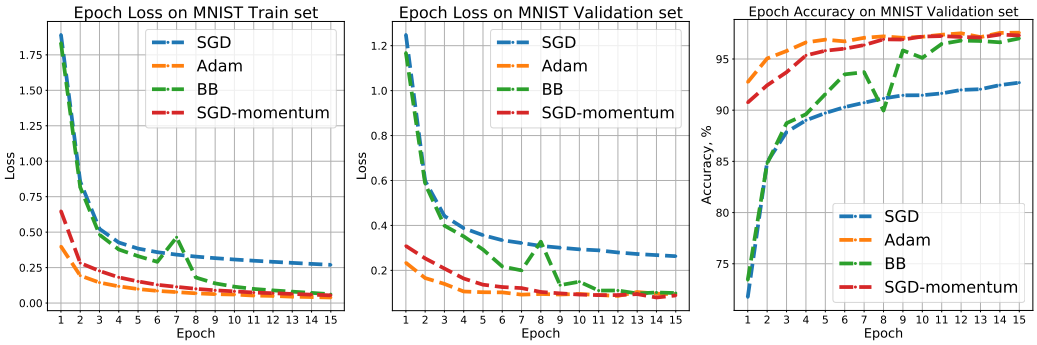

Рис. 2: Графики сходимости методов в эксперименте с MNIST

По времени все четыре алгоритма практически идентичны. Adam работает немного дольше, поскольку он делает больше вычислений на каждой итерации. По достигаемой метрике качества Adam и SGD-Momentum показывают близкие к идентичным результаты, превосходя два других алгоритма на первых итерациях. На последних эпохах SGD-BB достигает сравнимого с ними качества. SGD-BB оказывается более точным, чем SGD, что показывает эффективность процедуры выбора шага, описанной в алгоритме 2. Полученные результаты согласуются с результатами, приведенными в статье [24].

4.2. CIFAR-10

В этом эксперименте MultiBatch LBFGS и SGD-BB сравниваются с Adam и SGD при оптимизации сверточной нейронной сети 3 на данных CIFAR-10 [11]. Этот набор данных состоит из изображений (Рис.3) размера 32×32, которые необходимо классифицировать на 10 классов.

Таблица 3: Архитектура нейронной сети в эксперименте с CIFAR-10

№	Слой
0: вход	32×32 изображение
1:	Сверточный, 6 5×5 ядер (шаг=1), с активацией ReLU
2:	Субдискретизирующий по максимуму, окно 2×2 (шаг=2)
3:	Сверточный, 16 5×5 ядер (шаг=1), с активацией ReLU
4:	Полносвязный, 1000 нейронов, с активацией ReLU
5: выход	Полносвязный, 10 нейронов

Рис. 3: Примеры изображений в эксперименте с CIFAR-10

Как и в эксперименте 4.1, алгоритмы Adam, SGD, SGD-Momentum и SGD-BB практически не отличаются по затрачиваемому времени, в то время как MB-LBFGS работает дольше на четверть. Это объясняется тем, что этот метод требует нескольких вычислений градиента на каждой итерации. По достигаемой точности SGD показывает результат хуже остальных алгоритмов, что является распространенным явлением при использовании постоянного шага в глубоких нейронных сетях. Adam показывает лучший результат в данном эксперименте. SGD-BB,

Таблица 4: Результаты 15 эпох обучения нейронной сети в эксперименте с CIFAR-10

Метод	Время, 15 эпох	$\mathcal{A}_5$	$\mathcal{A}_{10}$	$\mathcal{A}_{15}$
LBFGS	6.37 мин	42.39	44.85	47.32
Adam	5.35 мин	**52.07**	**59.98**	**62.38**
SGD	5.19 мин	10.06	15.79	24.26
SGD-momentum	5.40 мин	39.43	47.70	51.11
SGD-BB	**5.16 мин**	28.98	47.48	54.68

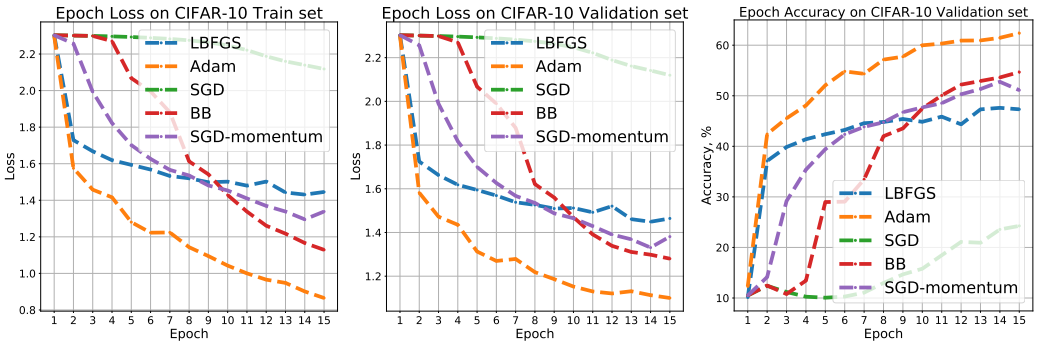

Рис. 4: Графики сходимости методов в эксперименте с CIFAR-10

SGD-Momentum и MB-LBFGS показывают схожие между собой результаты. Первый оказывается предпочтительнее остальных в силу большего значения метрики качества при меньшем времени работы. В оригинальной статье [23] MB-LBFGS сравнивается с SGD, результаты, полученные ее авторами, согласуются с результатами этого раздела.

5. Заключение

В результате экспериментов получена предпочтительность квазиньютоновских методов алгоритму SGD, так как последний сходится за большее число итераций, как на практике и происходит в задачах выпуклой оптимизации. Полученные результаты согласуются с результатами экспериментов, приведенных в [23, 24]. Квазиньютоновские методы показывают результаты, сопоставимые по качеству с результатами методов Adam и SGD-Momentum. За исключением LBFGS, который работает на 20% дольше остальных методов, все рассматриваемые методы сравнимы по затрачиваемому на фиксированное число итераций времени.

В данной работе без рассмотрения остались стохастические модификации квазиньютоновского метода DFP [28], который, однако, на практике используется меньше, чем BFGS [3]. Кроме того, остается открытым вопрос о применимости стохастической модификации метода Ньютона [3], время работы которого необходимо сравнить со временем, затрачиваемым квазиньютоновскими методами, для получения вывода о практической предпочтительности использования приближения гессиана. Задачей оптимизации, оставшейся без рассмотрения, является обучение рекуррентной нейронной сети в задаче классификации текста. К ее решению могут быть применены все описанные выше методы оптимизации.

Список литературы

[1] J. BARZILAI, J. M. BORWEIN, Two-Point Step Size Gradient Methods, IMA Journal of Numerical Analysis 8 (1988) 141–148. URL: https://doi.org/10.1093/imanum/8.1.141. doi:10.1093/imanum/8.1.141.

[2] R. Fletcher, Practical methods of optimization, number т. 1 in Wiley-Interscience publication, Wiley, 1987. URL: https://books.google.ru/books?id=3EzvAAAAMAAJ.

[3] J. Nocedal, S. Wright, Numerical optimization, Springer Science & Business Media, 2006.

[4] J. Liu, Y. Rong, M. Takac, J. Huang, On the acceleration of l-bfgs with second-order information and stochastic batches, 2018. arXiv:1807.05328.

[5] C. Tan, S. Ma, Y.-H. Dai, Y. Qian, Barzilai-borwein step size for stochastic gradient descent (2016). arXiv:1605.04131.

[6] S. Mishra, M. Kamaraj, K. Senthilkumar, Digit recognition using deep learning (2018).

[7] T. Ho-Phuoc, Cifar10 to compare visual recognition performance between deep neural networks and humans, arXiv preprint arXiv:1811.07270 (2018).

[8] S. Ruder, An overview of gradient descent optimization algorithms, arXiv preprint arXiv:1609.04747 (2016).

[9] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization (2014). arXiv:1412.6980.

[10] Y. LeCun, C. Cortes, MNIST handwritten digit database (2010). URL: http://yann.lecun.com/exdb/mnist/.

[11] A. Krizhevsky, Learning multiple layers of features from tiny images, Technical Report, 2009.

[12] J. Kiefer, J. Wolfowitz, Stochastic estimation of the maximum of a regression function, Ann. Math. Statist. 23 (1952) 462–466. URL: https://doi.org/10.1214/aoms/1177729392. doi:10.1214/aoms/1177729392.

[13] R. M. Gower, N. Loizou, X. Qian, A. Sailanbayev, E. Shulgin, P. Richtarik, Sgd: General analysis and improved rates, 2019. arXiv:1901.09401.

[14] Y. Lei, T. Hu, G. Li, K. Tang, Stochastic gradient descent for nonconvex learning without bounded gradient assumptions (2019). arXiv:1902.00908.

[15] B. T. Polyak, Введение в оптимизацию, Наука. Гл. ред. физ.-мат. лит., 1983.

[16] E. Ghadimi, H. R. Feyzmahdavian, M. Johansson, Global convergence of the heavy-ball method for convex optimization, 2014. arXiv:1412.7457.

[17] Y. E. NESTEROV, A method for solving the convex programming problem with convergence rate $o(1/k^2)$, Dokl. Akad. Nauk SSSR 269 (1983) 543–547. URL: https://ci.nii.ac.jp/naid/10029946121/en/.

[18] M. Assran, M. Rabbat, On the convergence of nesterov's accelerated gradient method in stochastic settings (2020). arXiv:2002.12414.

[19] J. Duchi, E. Hazan, Y. Singer, Adaptive subgradient methods for online learning and stochastic optimization, Journal of machine learning research 12 (2011) 2121–2159.

[20] T. Tieleman, G. Hinton, Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude, COURSERA: Neural networks for machine learning 4 (2012) 26–31.

[21] N. N. Schraudolph, J. Yu, S. Günter, A Stochastic Quasi-Newton Method for Online Convex Optimization, volume 2 of *Proceedings of Machine Learning Research*, PMLR, San Juan, Puerto Rico, 2007. URL: http://proceedings.mlr.press/v2/schraudolph07a.html.

[22] A. S. Berahas, M. Jahani, M. Takáč, Quasi-newton methods for deep learning: Forget the past, just sample (2019). arXiv:1901.09997.

[23] A. S. Berahas, J. Nocedal, M. Takáč, A multi-batch l-bfgs method for machine learning (2016). arXiv:1605.06049.

[24] J. Liang, Y. Xu, C. Bao, Y. Quan, H. Ji, Barzilai–borwein-based adaptive learning rate for deep learning, Pattern Recognition Letters 128 (2019) 197–203.

[25] M. Staib, S. J. Reddi, S. Kale, S. Kumar, S. Sra, Escaping saddle points with adaptive gradient methods, 2019. arXiv:1901.09149.

[26] O. Burdakov, Y.-H. Dai, N. Huang, Stabilized barzilai-borwein method, 2019. arXiv:1907.06409.

[27] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition (2014). arXiv:1409.1556.

[28] W. C. Davidon, Variable metric method for minimization, SIAM Journal on Optimization 1 (1991) 1–17.